Student: Ahmet Burak Koç
ID: N20152984

CMP 756 Swarm Systems Homework 3

Problem:
This is the six hump camelback function where x lise in [-3,3] and y lies in [-2,2]. The objective is to minimize z. The global minimum lies at (-0.0898, 0.7126) where z=-1.0316.

$$z = \left(4 - 2.1x^2 + \frac{x^4}{3}\right)x^2 + xy + (-4 + 4y^2)y^2$$

Solution: To solve this problem we used Particle Swarm Optimization. "HW3.m" Matlab script is written on VSCode and MATLAB 2019a. It is ready to run on MATLAB 2019a.
"HW3.m" code requires below files.

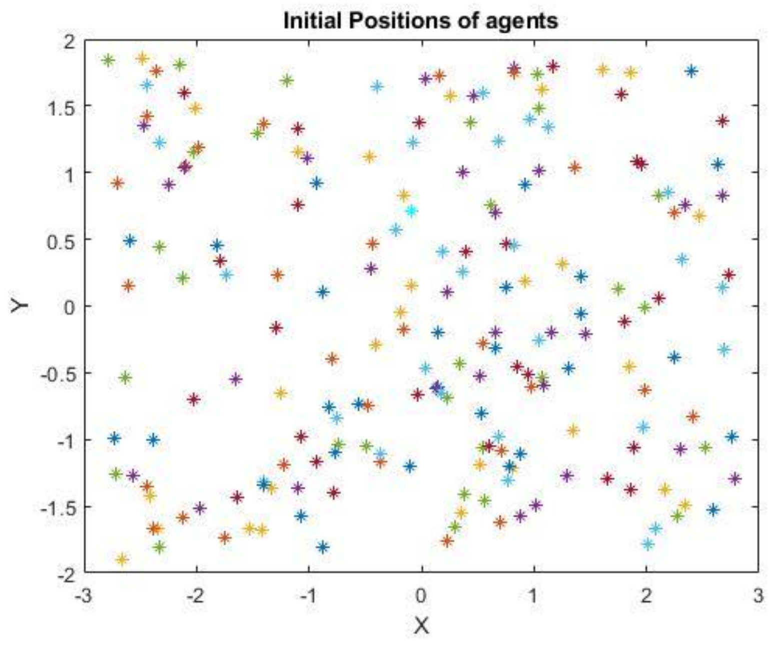
- initAgents.m
- initBest.m
- initStopping.m
- PSO.m
- SixHumpCamelback.m

Note that, attached workspace includes all necessary files so user can easily run the "HW3.m" script on MATLAB.

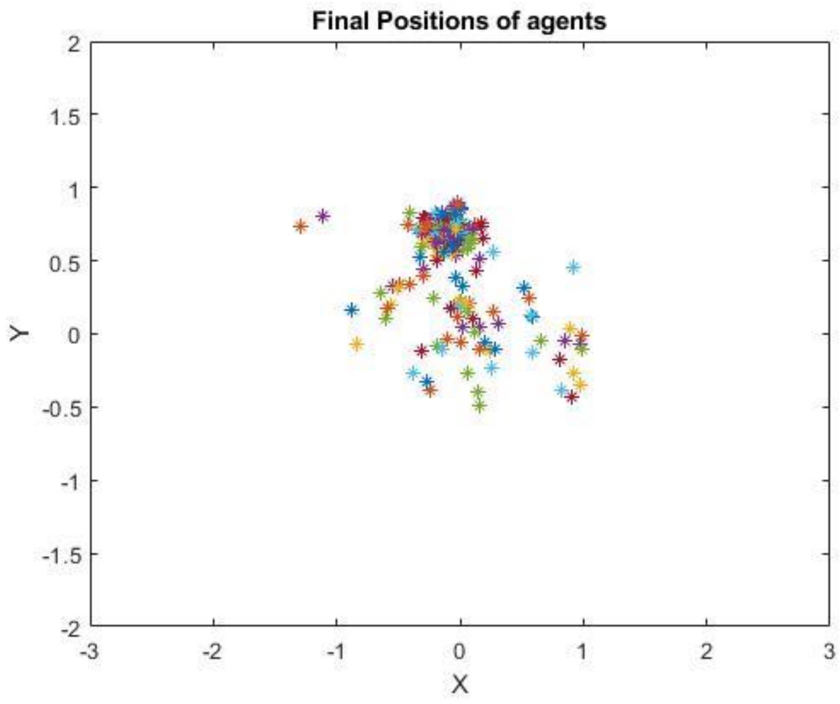
1) Description of Script "HW3.m"

I tried to implement the modular solution on Matlab script so that it would be easier to understand by user. The script is composed of sevral sections. Sections and functions of sections are given below.

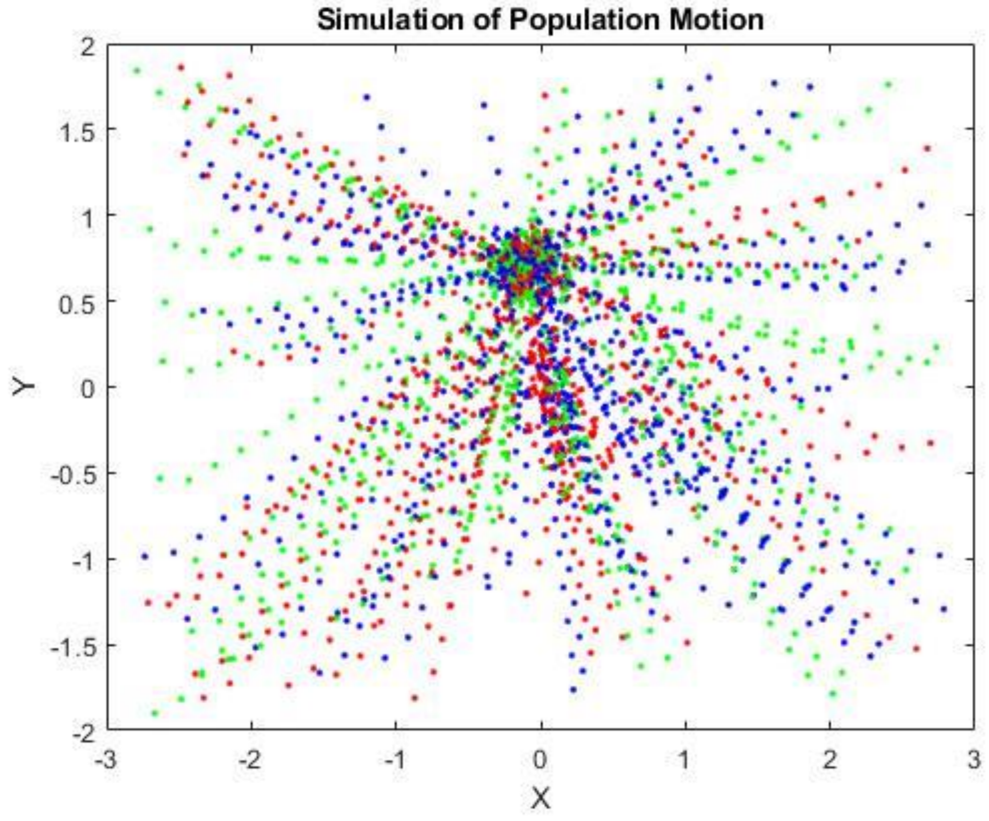
- Informative Sections
  - `%% Description`: Information about university, course and student is given in this section.
  - `%% How to Run Code`: Code is ready to run with default parameters but sequence to run the code is given in this section.
- Configurable Sections: Some parameters are fixed to solve problem. Some other parameters can be configured by user. The current code is ready to run with default parameters. If user want to configure some parameters, s/he can adjust below parameters,
  - `%% PSO Parameters`
    - c1
    - c2
    - vMax: speed limit
  - `%% Stopping Conditions`
    - maxIter: maximum number of iterations
    - minError: error limit iterations stops when reached
  - `%% Agents Initialization Parameters`
    - numAgents: number of agents
- Running Algorithm Sections:
  - `%% Run algorithm:` This section uses other Matlab scripts and functions to initialize agents population and run the PSO algorithm to find minimum of objective function.
- *Displaying Results Sections:* Following sections are used to demonstrate results.
  - `%% Rearrange history of positions, velocities and error:` This section is used to rearrange results for following plots.
  - `%% Plot Initial Positions`



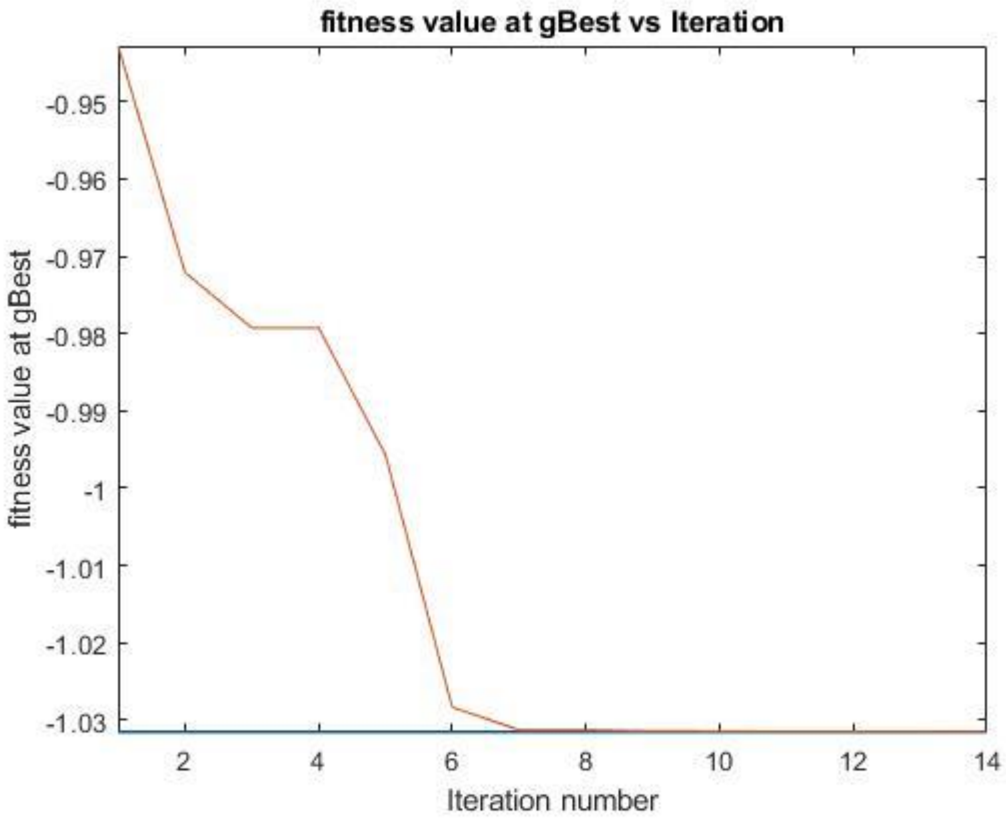
- `%% Plot Final Positions`



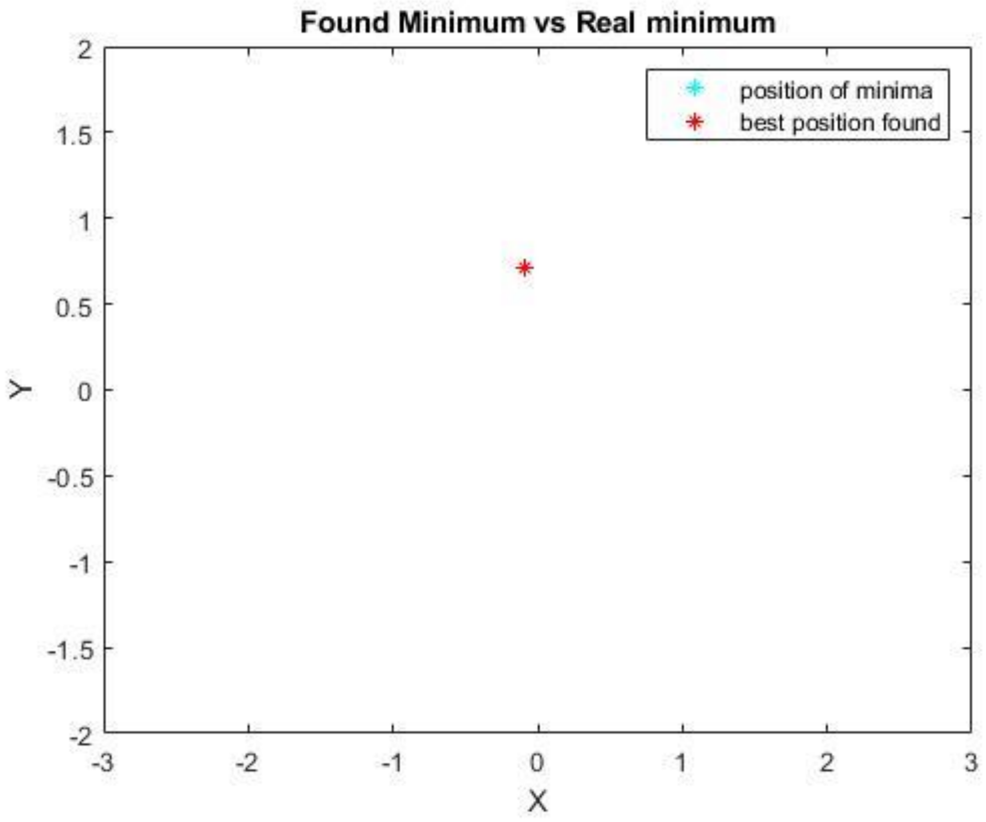
- `%% Plot with time`



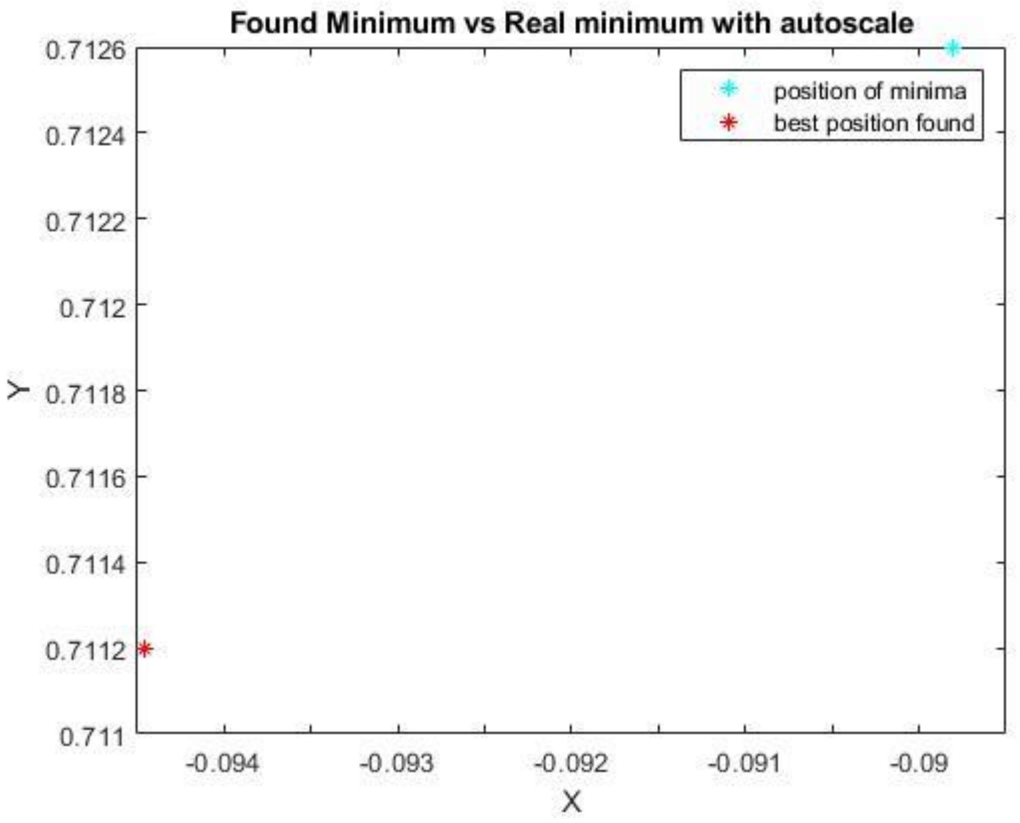
- `%% Plot gBest vs iteration`



- `%% Found Minimum vs Real minimum without autoscale`



- `%% Found Minimum vs Real minimum with autoscale`



2) Description of Script “***initAgents.m***”

Create a ‘population’ of agents (particles) uniformly distributed over boundary x in [-3 3] and y in [-2 2].

3) Description of Script “***initBest.m***”

Initialize pBest and gBest.

4) Description of Script “***initStopping.m***”

Initialize stopping conditions.

5) Description of Script “***PSO.m***”

Main PSO Algorithm

- Line 2-7: Define some arrays to remember informations(positions, velocities etc.) of each agents in each iteration.
- Line 8- 62: Main PSO loop iterate until stopping criterion fulfills and “stopFlag” gets TRUE.
- Line 11-17:
    - For each agent
        - Calculate fitness value,
        - If the fitness value is better (**lower fitness value is better**) than its peronal best set current value as the new pBest
- Line 19-22:
    - Choose the particle with the best fitness value of all as gBest
- Line 24-34:
    - For each particle,
        - Calculate particle velocity according equation (a)
        - Update particle position according equation (b)
- Line 37-42:
    - Update variables for Stopping Criterion
- Line 44-54:
    - Check Stopping Criterion
- Line 56-61:
    - Store informations for current iteration before going next iteration.