**Finding the minimum of a function using Particle Swarm Optimization**

Optimization is defined as methods, algorithms, and mathematical principles to solve quantitative problems in a broad range of scientific disciplines. Problems belonging to the field of optimization can be reduced to maximizing or minimizing the value of a numerical quantity or an objective function. The objective function is simply a collection of variables or quantities whose values are maximized or minimized. PSO algorithm is a swarm intelligence based optimization technique. It is similar to evolutionary algorithms which use mechanisms such as crossover, mutation, reproduction and recombination to reach optimum solutions. PSO is a stochastic optimization technique where the system is initialized with a population of random candidate solutions called particles. It searches for the optimum value of the objective function by iteratively updating the candidate solution about a given measure of optimality in successive generations.

**Research Method:**

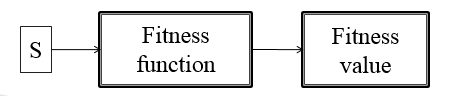
To verify the performance of the proposed PSO algorithm, a set of standard test functions are used and we will consider these parameters:

* Swarm: a set of particles (S)
* Particle: a potential solution
  + Position
  + Velocity
* Each particle maintains
  + Individual best position (PBest)
  + Swarm maintains its global best (GBest)

Basic algorithm of PSO

* 1. Initialize the swarm form the solution space
  2. Evaluate the fitness of each particle
  3. Update individual and global bests
  4. Update velocity and position of each particle
  5. Go to step 2, and repeat until termination condition

This is the blockchart of our system:



We need to update our parameters till we get the best possible value:

* Original velocity update equation

* Position update



***Example: Find the minimum of this function***

******

**Source code:**

**PSO.m**

%Finding maximum of function

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

funstr='3\*x^2-2\*x\*y+3\*y^2-x-y';

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

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

% 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---------------------------------------------------

disp(sprintf('\n'));

disp(sprintf('\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*'));

disp(sprintf('Final Results-----------------------------'));

[bestfun,bestrun]=min(fff)

best\_variables=rgbest(bestrun,:)

disp(sprintf('\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*'));

toc

% PSO convergence characteristic

figure;

plot(ffmin(1:ffite(bestrun),bestrun),'-k');

xlabel('Iteration');

ylabel('Fitness function value');

title('PSO convergence characteristic')

**ofun.m**

function f=ofun(x)

% objective function (minimization)

% x=9

% x0(i,j)=round(LB(j)+C\*(UB(j)-LB(j)))

of=10\*(x(1)-1)^2+20\*(x(2)-2)^2+30\*(x(3)-3)^2;

a=x(1);

b=x(2);

c=x(3);

% of=8\*(x-1)^2+16\*(x-2)^2+24\*(x-3)^2

% constraints (all constraints must be converted into <=0 type)

% if there is no constraints then comments all c0 lines below

c0=[] ;

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

% c0(1)=x+x+x-5 % <=0 type constraints

% c0(2)=-x^2+2\*x-x % <=0 type constraints

% c0(3)=-x^3

% defining penalty for each constraint

for i=1:length(c0)

p=i;

if c0(i)>0

q=c0(i);

c(i)=1 ;

r=c(i);

else

c(i)=0;

s=c(i);

end

end

penalty=10000;

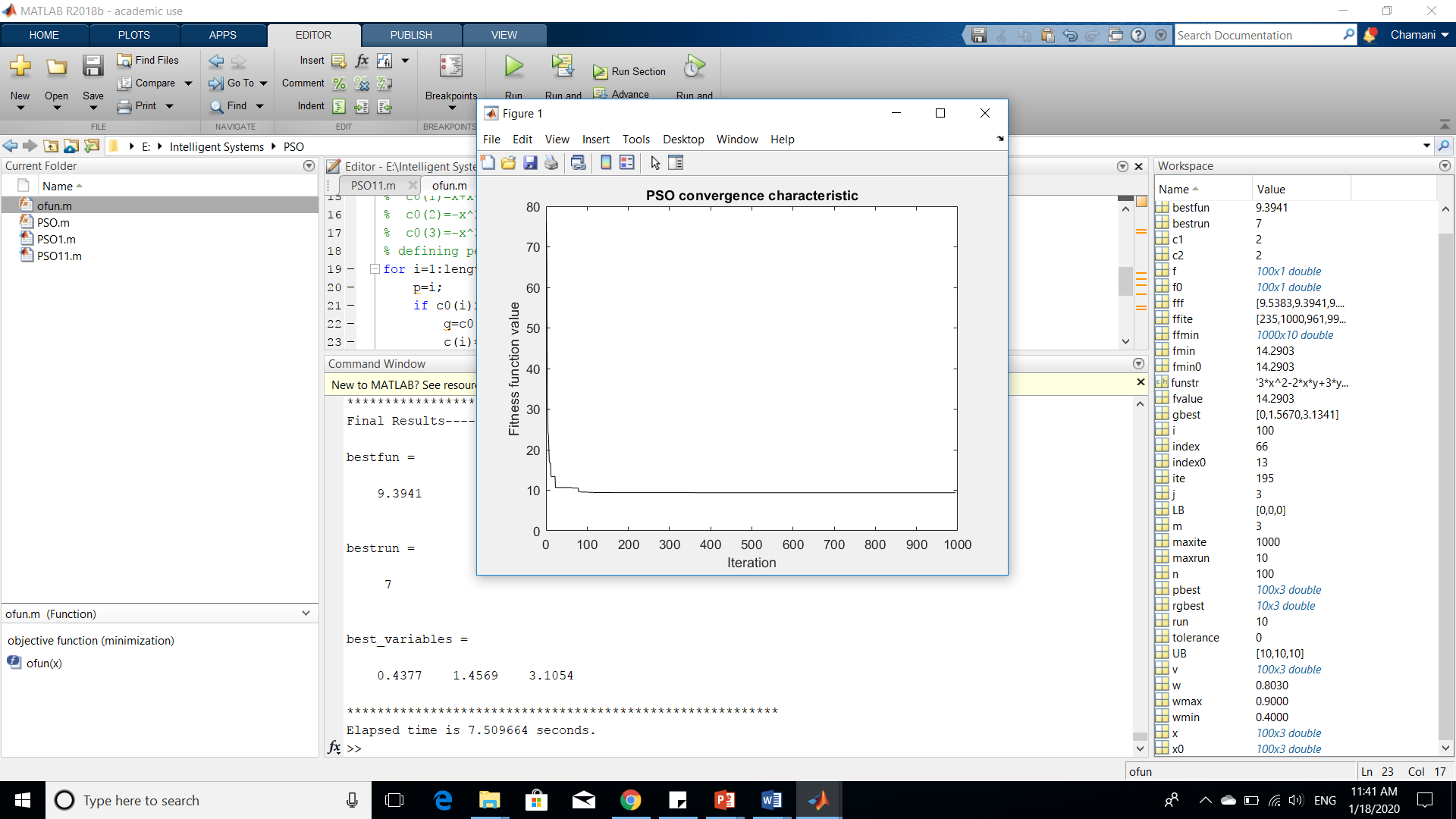
u=c;

t=sum(c);% penalty on each constraint violation

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

end

**Final Results:**



bestfun =

9.3941

bestrun =

7

best\_variables =

0.4377 1.4569 3.1054