PSO (Particle Swarm Optimization)

for solving

$$\mathbf{x}^* = \underset{\mathbf{x}\in\Omega_{\mathbf{x}}}{\operatorname{argmin}}\left(C\left(\mathbf{x}\right)\right)$$

$$\mathbf{x}^* = \underset{\mathbf{x}\in\Omega_{\mathbf{x}}}{\operatorname{argmin}}\left(C\left(\mathbf{x}\right)\right)$$

Optimization

Certain class of optimization problems require more powerful approaches, better suited to deal with complicated cases

(nonconvex cost functions, discontinuous cost functions , etc.)

The PSO approach is known to offer powerful capabilities.

$$\mathbf{x}^* = \underset{\mathbf{x}\in\Omega_{\mathbf{x}}}{\operatorname{argmin}}\left(C\left(\mathbf{x}\right)\right)$$

PSO

A typical **swarm intelligence system** usually presents the following properties:

* It is composed by many individuals.

* The individuals are relatively homogeneous.

* The interactions between the individuals are based on simple behavioral rules exploiting local information which the individuals directly exchange, or which is perceived from the environment.

* The global behavior of the system results from the local interactions of the individuals with each other and with their surrounding context.

swarm intelligence system

Here , we want those individuals to optimize (i.e. ,minimize or maximize) a function.

(Minimize a cost function, Maximize a reward function)

e.g., minimizing a cost function

$$\mathbf{x}^* = \underset{\mathbf{x}\in\Omega_{\mathbf{x}}}{\operatorname{argmin}}\left(C\left(\mathbf{x}\right)\right)$$

$$\mathbf{x}^* = \underset{\mathbf{x}\in\Omega_{\mathbf{x}}}{\operatorname{argmin}}\left(C\left(\mathbf{x}\right)\right)$$

swarm intelligence system, for optimization

→ PSO

In PSO, we name those individuals as "particles"

Particles "move" and are updated, seeking for optimal points.

We will focus on the problem of minimizing a cost function (maximizing a reward function is an equivalent problem).

$$\mathbf{x}^* = \underset{\mathbf{x}\in\Omega_{\mathbf{x}}}{\operatorname{argmin}}\left(C\left(\mathbf{x}\right)\right)$$

The PSO process will iterate, updating its particles , searching for the cost function's minimum.

The PSO process applies certain rules on the movement of the particles

Particles are samples of X (the argument of the cost function), so they live in the domain of the function being minimized

We have/use a population of particles.

Population : M particles $\left\{\mathbf{p}_i\right\}_{i=1}^{M}$

properties of particles do change at each iteration of optimization process

$\left\{\mathbf{p}_{i,\mathrm{k}}\right\}_{i=1}^{M}$

k : iteration count

Particles are defined by three vector properties

properties of particle #i at iteration n

$$\mathbf{p}_{i,n} = \left(\mathbf{X}_{i,n}, \mathbf{V}_{i,n}, \mathbf{P}_{i,n}\right)$$

* Particle's current position vector:

$$\mathbf{X}_{i,n} = (X_{i,n}^{1}, \cdots, X_{i,n}^{j}, \cdots, X_{i,n}^{N}) \in \mathbb{R}^{N}$$
$$1 \leq j \leq N$$

* Particle's velocity vector:

$$\mathbf{V}_{i,n} = (V_{i,n}^{1}, \cdots, V_{i,n}^{j}, \cdots, V_{i,n}^{N})$$
$$1 \leq j \leq N$$

* Particle's personal best $\left(\text{experienced}\right)$ position vector:

$$\mathbf{P}_{i,n} = (P_{i,n}^{1}, \cdots, P_{i,n}^{,} \cdots, P_{i,n}^{N})$$
$$1 \leq j \leq N$$

Common/shared property: The global best position

$$\mathbf{G}_n = \mathbf{P}_{g,n}$$

$$\left(g = \text{argmax}_{1 \le i \le M} \{ f(\mathbf{P}_i, n) \}\right)$$

"It is the best position we remember as a group, since iteration 0" (the particles say)

Particles evolve according to their individual velocity vectors.

The PSO process does control those velocity vectors , based on rules and on what the individual particles “experience”

* Updating individual velocity vector

.

$$\mathbf{V}_{i,n+1} = \mathbf{V}_{i,n} + c_1 \cdot r_{i,n} \cdot \left(\mathbf{P}_{i,n} - \mathbf{X}_{i,n} \right) + c_2 \cdot R_{i,n} \cdot \left(\mathbf{G}_n - \mathbf{X}_{i,n} \right),$$

(velocity of particle #i at iteration $n+1$, $\mathbf{V}_{i,n+1}$)

$$r_{i,n} \sim U(0,1), \quad R_{i,n} \sim U(0,1)$$ (random coefficients)

c_1, c_2 : constants ("Acceleration constants")

* Updating individual position vector

$$\mathbf{X}_{i,n+1} = \mathbf{X}_{i,n} + \mathbf{V}_{i,n+1},$$

$$(1 \le j \le N)$$

PSO

$$\mathbf{V}_{i,n+1} = \mathbf{V}_{i,n} + c_1 \cdot r_{i,n} \cdot \left(\mathbf{P}_{i,n} - \mathbf{X}_{i,n}\right) + c_2 \cdot R_{i,n}^{j} \cdot \left(\mathbf{G}_{\mathbf{n}} - \mathbf{X}_{i,n}\right),$$

$$r_{i,n} \sim U(0,1)$$

$$R_{i,n} \sim U(0,1)$$

$\mathbf{V}_{i,n}$: previous velocity , known as the "inertia" part

$c_1 \cdot r_{i,n} \cdot \left(\mathbf{P}_{i,n} - \mathbf{X}_{i,n}\right)$: The "cognition" part

$c_2 \cdot R_{i,n}^{j} \cdot \left(\mathbf{G}_{n} - \mathbf{X}_{i,n}\right)$: The "social" part

Where the scalar factors c_1 and c_2 are used to define the influence of the individual's knowledge and that of the group's knowledge, respectively.

This way of updating the velocity vectors implements what is called "the original **PSO"**

$$\mathbf{V}_{i,n+1} = \mathbf{V}_{i,n} + c_1 \cdot r_{i,n} \cdot \left(\mathbf{P}_{i,n} - \mathbf{X}_{i,n} \right) + c_2 \cdot R_{i,n}^{j} \cdot \left(\mathbf{G_n} - \mathbf{X}_{i,n} \right),$$

$$r_{i,n} \sim U\left(0,1\right)$$

$$R_{i,n} \sim U\left(0,1\right)$$

The values of the components of the velocity vector are limited (in an interval of valid values)

$$-\mathbf{v}_{Max,d} \leq \mathbf{v}_{i,d} \leq \mathbf{v}_{Max,d}$$

(then, after updating the positions of all the particles)

Updating particle's personal best position

$$\mathbf{P}_{i,n} = \begin{cases} \mathbf{X}_{i,n}, & f(\mathbf{X}_{i,n}) < f(\mathbf{P}_{i,n-1}) \\ \mathbf{P}_{i,n-1}, & f(\mathbf{X}_{i,n}) \geq f(\mathbf{P}_{i,n-1}), \end{cases}$$

.

Updating the global best position

$$\mathbf{G}_n = \mathbf{P}_{g,n}$$

$$g = \text{argmax}_{1 \leq j \leq M} \{ f(\mathbf{P}_{j,n}) \}.$$

“acceleration” coefficients c1 and c2 are tuning parameters for our PSO optimizer

Usual values are The values for c1 and c2, is c1 = 2, c2=2.

We may change their values, during the optimization process, to incentive

exploration
or
refinement of solution.

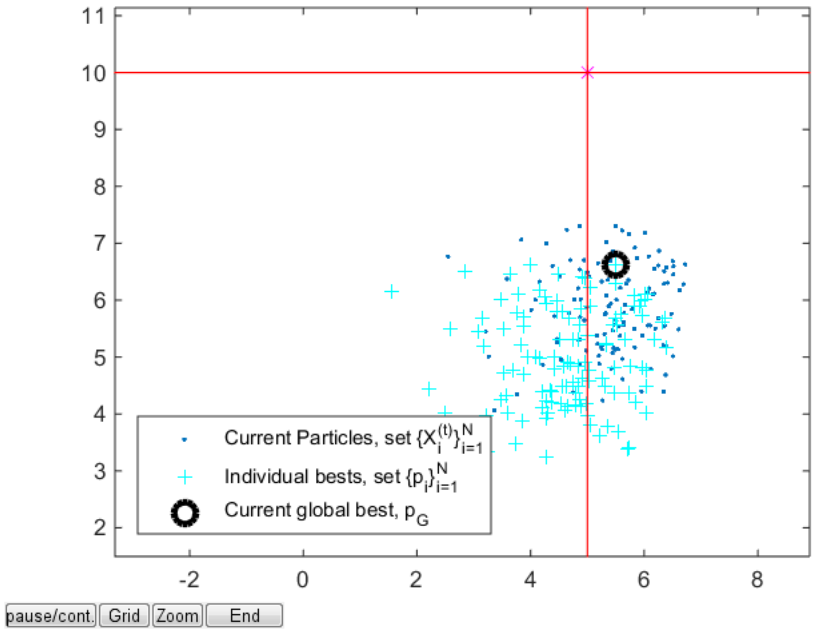
PSO simplified Pseudo Code;

```
P=particle initialization();
for i =1 to L                                          //max num iterations
{           for each particle p_k in P                 //P: population of M particles)
            {           do cost= F( p_k.X )
                        If cost < p_k.BestCost
                        {           p_k.bestX =p_k.X; p_k.BestCost=cost ; }
                        }
            gbest.X  :  choose based on  best p_k.BestCost  fpr all p_k  in P.

            if(gBest.X)<tolerance)                 // gbest is good enough, done!
            {           solution = gbest.X;
                        END NOW
            }
for each particle p_k in P do
{           update p_k.V                    //following PSO equation
            update p_k.X
}
}
//end of iteration loop (L iterations) //max num iterations reached,
solution = gbest.X ;
```

We see some low dimensional case working.
(low dim, so we can see its particles)

We see it in slow motion.



Swarm size

The size of the swarm, M, means the number of particles. Each particle has a unique index, between 1 and M.
A proper value for M is usually estimated by using the following formula:
$M=10+\lceil 2\cdot sqrt(D)\rceil$,
in which D is the dimension of the state space. However, in many applications, the heuristically recommended number is M=40. In fact, there is not a theoretical analysis about how to tune this parameter.

(Remember we evaluateour cost function for each particle at for each iteration!)

PSO is an area of research.
New variants are proposed for faster convergence and better performance

Certain variants are intended for certain class of problems.

PSO Variant

The concept of an **inertia weight**, w, was introduced.
The question was about preferring a swarm inclined to explore far away, or to explore in more detail, locally.

$$\mathbf{V}_{i,n+1} = w \cdot \mathbf{V}_{i,n} + c_1 \cdot r_{i,n} \cdot \left(\mathbf{P}_{i,n} - \mathbf{X}_{i,n} \right) + c_2 \cdot R_{i,n}^{j} \cdot \left(\mathbf{G}_{\mathbf{n}} - \mathbf{X}_{i,n} \right),$$

$$r_{i,n} \sim U(0,1)$$

$$R_{i,n} \sim U(0,1)$$

.. **inertia weight**, w
(variant named "**PSO-In**")

$$\mathbf{V}_{i,n+1} = w \cdot \mathbf{V}_{i,n} + c_1 \cdot r_{i,n} \cdot \left(\mathbf{P}_{i,n} - \mathbf{X}_{i,n} \right) + c_2 \cdot R_{i,n}^{j} \cdot \left(\mathbf{G}_{\mathbf{n}} - \mathbf{X}_{i,n} \right),$$

Typical values for W are in the range [0.8 to 1.2]

Low values: refine/search nearby
High values: incentive to go far away.

Comment : W>1 ➔ " V(n+1)=w*V(n)" →unstable (you may know that from MTRN3020)
No worries: cognitive and/or social terms will act as a REGULATOR (it is closed loop!, like a stabilizing PID controller, in MMAN3200)

$$\mathbf{V}_{i,n+1} = w \cdot \mathbf{V}_{i,n} + c_1 \cdot r_{i,n} \cdot \left(\mathbf{P}_{i,n} - \mathbf{X}_{i,n} \right) + c_2 \cdot R^{j}_{i,n} \cdot \left(\mathbf{G}_{\mathbf{n}} - \mathbf{X}_{i,n} \right),$$

Typical values for W are in the range [0.8 to 1.2]

Low values: refine/search nearby
High values: incentive to go far away.

W>1 ➔ " V(n+1)=w*V(n)" →unstable (you may know that from MTRN3020)

No worries: cognitive and/or social terms will act as a REGULATOR (it is closed loop!, like a stabilizing PID controller, in MMAN3200)

→ It will converge to some finite solution (except the function does not have actual solution) (maybe, because we did not properly define the problem we want to solve!)

(However, there is not guarantee it will be the global solution.)

Adaptive versions and time variant W

$$\mathbf{V}_{i,n+1} = w_n \cdot \mathbf{V}_{i,n} + c_1 \cdot r_{i,n} \cdot \left(\mathbf{P}_{i,n} - \mathbf{X}_{i,n}\right) + c_2 \cdot R^j_{i,n} \cdot \left(\mathbf{G}_{\mathbf{n}} - \mathbf{X}_{i,n}\right),$$

$$w_n = \frac{(w_{initial} - w_{final}) \cdot (n_{max} - n)}{n_{max}} + w_{final},$$

W factor linearly decreases during optimization process.
So, the optimization process incentives exploration at early stages, and try to refine solution during the last iterations.

Other variants (read lecture notes, for a brief description of those variants)

PSO lecture ends here.

We briefly described the PSO approach

You may try it in the same way you use other optimizers.

(There is a PSO toolbox in Matlab)

Next step?

Apply those concepts/approaches in our case of study (Project 2)

Part B:Use EKF for estimating parameters in real-time

Part C : Tuning model parameters via optimization (off-line fashion)
Here you use the Matlab optimizer you prefer
(fminearch, PSO, etc)