# Particle Swarm Optimization: Algorithm and its Codes in MATLAB

**Research** · March 2016

DOI: 10.13140/RG.2.1.4985.3206



1 author:

Mahamad Nabab Alam
Indian Institute of Technology Roorkee

# Particle Swarm Optimization: Algorithm and its Codes in MATLAB

Mahamad Nabab Alam[a]

[a] *Department of Electrical Engineering, Indian Institute of Technology, Roorkee-247667, India*

**Abstract**

In this work, an algorithm for classical particle swarm optimization (PSO) has been discussed. Also, its codes in MATLAB environment have been included. The effectiveness of the algorithm has been analyzed with the help of an example of three variable optimization problem. Also, the convergence characteristic of the algorithm has been discussed.

*Keywords:* Algorithm, Codes, MATLAB, Particle swarm optimization, Program.

## 1. Introdunction

Particle swarm optimization is one of the most popular nature-inspired metaheuristic optimization algorithm developed by James Kennedy and Russell Eberhart in 1995 [1, 2]. Since its development, namy variates have also been develop for solving practical issues of related to optimization [3, 4, 5, 6, 7, 8, 9, 10]. Recently, PSO has emerged as a promising algorithm in solving various optimization problems in the field of science and engineering [11, 12, 13, 14, 15, 16, 17, 18, 19, 20, 21, 22, 23, 24, 25].

## 2. Particle Swarm Optimization: *Algorithm* [25]

Particle swarm optimization (PSO) is inspired by social and cooperative behavior displayed by various species to fill their needs in the search space. The algorithm is guided by personal experience (Pbest), overall experience (Gbest) and the present movement of the particles to decide their next positions in the search space. Further, the experiences are accelerated by two factors $c_1$ and $c_2$, and two random numbers generated between [0, 1] whereas the present movement is multiplied by an inertia factor $w$ varying between $[w_{min}, w_{max}]$.

The initial population (swarm) of size $N$ and dimension $D$ is denoted as $\mathbf{X} = [\mathbf{X}_1,\mathbf{X}_2,...,\mathbf{X}_N]^T$, where $'T'$ denotes the transpose operator. Each individual (particle) $\mathbf{X}_i$ $(i = 1, 2, ..., N)$ is given as $\mathbf{X}_i$=$[X_{i,1}, X_{i,2}, ..., X_{i,D}]$. Also, the initial velocity of the population is denoted as $\mathbf{V} = [\mathbf{V}_1,\mathbf{V}_2,...,\mathbf{V}_N]^T$. Thus, the velocity of each particle $\mathbf{X}_i$ $(i = 1, 2, ..., N)$ is given as $\mathbf{V}_i$=$[V_{i,1}, V_{i,2}, ..., V_{i,D}]$. The index $i$ varies from 1 to $N$ whereas the index $j$ varies from 1 to $D$. The detailed algorithms of various methods are described below for completeness.

*Mahamad Nabab Alam
*Email address:* itsmnalam@gmail.com (Mahamad Nabab Alam)

$$V_{i,j}^{k+1} = w \times V_{i,j}^{k} + c_1 \times r_1 \times (Pbest_{i,j}^{k} - X_{i,j}^{k}) + c_2 \times r_2 \times (Gbest_{j}^{k} - X_{i,j}^{k}) \quad (1)$$

$$X_{i,j}^{k+1} = X_{i,j}^{k} + V_{i,j}^{k+1} \quad (2)$$

In eqn. (1) $Pbest_{i,j}^{k}$ represents personal best $j^{th}$ component of $i^{th}$ individual, whereas $Gbest_{j}^{k}$ represents $j^{th}$ component of the best individual of population upto iteration $k$. Figure (1) shows the search mechanism of PSO in multidimensional search space.

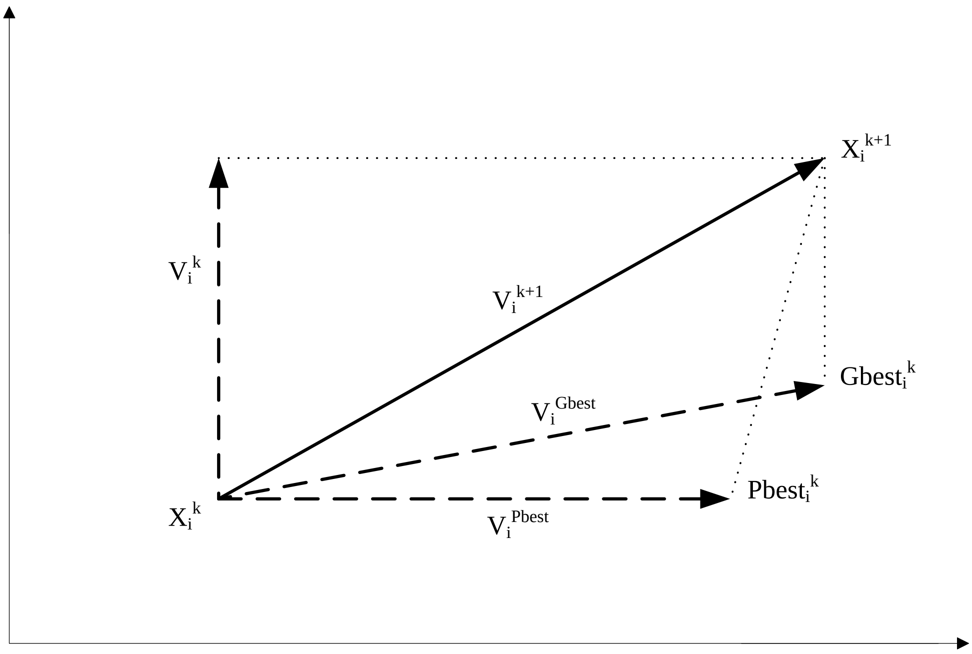


Figure 1: PSO search mechanism in multidimentional search space.

The different steps of PSO are as follows [25]:

1. Set parameter $w_{min}$, $w_{max}$, $c_1$ and $c_2$ of PSO
2. Initialize population of particles having positions $\mathbf{X}$ and velocities $\mathbf{V}$
3. Set iteration $k = 1$
4. Calculate fitness of particles $F_i^k = f(\mathbf{X}_i^k), \forall i$ and find the index of the best particle $b$
5. Select $\mathbf{Pbest}_i^k = \mathbf{X}_i^k, \forall i$ and $\mathbf{Gbest}^k = \mathbf{X}_b^k$
6. $w = w_{max} - k \times (w_{max} - w_{min})/Maxite$
7. Update velocity and position of particles
   $V_{i,j}^{k+1} = w \times V_{i,j}^{k} + c_1 \times rand() \times (Pbest_{i,j}^{k} - X_{i,j}^{k}) + c_2 \times rand() \times (Gbest_{j}^{k} - X_{i,j}^{k})$; $\forall j$ and $\forall i$
   $X_{i,j}^{k+1} = X_{i,j}^{k} + V_{i,j}^{k+1}$; $\forall j$ and $\forall i$
8. Evaluate fitness $F_i^{k+1} = f(\mathbf{X}_i^{k+1}), \forall i$ and find the index of the best particle $b1$
9. Update Pbest of population $\forall i$
   If $F_i^{k+1} < F_i^k$ then $\mathbf{Pbest}_i^{k+1} = \mathbf{X}_i^{k+1}$ else $\mathbf{Pbest}_i^{k+1} = \mathbf{Pbest}_i^k$

10. Update Gbest of population

    If $F_{b1}^{k+1} < F_b^k$ then $\mathbf{Gbest}^{k+1} = \mathbf{Pbest}_{b1}^{k+1}$ and set $b = b1$ else $\mathbf{Gbest}^{k+1} = \mathbf{Gbest}^k$

11. If $k < Maxite$ then $k = k + 1$ and goto step 6 else goto step 12

12. Print optimum solution as $\mathbf{Gbest}^k$

The most commonly used parameters of PSO algorithm are considered as follows:

- Inertial weight: 0.9 to 0.4
- Acceleration factors ($c_1$ and $c_2$): 2 to 2.05
- Population size: 10 to 100
- Maximum iteration (Maxite): 500 to 10000
- Initial velocity: 10 % of position

A detailed flowchart of PSO considering the above steps is shown in Figure (2).

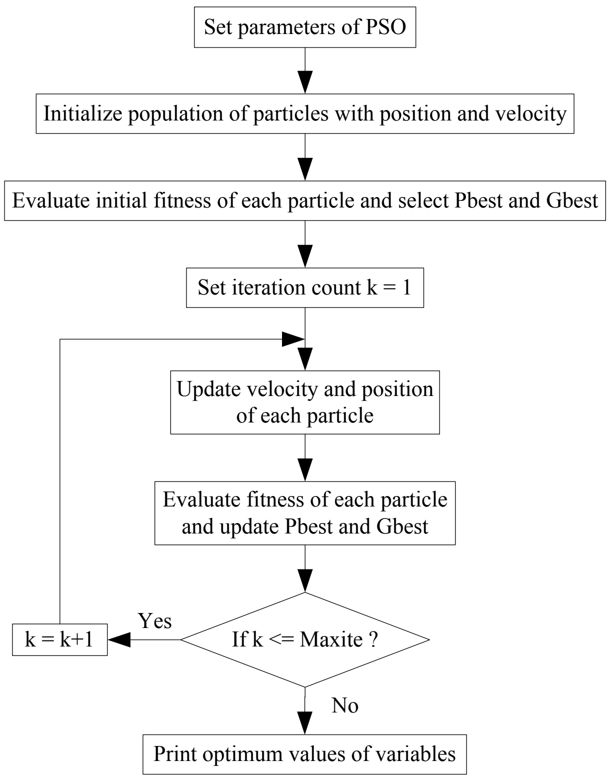


Figure 2: Flowchart of PSO.

## 3. Problem Formulation of Optimization Problem: An example

The objective function (OF) required here must be a minimization problem. Let us suppose that there is a problem defined below is need to be optimized. The OF is of minimization type subjected to inequality type constraints. The statement of the problem is as follows:

$$OF = \min\ [\ 10 \times (1 - x(1))^2 + 20 \times (2 - x(2))^2 + 30 \times (3 - x(3))^2] \tag{3}$$

Subjected to

$$x(1) + x(2) + x(3) <= 5 \tag{4}$$

$$x(1)^2 + 2 \times x(2) <= x(3) \tag{5}$$

The optimum value of the $OF$ (defined by Eq. 3) is 9.3941 at $x(1) = 0.4377$, $x(2) = 1.4569$ and $x(3) = 3.1054$. In the next section, MATLAB codes are developed for solving this problem.

## 4. Particle Swarm Optimization: *Codes in MATLAB environment*

Two MATLAB script files (*. m file) are needed to fully write the codes. In the first file, the objective function is defined, whereas in the second file, the main PSO program is developed [26].

Now, this problem will be solved by using the PSO algorithm. The objective function file and main program file can be written as follows:

### *4.1. Objective function file*

The problem defined in the last section can be expressed in MATLAB script file (*.m) as follows:

```
function f=ofun(x)

c0=[];

% objective function (minimization)
of=10*(x(1)-1)^2+20*(x(2)-2)^2+30*(x(3)-3)^2;

% constraints (all constraints must be converted into <=0 type)
% if there is no constraints then comments all c0 lines below
c0(1)=x(1)+x(2)+x(3)-5;      % <=0 type constraints
c0(2)=x(1)^2+2*x(2)-x(3);    % <=0 type constraints

% defining penalty for each constraint
for i=1:length(c0)
    if c0(i)>0
        c(i)=1;
    else
        c(i)=0;
    end
end

penalty=10000; % penalty on each constraint violation

f=of+penalty*sum(c); % fitness function
```

Save the above codes as ofun.m. The "ofun.m" file defines the problem discussed above. In main program file this function will be called again and again as per the requirement.

### *4.2. Main program file*

The main program file can be expressed as folows:

```
tic
clc
clear all
close all
rng default

LB=[0 0 0];          %lower bounds of variables
UB=[10 10 10];       %upper bounds of variables

% pso parameters values
m=3;                % number of variables
n=100;              % population size
wmax=0.9;           % inertia weight
wmin=0.4;           % inertia weight
c1=2;               % acceleration factor
c2=2;               % acceleration factor

% pso main program----------------------------------------------------start
maxite=1000;      % set maximum number of iteration
maxrun=10;        % set maximum number of runs need to be
for run=1:maxrun
    run
    % pso initialization----------------------------------------------start
    for i=1:n
        for j=1:m
            x0(i,j)=round(LB(j)+rand()*(UB(j)-LB(j)));
        end
    end
    x=x0;           % initial population
    v=0.1*x0;       % initial velocity
    for i=1:n
        f0(i,1)=ofun(x0(i,:));
    end
    [fmin0,index0]=min(f0);
    pbest=x0;                   % initial pbest
    gbest=x0(index0,:);         % initial gbest
    % pso initialization------------------------------------------------end

    % pso algorithm---------------------------------------------------start
    ite=1;
    tolerance=1;
    while ite<=maxite && tolerance>10^-12

        w=wmax-(wmax-wmin)*ite/maxite; % update inertial weight

        % pso velocity updates
        for i=1:n
            for j=1:m
                v(i,j)=w*v(i,j)+c1*rand()*(pbest(i,j)-x(i,j))...
                        +c2*rand()*(gbest(1,j)-x(i,j));
            end
        end

        % pso position update
        for i=1:n
            for j=1:m
                x(i,j)=x(i,j)+v(i,j);
            end
        end

```

```matlab
        % handling boundary violations
        for i=1:n
            for j=1:m
                if x(i,j)<LB(j)
                    x(i,j)=LB(j);
                elseif x(i,j)>UB(j)
                    x(i,j)=UB(j);
                end
            end
        end

        % evaluating fitness
        for i=1:n
            f(i,1)=ofun(x(i,:));
        end

        % updating pbest and fitness
        for i=1:n
            if f(i,1)<f0(i,1)
                pbest(i,:)=x(i,:);
                f0(i,1)=f(i,1);
            end
        end

        [fmin,index]=min(f0);   % finding out the best particle
        ffmin(ite,run)=fmin;    % storing best fitness
        ffite(run)=ite;         % storing iteration count

        % updating gbest and best fitness
        if fmin<fmin0
            gbest=pbest(index,:);
            fmin0=fmin;
        end

        % calculating tolerance
        if ite>100;
            tolerance=abs(ffmin(ite-100,run)-fmin0);
        end

        % displaying iterative results
        if ite==1
            disp(sprintf('Iteration    Best particle    Objective fun'));
        end
        disp(sprintf('%8g  %8g          %8.4f',ite,index,fmin0));
        ite=ite+1;
    end
    % pso algorithm-----------------------------------------------------end
    gbest;
    fvalue=10*(gbest(1)-1)^2+20*(gbest(2)-2)^2+30*(gbest(3)-3)^2;
    fff(run)=fvalue;
    rgbest(run,:)=gbest;
    disp(sprintf('---------------------------------------'));
end
% pso main program------------------------------------------------------end
disp(sprintf('\n'));
disp(sprintf('*********************************************************'));
disp(sprintf('Final Results-----------------------------'));
[bestfun,bestrun]=min(fff)
best_variables=rgbest(bestrun,:)
disp(sprintf('*********************************************************'));
```

```
toc

% PSO convergence characteristic
plot(ffmin(1:ffite(bestrun),bestrun),'-k');
xlabel('Iteration');
ylabel('Fitness function value');
title('PSO convergence characteristic')
%###########################################################################
```

Subsections 4.1 and 4.2 represent the complete codes for PSO to solve the optimization problems defined in Section 3. Now, to solve the problem using PSO, it is only required to run the main program file developed in subsection 4.2. It is to be noted that the same codes are availabe in my older work [26] uploaded on the ResearchGate.

## 5. Particle Swarm Optimization: *Optimized results of the problem considered*

The snapshot of the optimized solution of the problem is shown in Figure 3. From this figure, it is observed that the best value of the objective function obtained after 10 independent runs is 9.3941. This value is obtained at $x(1) = 0.4377$, $x(2) = 1.4569$ and $x(3) = 3.1054$. Out of the 10 runs, $7^{th}$ run gives this best results. The simulation total time taken is 22.9852 seconds. It is to be noted that the simulation time depends on computer configuration. Further, Figure 4 shows the convergence characteristic of PSO.

```
--------------------------------------


*********************************************************
Final Results-----------------------------

bestfun =

    9.3941


bestrun =

     7


best_variables =

    0.4377    1.4569    3.1054

*********************************************************
Elapsed time is 22.985214 seconds.
>>
```

Figure 3: Results displayed in command window of MATLAB.

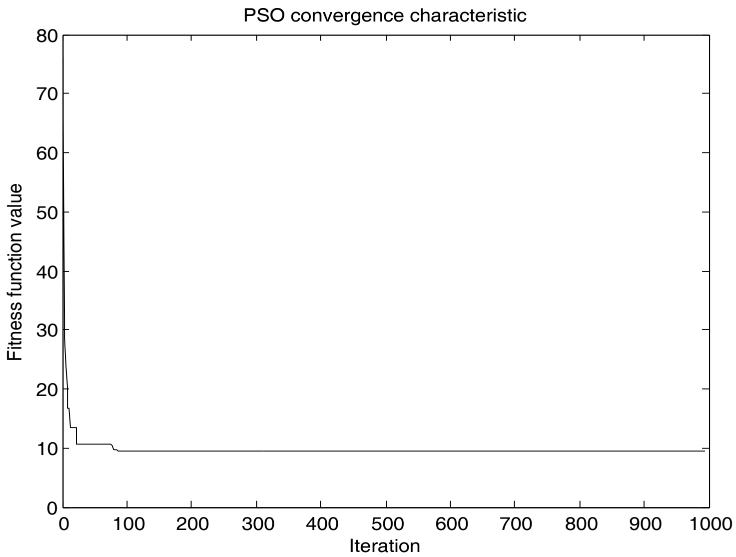


Figure 4: PSO convergence characteristic

## 6. Conclusion

In this paper, the concepts of particle swarm optimization have been discussed in a very simple way. Further, its algorithm has been developed. Also, PSO programming codes in MATLAB environment have been given and an example has been solved successfully which demonstrate the effectiveness of the algorithm. The following conclusions can be drawn from this work:

(i) The MATLAB codes discussed here can be extended to solve any type of optimization problem of any size.

(ii) Any equality constraint needs to convert into corresponding two inequality constraints.

(iii) The codes discussed here are generalized for solving any optimization problem with inequality constraints of any size.

## Appendix

If X is a set of equality constraint need to be converted into a set inequality constraint , the following procedures need to be followed:

Equality constraint:

$$X = 0 \tag{6}$$

The corresponding set of inequality constraint can be represented as follows:

$$X <= 0 \tag{7}$$

$$-X <= 0 \tag{8}$$

## References


[1] J. Kennedy, R. Eberhart, Particle swarm optimization, in: IEEE International Conference on Neural Networks, Vol. 4, 1995, pp. 1942–1948.

[2] R. Eberhart, J. Kennedy, A new optimizer using particle swarm theory, in: IEEE Proceedings of the Sixth International Symposium on Micro Machine and Human Science, 1995, pp. 39–43.

[3] R. Eberhart, Y. Shi, Comparing inertia weights and constriction factors in particle swarm optimization, in: Evolutionary Computation, 2000. Proceedings of the 2000 Congress on, Vol. 1, 2000, pp. 84–88 vol.1.

[4] M. Clerc, J. Kennedy, The particle swarm - explosion, stability, and convergence in a multidimensional complex space, IEEE Transactions on Evolutionary Computation, 6 (1) (2002) 58–73.

[5] J. J. Liang, A. K. Qin, P. N. Suganthan, S. Baskar, Comprehensive learning particle swarm optimizer for global optimization of multimodal functions, IEEE Transactions on Evolutionary Computation 10 (3) (2006) 281–295.

[6] C. A. C. Coello, G. T. Pulido, M. S. Lechuga, Handling multiple objectives with particle swarm optimization, IEEE Transactions on Evolutionary Computation 8 (3) (2004) 256–279.

[7] Y. del Valle, G. K. Venayagamoorthy, S. Mohagheghi, J. C. Hernandez, R. G. Harley, Particle swarm optimization: Basic concepts, variants and applications in power systems, IEEE Transactions on Evolutionary Computation 12 (2) (2008) 171–195. `doi:10.1109/TEVC.2007.896686`.

[8] J.-H. Seo, C.-H. Im, C.-G. Heo, J.-K. Kim, H.-K. Jung, C.-G. Lee, Multimodal function optimization based on particle swarm optimization, IEEE Transactions on Magnetics 42 (4) (2006) 1095–1098.

[9] Z. H. Zhan, J. Zhang, Y. Li, Y. H. Shi, Orthogonal learning particle swarm optimization, IEEE Transactions on Evolutionary Computation 15 (6) (2011) 832–847. `doi:10.1109/TEVC.2010.2052054`.

[10] C. Li, S. Yang, T. T. Nguyen, A self-learning particle swarm optimizer for global optimization problems, IEEE Transactions on Systems, Man, and Cybernetics, Part B (Cybernetics) 42 (3) (2012) 627–646. `doi:10.1109/TSMCB.2011.2171946`.

[11] H. Yoshida, K. Kawata, Y. Fukuyama, S. Takayama, Y. Nakanishi, A particle swarm optimization for reactive power and voltage control considering voltage security assessment, IEEE Transactions on Power Systems 15 (4) (2000) 1232–1239. `doi:10.1109/59.898095`.

[12] Z.-L. Gaing, Particle swarm optimization to solving the economic dispatch considering the generator constraints, IEEE Transactions on Power Systems, 18 (3) (2003) 1187–1195.

[13] J.-B. Park, K.-S. Lee, J.-R. Shin, K. Lee, A particle swarm optimization for economic dispatch with nonsmooth cost functions, IEEE Transactions on Power Systems, 20 (1) (2005) 34–42.

[14] M. AlRashidi, M. El-Hawary, A survey of particle swarm optimization applications in electric power systems, IEEE Transactions on Evolutionary Computation, 13 (4) (2009) 913–918.

[15] J. Robinson, Y. Rahmat-Samii, Particle swarm optimization in electromagnetics, IEEE Transactions on Antennas and Propagation 52 (2) (2004) 397–407. doi:10.1109/TAP.2004.823969.

[16] Z.-L. Gaing, A particle swarm optimization approach for optimum design of pid controller in avr system, IEEE Transactions on Energy Conversion 19 (2) (2004) 384–391.

[17] R. Abousleiman, O. Rawashdeh, Electric vehicle modelling and energy-efficient routing using particle swarm optimisation, IET Intelligent Transport Systems 10 (2) (2016) 65–72.

[18] C. Chen, S. Duan, T. Cai, B. Liu, G. Hu, Smart energy management system for optimal microgrid economic operation, IET Renewable Power Generation, 5 (3) (2011) 258–267.

[19] Binary particle swarm optimisation-based optimal substation coverage algorithm for phasor measurement unit installations in practical systems, IET Generation, Transmission Distribution 10 (2) (2016) 555–562. doi:10.1049/iet-gtd.2015.1077.

[20] Hybrid quantum particle swarm optimisation to calculate wideband green's functions for microstrip structures, IET Microwaves, Antennas Propagation 10 (3) (2016) 264–270. doi:10.1049/iet-map.2015.0169.

[21] L. Liu, F. Zhou, M. Tao, Z. Zhang, A novel method for multi-targets isar imaging based on particle swarm optimization and modified clean technique, IEEE Sensors Journal 16 (1) (2016) 97–108.

[22] K.-B. Lee, J.-H. Kim, Multiobjective particle swarm optimization with preference-based sort and its application to path following footstep optimization for humanoid robots, IEEE Transactions on Evolutionary Computation 17 (6) (2013) 755–766. doi:10.1109/TEVC.2013.2240688.

[23] Y. Fu, M. Ding, C. Zhou, Phase angle-encoded and quantum-behaved particle swarm optimization applied to three-dimensional route planning for uav, IEEE Transactions on Systems, Man, and Cybernetics - Part A: Systems and Humans 42 (2) (2012) 511–526. doi:10.1109/TSMCA.2011.2159586.

[24] M. N. Alam, T. R. Chelliah, A new sensitivity evaluation based optimization algorithm for economic load dispatch problems, in: Energy Efficient Technologies for Sustainability (ICEETS), 2013 International Conference on, 2013, pp. 995–1000. doi:10.1109/ICEETS.2013.6533522.

[25] M. N. Alam, B. Das, V. Pant, A comparative study of metaheuristic optimization approaches for directional overcurrent relays coordination, Electric Power Systems Research 128 (2015) 39–52. doi:doi:10.1016/j.epsr.2015.06.018.

[26] M. N. Alam, Codes in matlab for particle swarm optimization, ResearchGate (2016) 1–3doi:10.13140/RG.2.1.1078.7608.