ME 7015 Design Optimization Project

Instructor	: Assoc. Prof. Aykut Kentli
Student	: Hasan Şener
ID	:524617038

Contents

Solving Engineering Optimization Problems Particle Swarm Optimization

1. Introduction

Particle Swarm Optimization (PSO) is a computational method that optimizes a problem by iteratively trying to improve a candidate solution with regard to a given measure of quality. It solves a problem by having a population of candidate solutions (particles), and moving these particles around in the search-space according to simple mathematical formulae over the particle's position and velocity. Each particle's movement is influenced by its local best-known position, but is also guided toward the best-known positions in the search-space, which are updated as better positions are found by other particles. This is expected to move the swarm toward the best solutions.

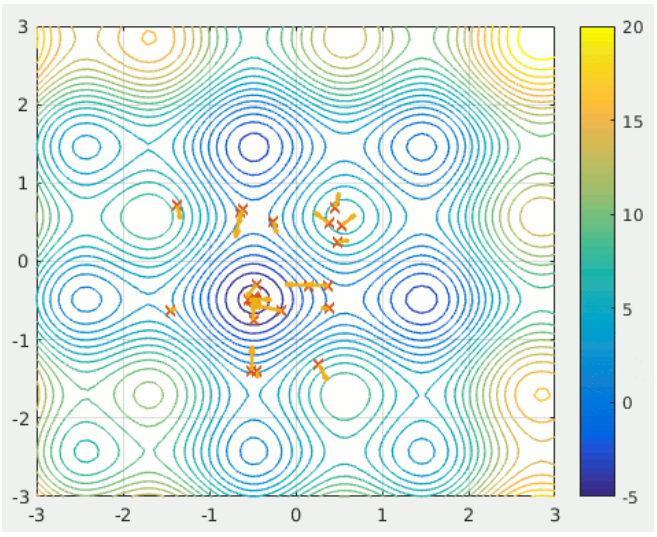

Figure 1 Particle Swarm Opimization Simulation

As you see in the Fig.1. the **x** shows the where the particles are. These are our solutions in search space. Then the **arrows** in figure are velocities of each particle.

2. The Algorithm

PSO simulates the behaviors of bird flocking. Suppose the following scenario: a group of birds are randomly searching food in an area. There is only one piece of food in the area being searched. All the birds do not know where the food is. But they know how far the food is in each iteration. So, what's the best strategy to find the food? The effective one is to follow the bird which is nearest to the food.

PSO learned from the scenario and used it to solve the optimization problems. In PSO, each single solution is a "bird" in the search space. We call it "particle". All of particles have fitness values which are evaluated by the fitness function to be optimized, and have velocities which direct the flying of the particles. The particles fly through the problem space by following the current optimum particles.

PSO is initialized with a group of random particles (solutions) and then searches for optima by updating generations. In every iteration, each particle is updated by following two "best" values. The first one is the best solution (fitness) it has achieved so far. (The fitness value is also stored.) This value is called **pbest** (particle best). Another "best" value that is tracked by the particle swarm optimizer is the best value, obtained so far by any particle in the population. This best value is a global best and called **gbest** (global best).

The PSO algorithm consists of just three steps:

1) Evaluate fitness of each particle

2) Update individual and global bests

3) Update velocity and position of each particle with respect to given equations

The pseudocode of PSO can be seen below:

```
For each particle
    Initialize particle
End For
While iterations < max iterations
    For each particle
        Calculate fitness value
        If the fitness value is better than the best fitness value (pBest)
in history
            set current value as the new pBest
    End For
* Choose the particle with the best fitness value of all the particles as
the gBest
  For each particle
    Calculate particle velocity according equation (a)
    Update particle position according equation (b)
  End For
End While
```

The equations for PSO are:

// Velocity update equation

$$v_i\ (t+1) = wv_i\ (t) + c_1 r_1[\hat{x_i}(t) - x_i(\ t)] + c_2 r_2[g(t) - x_i\ (t)]$$

i is the particle index

w is the inertial coefficient

c_1, c_2 are acceleration coefficients, $c_1 \geq 0, c_2 \leq 2$

r_1, r_2 are random values $0 \leq r_1, r_2 \leq 1$ regenerated every velocity update

$v_i(t)$ is the particle's velocity at time t

x_i is the particle's position at time t

$\hat{x_i}(t)$ is the particle's individual best solution as of time t

$g(t)$ is the swarm's best solution as of time t

Since velocity update function is known, each particle's position can be updated using the below equation:

// Position update equation

$$x_i(t+1) = x_i(t) + v_i(t+1)$$

3. MATLAB Implementation

First define the PSO parameters,

```
swarm_size = 8; % Swarm Size
no_design_variable = 4; % # of Design Variables
x_max = 3; % Maximum Position

iter_max = 30; % Maximum no. of Iterations
iter = 0; % Current Iterataion

particle_best = cell(1,swarm_size); % Particle Best Position
particle_best_objective = ones(1,swarm_size)*1E50; % Particle Best Position
Objective Value

global_best_objective = 1E50; % Global Best Position Objective Value
global_best = zeros(1,no_design_variable); % Global Best Position

particle_position = cell(1,swarm_size); % Particle Position
obj_fun_val_particle = zeros(1,swarm_size);% Objective Function Value of the
Particle

particle_velocity = cell(1,swarm_size);              % Particle_Velocity
dummy = zeros(1,length(no_design_variable));         % Dummy List
dt = 1;                                              % Time Step
```

Secondly, the problem definition will be taken into account;

```
P = 6000; % Applied Tip Load
E = 30e6; % Young Modulus
G = 12e6; % Shear Modulus
L = 14;   % Length of the Beam

PCONST = 1000000; % Penalty Function Constant
TAUMAX = 13600;   % Max Allowed Shear Stress
SIGMAX = 30000;   % Max Allowed Bending Stress
DELTMAX = 0.25;   % Max Allowed Tip Perfection

M = @(x) P*(L+x(2)/2);       % Bending Moment at Welding Point
R = @(x) sqrt((x(2)^2)/4+((x(1)+x(3))/2)^2); % Constant
J = @(x) 2*(sqrt(2)*x(1)*x(2)*((x(2)^2)/12+((x(1)+x(3))/2)^2)); % Polar Moment of
Inertia

objective_function = @(x) 1.10471*x(1)^2*x(2)+0.04811*x(3)*x(4)*(14+x(2));
sigma = @(x) (6*P*L)/(x(4)*x(3)^2); % Bending Stress
delta = @(x) (4*P*L^3)/(E*x(4)*x(3)^3); % Tip Deflection


Pc = @(x) 4.013*E*sqrt((x(3)^2*x(4)^6)/36)*(1-x(3)*sqrt(E/(4*G))/(2*L))/(L^2);  %
Buckling Load
tau_p = @(x) P/(sqrt(2)*x(1)*x(2));                                              %
Tau_prime
tau_pp = @(x) (M(x)*R(x))/J(x);                                                  %
Tau_double_prime
tau = @(x) sqrt(tau_p(x)^2+2*tau_p(x)*tau_pp(x)*x(2)/(2*R(x))+tau_pp(x)^2);      %
Tau (Shear Stress)
```

Then define the constraints $g_i(x)$:

```
% Constraints
g1 = @(x) tau(x)-TAUMAX;
g2 = @(x) sigma(x)-SIGMAX;
g3 = @(x) x(1)-x(4);
g4 = @(x) 0.10471*x(1)^2+0.04811*x(3)*x(4)*(14+x(2))-5;
g5 = @(x) 0.125-x(1);
g6 = @(x) delta(x)-DELTMAX;
g7 = @(x) P-Pc(x);
```

Define penalty function to create a barrier:

```
penalty_function = @(x) objective_function(x) +
PCONST*(max(0,g1(x))^2+max(0,g2(x))^2+max(0,g3(x))^2+...
    +max(0,g4(x))^2+max(0,g5(x))^2+...
    max(0,g6(x))^2+max(0,g7(x))^2); % Penalty Function
```

Third step is initializing particles;

```
for i = 1:swarm_size
    for j = 1:no_design_variable
        dummy(j) = rand()*x_max;
    end
    particle_position{i} = dummy;
    particle_velocity{i} = particle_position{i}/dt;
end
```

Lastly, main loop of PSO to calculate velocities & positions:

```
while iter < iter_max
    C1 = 1.8;    % personal learning factor
    C2 = 1.8;    % social learning factor
    W = 0.8;     % inertia factor
    iter = iter + 1;
    for i = 1:swarm_size
        obj_fun_val_particle(i) = penalty_function(particle_position{i});
        if obj_fun_val_particle(i) < particle_best_objective(i) &&
obj_fun_val_particle(i) >= 0  % Best Local
            particle_best{i} = particle_position{i};
            particle_best_objective(i) = obj_fun_val_particle(i);
        end
    end
    if min(obj_fun_val_particle) < global_best_objective &&
min(obj_fun_val_particle) >= 0       % Best Global
        global_best = particle_position{obj_fun_val_particle ==
min(obj_fun_val_particle)};
        global_best_objective = min(obj_fun_val_particle);
    end
    for i = 1:swarm_size    % Update Particle Positon
        R1 = rand();         % Random Number
        R2 = rand();         % Random Number
        particle_velocity{i} = W*particle_velocity{i}+C1*R1*(particle_best{i}-
particle_position{i})/dt+C2*R2*(global_best-particle_position{i})/dt; % Update
Particle Velocity
        particle_position{i} = particle_position{i}+particle_velocity{i}*dt; %
Update Particle Position
    end
    disp(['BEST PARTICLE VALUE >> ' num2str(global_best_objective)]);
end
disp(['BEST PARTICLE POSITION >> ' num2str(global_best)]);
```

4. Results

Four engineering optimization problems solved with Particle Swarm Optimization. The results for the best particles can be seen below and the script files for the problems as stated below:

- **Welded Beam Design Optimization Problem (pso_welded_beam.m)**

  In this problem the objective is to find the minimum fabrication cost, considerating four design variables: x_1, x_2, x_3, x_4 and constraints of shear stress, bending stress, buckling load and the deflection of the beam.

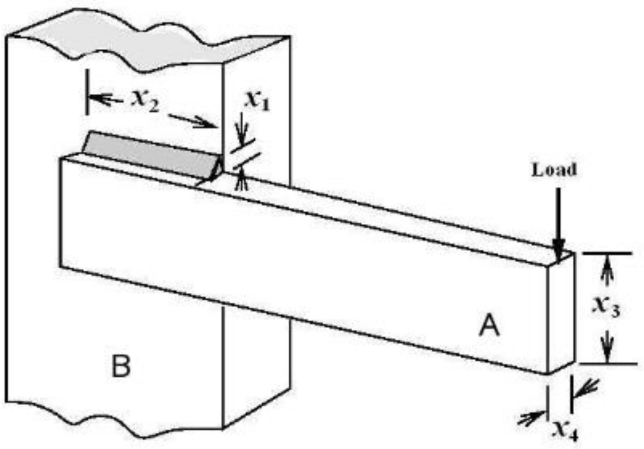

Figure 2 Welded Beam

Objective Function:

$$f(\vec{x}) = 1.10471{x_1}^2 x_2 + 0.04811 x_3 x_4 (14.0 + x_2)$$

Constraints:

$$g_1(\vec{x}) = \tau(\vec{x}) - 13,600 \leq 0$$
$$g_2(\vec{x}) = \sigma(\vec{x}) - 30,000 \leq 0$$
$$g_3(\vec{x}) = x_1 - x_4 \leq 0$$
$$g_4(\vec{x}) = 0.10471({x_1}^2) + 0.04811 x_3 x_4 (14 + x_2) - 5.0 \leq 0$$
$$g_5(\vec{x}) = 0.125 - x_1 \leq 0$$
$$g_6(\vec{x}) = \delta(\vec{x}) - 0.25 \leq 0$$

$$g_7(\vec{x}) = 6,000 - Pc(\vec{x}) \leq 0$$

with:

$$\tau(\vec{x}) = \sqrt{(\tau')^2 + (2\tau'\tau'')\frac{x_2}{2R} + (\tau'')^2}$$

$$\tau' = \frac{6,000}{\sqrt{2}x_1x_2}$$

$$\tau'' = \frac{MR}{J}$$

$$M = 6,000\left(14 + \frac{x_2}{2}\right)$$

$$R = \sqrt{\frac{x_2{}^2}{4} + \left(\frac{x_1 + x_3}{2}\right)^2}$$

$$J = 2\left\{x_1x_2\sqrt{2}\left[\frac{x_2{}^2}{12} + \left(\frac{x_1 + x_3}{2}\right)^2\right]\right\}$$

$$\sigma(\vec{x}) = \frac{504,000}{x_4x_3{}^2}$$

$$\delta(\vec{x}) = \frac{65,856,000}{(30 \times 10^6)x_4x_3{}^3}$$

$$Pc(\vec{x}) = \frac{4.013(30 \times 10^6)\sqrt{\frac{x_3{}^2x_4{}^6}{36}}}{196}\left(1 - \frac{x_3\sqrt{\frac{30\times10^6}{4(12\times10^6)}}}{28}\right)$$

with $0.1 \leq x_1, x_4 \leq 2.0$, and $0.1 \leq x_2, x_3 \leq 10.0$.

Results found in PSO are (x1, x2, x3, x4):

>> ***0.72576   1.94   3.451   1.6656***

- **Pressure Vessel Design Optimization Problem (pressure_vessel.m)**

In this problem the objective is to minimize the total cost, including the cost of materials forming the welding. Considering four design variables: x_1, x_2, x_3, x_4 which thickness, thickness of the head, the inner radius and the length of the cylindrical section of the vessel.

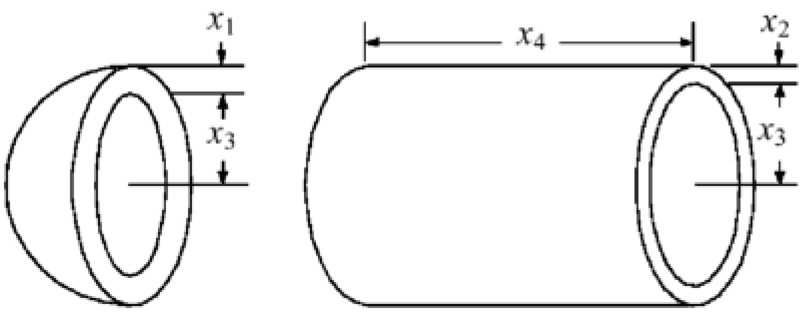

Figure 3 Pressure Vessel

The objective function and constraints are below:

$$f(\vec{x}) = 0.6224x_1x_3x_4 + 1.7781x_2{x_3}^2 + 3.1661{x_1}^2x_4 + 19.84{x_1}^2x_3$$

subject to:

$$g_1(\vec{x}) = -x_1 + 0.0193x_3 \leq 0$$

$$g_2(\vec{x}) = -x_2 + 0.00954x_3 \leq 0$$

$$g_3(\vec{x}) = -\pi {x_3}^2{x_4}^2 - \frac{4}{3}\pi {x_3}^3 + 1,296,000 \leq 0$$

$$g_4(\vec{x}) = x_4 - 240 \leq 0$$

with $1 \times 0.0625 \leq x_1, x_2 \leq 99 \times 0.0625, 10.0 \leq x_3$, and $x_4 \leq 200.0$.

Results found in PSO are (x1, x2, x3, x4):

>> 2.63319 13.4306 26.4714 23.9948

- **Speed Reducer Design Optimization Problem (speed_reducer.m)**

Considering the design variables face width x_1, module of the teeth x_2, number of teeth on pinion x_3, length of the second shaft between bearings x_4, length of the second shaft between bearings x_5, diameter of the first shaft x_6 and the diameter of the first shaft x_7. The weight of the speed reducer is to be minimized in this problem.

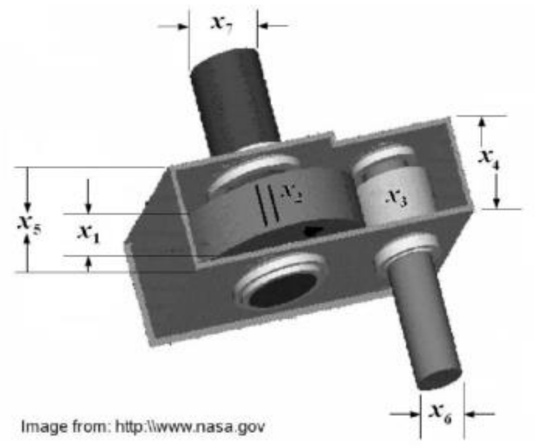

Figure 4 Speed Reducer

The objective function and constraints are:

$$f(\vec{x}) = 0.7854x_1{x_2}^2(3.3333x_3^2 + 14.9334x_3 - 43.0934) - 1.508x_1(x_6^2 + x_7^2) + 7.4777(x_6^3 + x_7^3) + 0.7854(x_4x_6^2 + x_5x_7^2)$$

$$g_1(\vec{x}) = \frac{27}{x_1x_2^2x_3} - 1 \leq 0$$

$$g_2(\vec{x}) = \frac{397.5}{x_1 x_2^2 x_3^2} - 1 \leq 0$$

$$g_3(\vec{x}) = \frac{1.93 x_4^3}{x_2 x_3 x_6^4} - 1 \leq 0$$

$$g_4(\vec{x}) = \frac{1.93 x_5^3}{x_2 x_3 x_7^4} - 1 \leq 0$$

$$g_5(\vec{x}) = \frac{1.0}{110 x_6^3} \sqrt{\left(\frac{745.0 x_4}{x_2 x_3}\right)^2 + 16.9 \times 10^6} - 1 \leq 0$$

$$g_6(\vec{x}) = \frac{1.0}{85 x_7^3} \sqrt{\left(\frac{745.0 x_5}{x_2 x_3}\right)^2 + 157.5 \times 10^6} - 1 \leq 0$$

$$g_7(\vec{x}) = \frac{x_2 x_3}{40} - 1 \leq 0$$

$$g_8(\vec{x}) = \frac{5 x_2}{x_1} - 1 \leq 0$$

$$g_9(\vec{x}) = \frac{x_1}{12 x_2} - 1 \leq 0$$

$$g_{10}(\vec{x}) = \frac{1.5 x_6 + 1.9}{x_4} - 1 \leq 0$$

$$g_{11}(\vec{x}) = \frac{1.1 x_7 + 1.9}{x_5} - 1 \leq 0$$

with $2.6 \leq x_1 \leq 3.6, 0.7 \leq x_2 \leq 0.8, 17 \leq x_3 \leq 28, 7.3 \leq x_4 \leq 8.3, 7.8 \leq x_5 \leq 8.3, 2.9 \leq x_6 \leq 3.9$, and $5.0 \leq x_7 \leq 5.5$.

Results found in PSO are (x1, x2, x3, x4,x5,x6,x7):

>> 8.24242   1.51559   15.6705   12.9359   7.92578   7.41818   5.61471

- **Tension Spring Design Optimization Problem (tension_spring.m)**

This problem minimizes the weight of tension spring subject to constraints of minimum deflection, shear stress, surge frequency, and limits on outside diameter and on design variables. There are three design variables in this problem: the wire diameter x_1, the mean coil diameter x_2 and the number of active coils x_3.

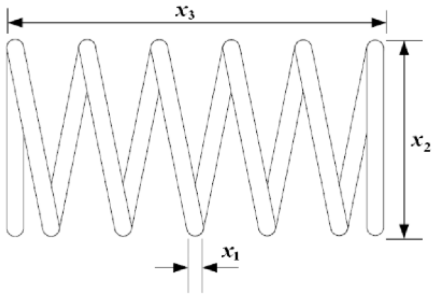

Figure 5 Tension/Compression Spring

The objective function and constraints are:

Minimize:

$$f(\vec{x}) = (x_3 + 2)x_2x_1^2$$

subject to:

$$g_1(\vec{x}) = 1 - \frac{x_2^3x_3}{7,178x_1^4} \leq 0$$

$$g_2(\vec{x}) = \frac{4x_2^2 - x_1x_2}{12,566(x_2x_1^3) - x_1^4} + \frac{1}{5,108x_1^2} - 1 \leq 0$$

$$g_3(\vec{x}) = 1 - \frac{140.45x_1}{x_2^2x_3} \leq 0$$

$$g_4(\vec{x}) = \frac{x_2 + x_1}{1.5} - 1 \leq 0$$

with $0.05 \leq x_1 \leq 2.0, 0.25 \leq x_2 \leq 1.3$, and $2.0 \leq x_3 \leq 15.0$.

Results found in PSO are (x1, x2, x3):

>> *0.055896   0.51442   2.4015*

5. Discussion

The Particle Swarm Optimization bases swarm intelligence. The particles (solutions) were initialized randomly at search space then while changing each particle's direction (velocity) every particle holds its best value. Then the best value of all personal best values is taken as global best value (swarm best value).

To accomplish swarm optimization below mathematical equations are used:

$$v_i\,(t+1) = w v_i\,(t) + c_1 r_1[\widehat{x_i}(t) - x_i(\,t)] + c_2 r_2[g(t) - x_i\,(t)]$$

$$x_i(t+1) = x_i(t) + v_i(t+1)$$

In engineering problems stated above, the PSO parameters taken from the given paper*.

Swarm Size	: 8 Particles
Neighborhood Size	: 3
Inertia Factor	: 0.8
Personal Learning Factor	: 1.8
Social Learning Factor	: 1.8

All the code can be found in ".m" files scripts.

*: *Solving Engineering Optimization Problems with the Simple Constrained Particle Swarm Optimizer, Leticia C. Cagnina, Susana C. Esquivel, LIDIC, Universidad Nacional de San Luis, Argentina*