Particle Swarm Optimization

- meta heuristic algorithm
- contains a population of candidate solutions
- particle i position -> xi(t) where xi(t) is a vector in the set of X
- particle i velocity -> vi(t)
- particle i memory -> pi(t) where pi(t) is the best solution for particle i
- g(t) is the common swarm experience, no i

Particle Update Functions

- vi(t+1) = a*vi(t) + b*(pi(t) -xi(t)) + c*(g(t) - xi(t))
- xi(t+1) = xi(t) + vi(t+1)
- vij(t+1) = inertia + cognitive + social where vij is the jth scalar
- xij(t+1) = xij(t) + vij(t+1)

```
% EXAMPLE
%
% inertia = coef*velocity(i,j);
% cognitive = rand()*accel1*(particleBest(i,j) - particlePos(i,j));
% social = rand()*accel2*(globalBest(j) - particlePos(i,j));
```

The Problem

- The sphere function
- http://benchmarkfcns.xyz/benchmarkfcns/spherefcn.html

The Algorithm

- problem definition

```
% define the cost function
costFcn = @(x) sphere(x);

% number of unknown variables
numParams = 5;

% matrix size of solutions
size = [1 vars];

% variable range
varMin = -20;
varMax = 20;
```

- parameters

```
% number of iterations
iterations = 100;
```

```
% number of particles
swarmSize = 50;

% inertia coefficient
w = 1;

% personal acceleration coefficient
accel1 = 2;

% social acceleration coefficient
accel2 = 2;
```

- initialization

```
% initialize the particle fields
particle.position  = [];
particle.velocity  = [];
particle.cost      = [];
particle.best.pos  = [];
particle.best.cost = [];

% initialize swarm best
swarmBest.cost = inf;

% initialize best cost vector
bestCosts = zeros(iterations, 1);

% initialize an empty population
particles = repmat(particle, swarmSize, 1);

% initialize the particle fields
for i=1:swarmSize
    % random position
    particles(i).position = unifrnd(varMin, varMax, 1, numParams);

    % zero velocity
    particles(i).velocity = zeros(numParams);

    % evaluate the particles position
    particles(i).cost = costFcn(particles(i).position);

    % particle best
    particles(i).best.position = particles(i).position;
    particles(i).best.cost = particles(i).cost;

    % swarm best
    if particles(i).best.cost < swarmBest.cost
        swarmBest.cost = particles(i).best.cost;
    end

end
```

Functions

sphere:

- the function to optimize

```
function z = sphere(x)
    z = sum(x.^2);
end
```