# Group Assignment 2: Particle Swarm Optimisation

Core Body of Knowledge classification: Abstraction, Design, Teamwork concepts & issues, Data, Programming, Systems development

Due date: 11:59pm, 8th October 2023, Weight: 10% of course

## 1 Assignment structure and practical requirements

The assignment is done in groups of 5-6 students. Each student has to take major responsibility for parts of the assignment and collaborate with the team members. The group has to make sure that the workload is evenly distributed. Report progress and task distribution to your tutor during the workshops. This is worth points!

Note that the amount of marks for the assignment is not proportionate to its importance. The amount of marks for group work is limited by the university, for example due to the risk of uneven contributions. The group assignments serve the learning process and their benefit in terms of marks will also show in the exam result. Specifically, the content of the assignment, e.g. new concepts like topologies or swarm convergence measures, will be relevant for the exam. Each group member should understand all parts of the assignment.

Use the programming language Python and store your code in the GitHub classroom repository supplied by the course. Commit updates frequently and write a concise but informative commit message. This will document that all group members have contributed regularly. Save results in the specified files and directories. Think before committing big output files, because this can make your repository very large over time. This is worth points! For example, developing the submission outside of GitHub classroom and then submitting the final result only will cost marks.

Use one file *pso_lib.py* for all of the modular library code and one or more files like *run_experiments.py* to assemble and execute the experiments and files to produce the analysis outputs. Document your code with concise but informative comments. This is worth points!

Make the figures self-explanatory and unambiguous. Always include axis labels, if available with units, unique colours and markers for each curve/type of data, a legend and a title. Give every figure a number (e.g. at start of title), so that it can be referred to from anywhere unambiguously. This is worth points!

Do not use generative AI for coding or text answers. Do not copy code or text from anywhere. We are automatically running plagiarism checks and generative AI detectors. Instead, make your submission original and unique by writing everything yourself, choosing function and variable names well and documenting your code with your own comments.

## 2 Objective

The objective is to design a library, implement particle swarm optimisation methods and investigate PSO variants. The PSO library should be able to solve real valued problems with different numbers of dimensions (parameters to be optimised) by maximising a fitness function that characterises the problem. Real-valued problems are common in science, engineering and technology. PSO is readily applicable with little need to customise the algorithm to the problem beyond choosing hyperparameters.

Read the survey paper Houssein et al. 2021 to get an overview of PSO methods development and open challenges. The paper is available on the MyUni assignment page.

> Houssein EH, Gad AG, Hussain K, Suganthan PN. 2021. Major Advances in Particle Swarm Optimization: Theory, Analysis, and Application. Swarm and Evolutionary Computation 63:100868, https://doi.org/10.1016/j.swevo.2021.100868.

## 3 Team roles and contributions (5 points)

Write a short paragraph for each group member about the role and contributions to the assignment. Write a draft at the start and update it as needed. It is normal that the roles change sometimes. It is important to plan and track the roles in a group throughout the process.

Write team contributions into the file *doc/team_contributions.txt*. The final versions should have at most 75 words per student.

## 4 Code architecture (10 points)

When designing your library for particle swarm optimisation, follow object-oriented design practices and build a modular system. It should be possible to extend the system by using different implementations of the components of the algorithm that are investigated in this assignment.

The survey paper of Houssein et al. (2021) lists and describes the components with alternative approaches comprehensively and can be used to understand the wider modular nature of particle swarm optimisation. But in the assignment code architecture, only the variations specified below need to be implemented.

Write your architecture design and rationales into the file *doc/architecture_design.txt*. Use this file to document your design incrementally and to coordinate within the group. The final version should have at most 300 words.

Make one diagram to summarise your object-oriented architecture, for example using UML (Unified Modelling Language). Save the figure as *doc/architecture_diagram.png*. The diagram can have sub sections, but has to be easily readable when printed on A4 paper.

# 5 Implementation

## 5.1 Real valued optimisation problems (5 points)

Write a class which represents real-valued optimisation problems of different numbers of dimensions. Write a derived class that implements the Ackley function with variable dimension and boundaries from Lecture 4, which is a benchmark problem for optimisers like PSO.

## 5.2 Optimisers (10 points)

Write an optimiser class, which implements standard PSO as defined in this paper (open access, also available on MyUni assignment page):

> Bratton D, Kennedy J (2007) Defining a Standard for Particle Swarm Optimization, Proceedings of 2007 IEEE Swarm Intelligence Symposium, Honolulu, HI, USA, IEEE, https://doi.org/10.1109/SIS.2007.368035

This version of PSO serves as a reference for comparing PSO variants.

## 5.3 Initialisation of swarm, velocities and topology (10 points)

Add code to initialise a swarm randomly with particle positions and velocities. Add code to set up neighbourhood topologies, such as a fully connected swarm (gbest), ring topology (lbest), star topology and random neighbourhood connectivity.

## 5.4 Metrics figure (10 points)

Write code to produce a figure that shows several metrics to describe the swarm and optimisation progress. You will reuse this figure in the next sections.

The figure should have 4 sub-figures:

- The first sub-figure shows current best fitness value (y-axis) over the iterations on the x-axis
- The second subfigure shows the distance of the swarm centre of mass (the mean of all positions in the swarm) to the global optimum, which is at the origin or the n-dimensional search space on the y axis and the iterations on the x-axis. It shows whether the swarm is converging towards the global optimum.
- The third sub-figure shows the standard deviation of particle positions around the centre of mass on the y-axis. This is a measure of the swarm size.
- The fourth sub-figure shows the mean length of the velocity vectors. This is a measure of the speed that the swarm is moving.

The display code should be able to plot multiple optimisation runs together as curves. Use a python package such as distinctipy to generate maximally differentiable colours for any number of curves.

## 6 Standard PSO on Ackley function (10 points)

Run Standard PSO on the Ackley function with 10 dimensions in a range of $+/-30$ in each dimension. Use 1000 iterations and an early stopping criterion. Repeat the optimisation 10x with different random initialisation. Plot the 10 runs into one metrics diagram using the figure code described earlier. Save the diagram to *results/metrics_std_pso.png.*

Analyse the metrics diagram while making specific references to observations. How variable are the runs for different initialisations? How many iterations are needed to converge approximately? Does the whole swarm converge towards the global optimum? How close do the optimisations get? Write your analysis into the file *results/analysis_std_pso.txt* using max. 300 words.

## 7 Swarm size comparison (10 points)

Repeat the same optimisation with swarm size 20, 100, and 200. Save a metrics diagram each into files *results/metrics_std_pso_swarm20.png*, *results/metrics_std_pso_swarm100.png*, *results/metrics_std_pso_swarm200.png*

Compare the metrics diagrams while making specific references to observations in specific diagrams. How variable are the runs for different swarm sizes? How many iterations are needed to converge approximately? Which swarm size gets closer to the global optimum? Which swarm size converges faster? How might those differences be caused by the swarm size? Which swarm size would you choose? Write your analysis into the file *results/analysis_swarm_size.txt* using max. 300 words.

## 8 Inertia weight adjustment (10 points)

Derive a modified PSO optimiser that uses the standard algorithm but implements the non-linear inertia weight adjustment described in Section 3.3.1 of Houssein et al. 2021 (See Reference 61). Run 10 optimisations with standard PSO settings and display the results in one metrics diagram that is saved in *results/metrics_std_pso_weight_adjust.png.*

Test and describe your choice of the modulation index. Compare the results with the std PSO experiment and the swarm size experiments while making specific references to observations in specific diagrams. Is the final result and convergence improved? Write your analysis into the file *results/analysis_weight_adjust.txt* using max. 300 words.

## 9 Neighbourhood topology comparison (10 points)

Derive modified PSO optimisers that uses the standard algorithm but implement the different neighbourhood topologies: fully connected swarm (gbest), ring topology (lbest), star topology and random 50% neighbourhood as described in Figure 5 of Houssein et al. 2021. Run 10 optimisations each and display the results in one metrics diagram each that is saved in *results/metrics_std_pso_topo_gbest.png*, etc.

Compare the results with the std PSO experiment and the swarm size experiments and the weight adjustment while making specific references to observations in specific diagrams. Which connectivity performs best? Which one leads to a wider swarm? Which one converges faster? Which one produces the closest result? Which one runs fastest and why? Write your analysis into the file *results/analysis_topologies.txt* using max. 300 words.

## 10 Find best combination of swarm size, weight adjustment and topology (10 points)

You have investigated several PSO variations individually. When using two or more variations at the same time, they may interact with each other. The task is to find a combination of swarm size choice, weight adjustment choice and topology choice out of the options explored in the earlier sections that performs well in the metrics diagrams.

Test different combinations and document their performance in files *results/metrics_comb_ .. .txt*.

Analyse and compare the combinations tested while making specific references to observations in specific diagrams. Which combination would you choose and why? Is there a combination where the interactions worsen the performance? Write your analysis into the file *results/analysis_combinations.txt* using max. 300 words.

## 11 Submission

Submit your Github repository to Gradescope within MyUni once per group by the spokes person. Before you submit, copy the final versions of the following files into the directory *final/*. We will mark these final versions.

- *doc/team_contributions.txt*
- *doc/architecture_design.txt*
- *doc/architecture_diagram.png*
- *results/metrics_std_pso.png*
- *results/analysis_std_pso.txt*
- *results/metrics_std_pso_swarm20.png*,
- *results/metrics_std_pso_swarm100.png*,
- *results/metrics_std_pso_swarm200.png*
- *results/analysis_swarm_size.txt*
- *results/metrics_std_pso_weight_adjust.png*
- *results/analysis_weight_adjust.txt*
- *results/metrics_std_pso_topo_gbest.png*, etc.
- *results/analysis_topologies.txt*

- *results/metrics_comb*...*txtresults/analysis_combinations.txt*

  Make sure you have addressed every single instruction in the assignment.

  Make sure the code runs and generates the outputs that you are submitting.

  Make sure the maximum word counts, figure formatting requirements and code documentation requirements are adhered to. Violating these requirements will reduce your mark.