Experiment: Advanced Computation for Energy Engineering

Name: Abisoye Akinloye

Table of Contents

Particle Swarm Optimization (PSO)

Subject 3: Particle Swarm Optimization (PSO)
Implement a Matlab m script of PSO to solve Rastrigin function in m-dimension:

$$Eval_i = 3m + \sum_{j=1}^{m}\left(x_{ij}^2 - 3\cos(2\pi x_{ij})\right)$$

Take 10 and 20 agents
Test it for m=2 and m=3, with visualization of agent's trajectories.

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 $x_i \in [-5.12, 5.12]$, for all $i = 1, \ldots, m$

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

PSO Algorithm

General steps

1. Initialize the algorithm parameter (ω, c_1, c_2). Algorithm parameter to be explained later
2. Initialize the population (swarm of particles) with random position x and velocity v (v can be otherwise initialized as zero)
3. Evaluate the fitness f 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 r_1 & r_2
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

- $P^{(t)}_{(i,\mathrm{lb})}$ is the personal best position of $i-\mathrm{th}$ particle in t generation. Assume minimization problem

$$p_{(i,lb)}^{(t+1)} = \begin{cases} x_i^{(t+1)} & \text{if } f(x_i^{(t+1)}) \leq f(p_{(i,lb)}^{(t)}) \\ p_{(i,lb)}^{(t)} & \text{Otherwise.} \end{cases} \qquad \textbf{equ. (a)}$$

- $P_{\mathrm{gb}}^{(t)}$ is the global best position in t generation. It is the minimum fitness which is calculated as:

$$p_{gb}^{(t)} \in \{p_{(1,best)}^{(t)}, \ldots, p_{(N,best)}^{(t)}\} \mid f(p_{gb}^{(t)}) = \min\{f(p_{(1,best)}^{(t)}), \ldots, f(p_{(N,best)}^{(t)})\}$$

$$p_{gb}^{(t)} \in \{x_1^{(t)}, \ldots, x_N^{(t)}\} \mid f(p_{gb}^{(t)}) = \min\{f(x_1^{(t)}), \ldots, f(x_N^{(t)})\} \qquad \textbf{equ. (b)}$$

- Velocity and position update

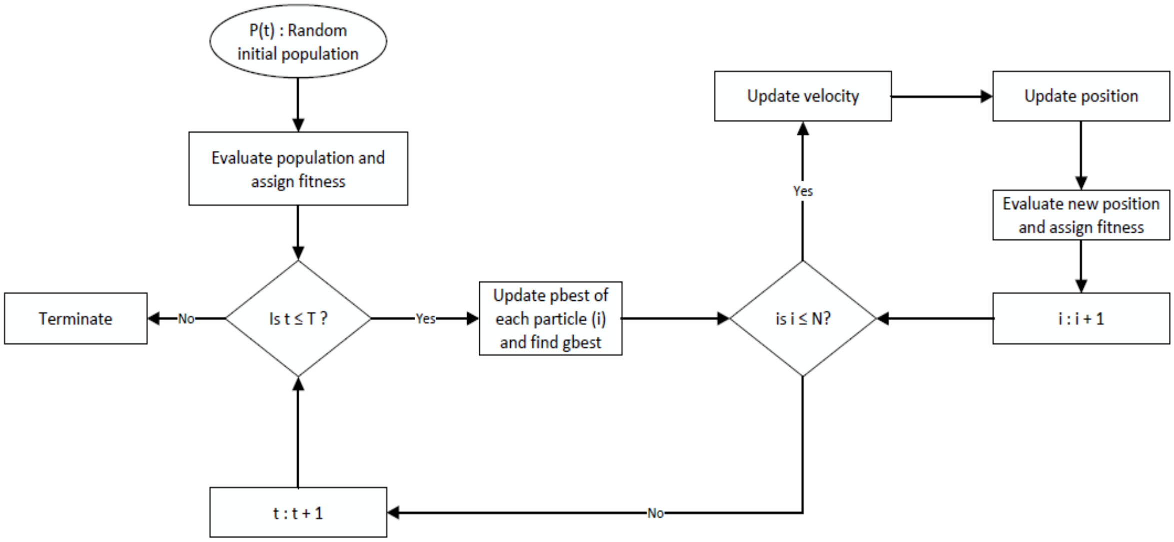
$$v_i^{t+1} = \omega \times v_i^t + c_1 r_1 \left(p_{(i,lb)}^{(t)} - x_i^{(t)}\right) + c_2 r_2 \left(p_{gb}^{(t)} - x_i^{(t)}\right) \qquad \textbf{equ. (c)}$$

$$x_i^{t+1} = x_i^t + v_i^{t+1} \qquad \textbf{equ. (d)}$$

Where

- N is the number of particles in the swarm
- lb is local (personal) best, gb is the global best
- i is the number of iteration
- ω is the inertia weight
- c_1 & c_2 are the acceleration coefficients
- r_1 & r_2 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



PSO Flowchart
Source: Module 4 Evolution Computation by Nptel

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/
- https://www.researchgate.net/publication/379603938_Particle_Swarm_and_Genetic_Algorithms_applied_to_the_identification_of_Induction_Machine_Parameters

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!
```

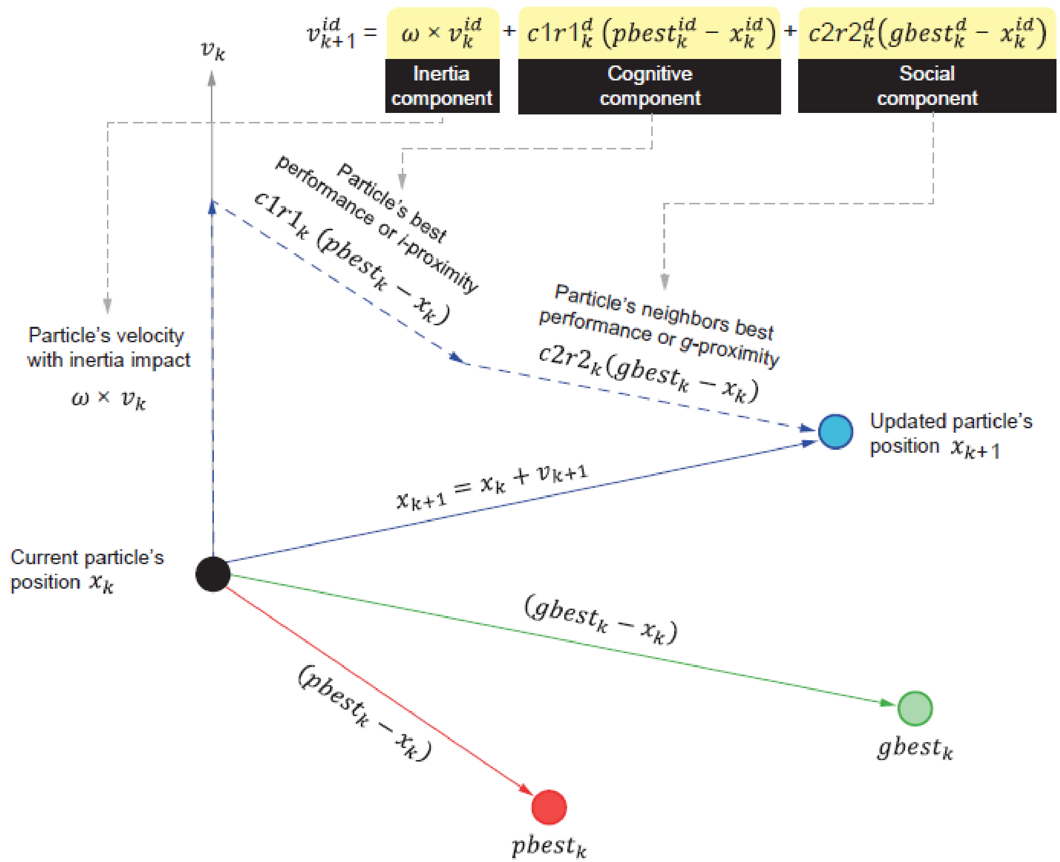
PSO hyperparameters

```
%% 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 (ω):** 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,:));
```

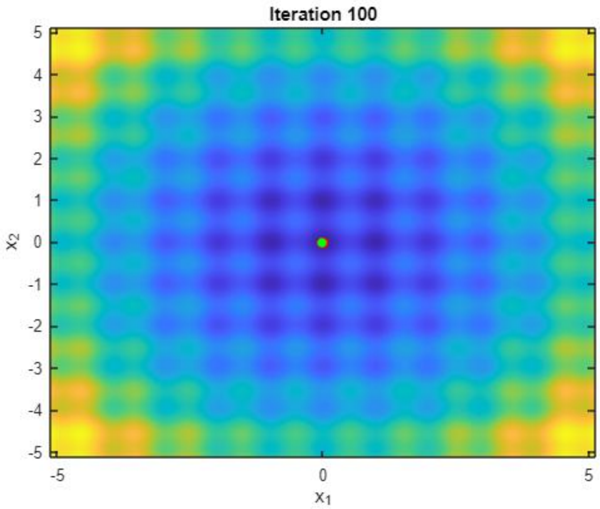
```
        % update the position
        x(i,:) = x(i,:) + v(i,:);

        % 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
```

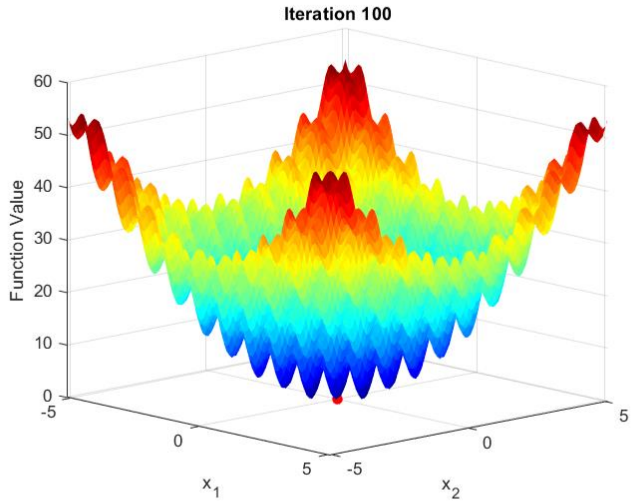
```
Iteration 1        | Global Best Value = 5.159142
Iteration 2        | Global Best Value = 2.398222
Iteration 3        | Global Best Value = 1.204072
Iteration 4        | Global Best Value = 1.204072
Iteration 5        | Global Best Value = 1.204072
Iteration 6        | Global Best Value = 1.204072
Iteration 7        | Global Best Value = 1.204072
Iteration 8        | Global Best Value = 1.204072
Iteration 9        | Global Best Value = 1.204072
Iteration 10       | Global Best Value = 1.204072
Iteration 11       | Global Best Value = 1.204072
Iteration 12       | Global Best Value = 0.201617
Iteration 13       | Global Best Value = 0.201617
Iteration 14       | Global Best Value = 0.201617
Iteration 15       | Global Best Value = 0.014447
Iteration 16       | Global Best Value = 0.014447
Iteration 17       | Global Best Value = 0.014447
Iteration 18       | Global Best Value = 0.014447
Iteration 19       | Global Best Value = 0.014447
Iteration 20       | Global Best Value = 0.014447
Iteration 21       | Global Best Value = 0.014447
Iteration 22       | Global Best Value = 0.014447
Iteration 23       | Global Best Value = 0.011929
Iteration 24       | Global Best Value = 0.011929
Iteration 25       | Global Best Value = 0.006460
Iteration 26       | Global Best Value = 0.006460
Iteration 27       | Global Best Value = 0.006460
Iteration 28       | Global Best Value = 0.006460
Iteration 29       | Global Best Value = 0.000836
Iteration 30       | Global Best Value = 0.000836
Iteration 31       | Global Best Value = 0.000836
Iteration 32       | Global Best Value = 0.000836
Iteration 33       | Global Best Value = 0.000836
Iteration 34       | Global Best Value = 0.000371
Iteration 35       | Global Best Value = 0.000371
```

```
Iteration 36        | Global Best Value = 0.000357
Iteration 37        | Global Best Value = 0.000357
Iteration 38        | Global Best Value = 0.000357
Iteration 39        | Global Best Value = 0.000357
Iteration 40        | Global Best Value = 0.000313
Iteration 41        | Global Best Value = 0.000313
Iteration 42        | Global Best Value = 0.000313
Iteration 43        | Global Best Value = 0.000313
Iteration 44        | Global Best Value = 0.000040
Iteration 45        | Global Best Value = 0.000040
Iteration 46        | Global Best Value = 0.000040
Iteration 47        | Global Best Value = 0.000040
Iteration 48        | Global Best Value = 0.000040
Iteration 49        | Global Best Value = 0.000005
Iteration 50        | Global Best Value = 0.000005
Iteration 51        | Global Best Value = 0.000000
Iteration 52        | Global Best Value = 0.000000
Iteration 53        | Global Best Value = 0.000000
Iteration 54        | Global Best Value = 0.000000
Iteration 55        | Global Best Value = 0.000000
Iteration 56        | Global Best Value = 0.000000
Iteration 57        | Global Best Value = 0.000000
Iteration 58        | Global Best Value = 0.000000
Iteration 59        | Global Best Value = 0.000000
Iteration 60        | Global Best Value = 0.000000
Iteration 61        | Global Best Value = 0.000000
Iteration 62        | Global Best Value = 0.000000
Iteration 63        | Global Best Value = 0.000000
Iteration 64        | Global Best Value = 0.000000
Iteration 65        | Global Best Value = 0.000000
Iteration 66        | Global Best Value = 0.000000
Iteration 67        | Global Best Value = 0.000000
Iteration 68        | Global Best Value = 0.000000
Iteration 69        | Global Best Value = 0.000000
Iteration 70        | Global Best Value = 0.000000
Iteration 71        | Global Best Value = 0.000000
Iteration 72        | Global Best Value = 0.000000
Iteration 73        | Global Best Value = 0.000000
Iteration 74        | Global Best Value = 0.000000
Iteration 75        | Global Best Value = 0.000000
Iteration 76        | Global Best Value = 0.000000
Iteration 77        | Global Best Value = 0.000000
Iteration 78        | Global Best Value = 0.000000
Iteration 79        | Global Best Value = 0.000000
Iteration 80        | Global Best Value = 0.000000
Iteration 81        | Global Best Value = 0.000000
Iteration 82        | Global Best Value = 0.000000
Iteration 83        | Global Best Value = 0.000000
Iteration 84        | Global Best Value = 0.000000
Iteration 85        | Global Best Value = 0.000000
Iteration 86        | Global Best Value = 0.000000
Iteration 87        | Global Best Value = 0.000000
Iteration 88        | Global Best Value = 0.000000
Iteration 89        | Global Best Value = 0.000000
Iteration 90        | Global Best Value = 0.000000
Iteration 91        | Global Best Value = 0.000000
Iteration 92        | Global Best Value = 0.000000
```

```
Iteration 93        | Global Best Value = 0.000000
Iteration 94        | Global Best Value = 0.000000
Iteration 95        | Global Best Value = 0.000000
Iteration 96        | Global Best Value = 0.000000
Iteration 97        | Global Best Value = 0.000000
Iteration 98        | Global Best Value = 0.000000
Iteration 99        | Global Best Value = 0.000000
Iteration 100       | Global Best Value = 0.000000
```



Rastrigin 2D



Rastrigin 3D

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));
```

```
Best position found: [0.000000, 0.000000]
```

```
elseif m == 3
    fprintf('Best position found: [%f, %f, %f]\n', gbest_pose(1),
gbest_pose(2), gbest_pose(3));
```

```
end
```

PSO Simulation Video

Click here: https://drive.google.com/drive/folders/16-pilizYJNG8Ki1unexyFoG23p-lYtej?usp=sharing

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: https://github.com/Abisoyesam/Computation-for-Energy/tree/main/PSO