# 

**Assessment 2**

**Optimization of Multi-Modal Functions Using Evolution Strategies**

**Unit Coordinator : Umair Ullah Tariq**

**Class : Tutorial 1**

**Date : 16 May 2025**

**Student Number Student Name**

**12223508 Furkan Tekkartal**

**COIT29224**

**Evolutionary Computation**

**2025 Term 1**

Table of Contents

[0](#_Toc198314234)

[1 Introduction 2](#_Toc198314235)

[2 Execution Instructions 3](#_Toc198314236)

[2.1 File Structure: 3](#_Toc198314237)

[2.2 Prerequisites: 3](#_Toc198314238)

[2.3 Running the Code: 3](#_Toc198314239)

[2.4 Output: 4](#_Toc198314240)

[2.5 GitHub Repository: 4](#_Toc198314241)

[3 Problem Formulation 5](#_Toc198314242)

[3.1 Rastrigin Function 5](#_Toc198314243)

[3.2 Optimization Objective 5](#_Toc198314244)

[4 Algorithm Implementation 6](#_Toc198314245)

[4.1 Evolution Strategy: (1+1)-ES with 1/5 Success Rule 6](#_Toc198314246)

[4.2 Baseline: Gradient Descent 7](#_Toc198314247)

[4.3 Implementation Details 8](#_Toc198314248)

[5 Experimental Evaluation 9](#_Toc198314249)

[5.1 Experimental Setup 9](#_Toc198314250)

[5.2 Performance Metrics 10](#_Toc198314251)

[6 Results and Comparison 11](#_Toc198314252)

[6.1 Performance Metrics by Dimension 11](#_Toc198314253)

[6.2 Final Score Comparison 13](#_Toc198314254)

[6.3 Execution Time Analysis 13](#_Toc198314255)

[6.4 Local Minima Count Analysis 14](#_Toc198314256)

[6.5 Score Reduction Analysis 14](#_Toc198314257)

[6.6 Optimization Trajectories Analysis 15](#_Toc198314258)

[7 Discussion 18](#_Toc198314259)

[7.1 Strengths and Limitations 18](#_Toc198314260)

[7.2 Comparison with Baseline Algorithm 19](#_Toc198314261)

[8 Conclusion 20](#_Toc198314262)

[9 References 21](#_Toc198314263)

# Introduction

Optimization problems with multiple local optima present significant challenges for traditional algorithms, which often converge to the nearest local optimum rather than identifying the global optimum (Eiben & Smith 2013). Evolution Strategies (ES), a class of evolutionary algorithms, offer promising alternatives by maintaining diverse populations and employing stochastic mechanisms that can help escape local optima.

This project implements and evaluates a (1+1)-Evolution Strategy with 1/5 Success Rule to optimize the Rastrigin function, a well-known benchmark problem characterized by numerous local minima and a single global minimum. The algorithm’s performance is compared against a standard Gradient Descent implementation across different dimensionalities (1D, 2D, and 30D).

**Objectives:**

1. Implement a (1+1)-ES algorithm with the 1/5 Success Rule for step size adaptation.
2. Apply the algorithm to optimize the Rastrigin function in different dimensions
3. Compare its performance against Gradient Descent in terms of final score, execution time, and ability to escape local minima
4. Analyse the impact of problem dimensionality on optimization performance

The study provides insights into the relative strengths of evolutionary approaches versus traditional gradient-based methods in multi-modal optimization scenarios.

# Execution Instructions

The project was implemented in Python using NumPy for numerical operations, Matplotlib for visualization, and tabulate for formatted output. The code structure is modular with separate classes for the algorithms, function, and visualization.

## File Structure

* **main.py**: Main script to run the optimization algorithms
* **Code/RastriginFunction.py**: Implementation of the Rastrigin function
* **Code/OnePlusOneESFifthRule.py**: Implementation of (1+1)-ES algorithm
* **Code/GradientDescent.py**: Implementation of Gradient Descent algorithm
* **Code/Visualizer.py**: Visualization utilities
* **Output/**: Output files of program stored here.

## Prerequisites

This project developed in Python 3.13 on Windows environment. Following part will explain for a windows user. Mac or Linux user may need to do some changes on commands.

## Running the Code

A screenshot of a computer

AI-generated content may be incorrect.Run windows PowerShell on project directory.

Image1: PowerShell in project directory

* Install dependencies.

pip install numpy matplotlib tabulate

* run the program.

python main.py

## Output

* Console logs show optimization progress and results.
* Visualization plots are saved to the Output directory.
* A screenshot of a computer

  AI-generated content may be incorrect.Comparisons between algorithms are displayed in tabular format.

Image2: Runing Program and Terminal Output

## GitHub Repository

<https://github.com/furkantekkartal/COIT29224_Asg2>

# Problem Formulation

## Rastrigin Function

The Rastrigin function is a non-convex, multimodal function commonly used as a performance test for optimization algorithms. It’s characterized by a large search space with numerous local minima, making it particularly challenging for optimization algorithms to locate the global minimum.

The function is defined as:

Where:

* is a constant
* is the number of dimensions
* is the value in the -th dimension

Key properties:

* **Search space:** Typically defined within the bounds [-5.12, 5.12] for each dimension
* **Global minimum:** Located at the origin with a function value of 0
* **Local minima:** Regularly distributed throughout the search space, forming a grid-like pattern
* **Number of local minima:** Increases exponentially with the number of dimensions
* **Modality:** Highly multimodal with many peaks and valleys

The function’s difficulty arises from its numerous local minima that can trap gradient-based algorithms, while its regular structure provides a predictable benchmark for comparing different optimization approaches.

## Optimization Objective

The primary objective is to minimize the Rastrigin function:

Specifically, we aim to:

1. Find the global minimum value (0)

2. Locate the global minimum position (the origin)

3. Compare how different algorithms perform as the dimensionality increases

4. Analyse how effectively algorithms escape local minima

This optimization challenge tests an algorithm’s ability to balance exploration (searching broadly throughout the space) and exploitation (refining solutions near promising areas).

# Algorithm Implementation

## Evolution Strategy: (1+1)-ES with 1/5 Success Rule

The (1+1)-Evolution Strategy is one of the simplest forms of evolution strategies, maintaining just a single parent and generating a single offspring in each generation. Key aspects of our implementation include:

1. **Population structure:**
   * Single parent solution
   * Single offspring generated per generation
   * Offspring replaces the parent only if it has better fitness
2. **Mutation operator:**
   * Gaussian mutation applied independently to each dimension
   * Self-adaptive mutation step size (sigma) controlled by the 1/5 success rule
3. **1/5 Success Rule:**
   * The mutation strength (sigma) is adjusted based on the success rate
   * If more than 1/5 of mutations are successful, increase sigma to explore more
   * If fewer than 1/5 are successful, decrease sigma to refine the search
   * The adaptation process uses an adaptation constant of 0.5
4. **Restart mechanism:**
   * When the algorithm detects it’s stuck (very small sigma and low success rate)
   * Jumps to a new random location while preserving the best solution found so far
5. **Termination criteria:**
   * Maximum number of generations reached (10,000)
   * Error tolerance reached (solution within 0.001 of global minimum)

The (1+1)-ES algorithm is particularly suitable for the Rastrigin function because:

* It requires minimal parameter tuning
* The 1/5 success rule provides automatic adaptation to the local landscape
* The restart mechanism helps escape local optima
* The independent mutation of dimensions scales well to higher dimensions

## Baseline: Gradient Descent

As a baseline for comparison, a standard Gradient Descent algorithm was implemented with the following features:

1. **Gradient calculation:**
   * Numerical approximation using finite differences
   * Central difference method for better accuracy
2. **Update rule:**
   * Position updated based on negative gradient direction
   * Step size controlled by the learning rate parameter
3. **Learning rate adaptation:**
   * Periodic reduction of learning rate (decay factor: 0.95)
   * Accelerated reduction when stagnation is detected
4. **Stagnation handling:**
   * Counter for generations without significant improvement
   * Learning rate reduction when stagnated
   * Random restart from the best position with perturbation when heavily stagnated
5. **Boundary handling:**
   * Positions clipped to stay within function bounds

Gradient Descent serves as an appropriate baseline because:

* It’s a widely used and understood optimization approach
* It typically struggles with multimodal functions, highlighting the challenges
* It provides a clear contrast to the evolutionary approach

## Implementation Details

The implementation uses the following libraries, each selected for specific purposes:

1. **NumPy:**
   * Efficient numerical operations on arrays
   * Vectorized calculations for better performance
   * Essential for handling higher-dimensional problems efficiently
2. **Matplotlib:**
   * Visualization of optimization trajectories and function landscapes
   * Performance metrics visualization
   * Multiple plot types (1D, 2D contour, 3D surface, high-dimensional radar charts)
3. **Random/Seed:**
   * Controlled randomization for reproducible experiments
   * Consistent random starting points for fair comparisons
4. **Tabulate:**
   * Formatted tables for clear presentation of comparative results
   * Enhanced readability of performance metrics
5. **Time:**
   * Accurate measurement of execution time for performance comparison

The architecture follows object-oriented design principles with clear separation of concerns:

* **Function class:** Encapsulates the Rastrigin function and its properties
* **Algorithm classes:** Implement optimization logic independently
* **Visualizer class:** Handles all visualization tasks
* **Main script:** Orchestrates experiments and comparisons

Code modularization enables easy extension to other functions or algorithms while maintaining a consistent interface.

# Experimental Evaluation

## Experimental Setup

The experiments were designed to compare the performance of (1+1)-ES and Gradient Descent across multiple dimensions:

1. **Dimensionality:**
   * 1D: Simplest case, allows for clear visualization of the optimization trajectory
   * 2D: Standard benchmark case, enables contour and 3D visualization
   * 30D: High-dimensional case, tests scalability and handling of curse of dimensionality
2. **Algorithm parameters:**
   * **Evolution Strategy:**
     + Initial sigma: 0.5
     + Adaptation constant: 0.5
     + Max generations: 10,000
   * **Gradient Descent:**
     + Initial learning rate: 0.01
     + Decay factor: 0.95
     + Max generations: 10,000
3. **Random seed:**
   * Fixed seed (42) for reproducibility
   * Ensure both algorithms start from the same positions
4. **Experiments conducted:**
   * Each algorithm was run on each dimension (1D, 2D, 30D)
   * Results were collected and compared

## Performance Metrics

Several metrics were used to evaluate and compare algorithm performance:

1. **Final score:**
   * The function value achieved at the end of optimization
   * Lower values indicate better optimization (global minimum is 0)
2. **Execution time:**
   * Time taken to complete the optimization process
   * Measures computational efficiency
3. **Error from global minimum:**
   * Absolute difference between achieved score and known global minimum (0)
   * Indicates optimization accuracy
4. **Local minima count:**
   * Theoretical number of local minima in the search space
   * Illustrates the challenge of increasing dimensionality
5. **Score reduction percentage:**
   * Relative improvement from initial to final score
   * Normalized measure of optimization effectiveness
6. **Mean squared error:**
   * Distance between found solution and known global minimum position
   * Spatial accuracy of the optimization

These metrics provide a comprehensive view of algorithm performance across different aspects of the optimization problem.

# Results and Comparison

## Performance Metrics by Dimension

A group of graphs showing different colored bars

AI-generated content may be incorrect.The performance metrics for both algorithms across different dimensions are summarized in the visualization plots below:

Image 3: Performance metrics for the (1+1)-ES algorithm across different dimensions

A group of different colored bars

AI-generated content may be incorrect.

Image 4: Performance metrics for the Gradient Descent algorithm across different dimensions

The plots reveal several important patterns:

1. **Final score by dimension:**
   * ES achieves perfect optimization in 1D (score 0.00), near optimal in 2D (0.99)
   * GD achieves a score of 0.99 in 1D and 25.87 in 2D
   * In 30D, ES significantly outperforms GD (226.85 vs 268.64)
   * The dramatic increase in final scores at 30D demonstrates the curse of dimensionality
2. **Execution time by dimension:**
   * Execution time increases with dimensionality for both algorithms
   * ES is consistently faster than GD, especially in higher dimensions (0.17s vs 2.86s for 30D)
   * This efficiency advantage becomes more pronounced as dimensions increase (nearly 17x faster in 30D)
3. **Local minima count:**
   * The number of local minima increases exponentially with dimension
   * In 30D, there are approximately 2.37 × 10^31 local minima, illustrating the immense challenge
   * This explains why both algorithms struggled to find the global minimum in higher dimensions
4. **Score reduction percentage:**
   * ES shows excellent score reduction across dimensions   
     (99.99% in 1D, 98.11% in 2D, 57.01% in 30D)
   * GD shows less consistent performance   
     (92.80% in 1D, 50.82% in 2D, 49.10% in 30D)
   * ES demonstrates more robust performance across dimensions, particularly in 30D

A screenshot of a computer

AI-generated content may be incorrect.  
 From the console output, we can observe the optimization progress.

This shows the ES algorithm successfully utilizing its restart mechanism and achieving the global minimum in only 1033 generations in the 1D case.

Image 5: Console Output

## Final Score Comparison

Comparing the final scores achieved by both algorithms:

| Dimension | Evolution Strategy | Gradient Descent |
| --- | --- | --- |
| 1D | 0.00 | 0.99 |
| 2D | 0.99 | 25.87 |
| 30D | 226.85 | 268.64 |

Table 1: Final Scores Comparison Table

Key observations:

* ES achieves the global minimum (0.00) in 1D with an MSE of just 0.000005, while GD achieves 0.99 with an MSE of 0.989943
* ES significantly outperforms GD in 2D (0.99 vs 25.87)
* In 30D, ES outperforms GD by approximately 15.6%
* The advantage of ES becomes most evident in the 2D case, where it achieves a score 26 times better than GD

## Execution Time Analysis

Execution time comparison reveals efficiency differences:

| Dimension | Evolution Strategy | Gradient Descent |
| --- | --- | --- |
| 1D | 0.00s | 0.05s |
| 2D | 0.05s | 0.14s |
| 30D | 0.17s | 2.86s |

Table 2: Execution Time Comparison Table

Key observations:

* ES is consistently faster than GD across all dimensions
* The execution time difference becomes significantly more pronounced in higher dimensions
* In 30D, ES is approximately 16.8 times faster than GD
* This demonstrates the computational efficiency of the ES approach, particularly for complex problems

## Local Minima Count Analysis

The number of local minima increases exponentially with dimension:

| Dimension | Local Minima Count |
| --- | --- |
| 1D | 12 |
| 2D | 144 |
| 30D | ~2.37 × 10^31 |

Table 3: Local Minima Count Table

Key observations:

* The exponential increase illustrates why high-dimensional optimization is challenging
* In 30D, the vast number of local minima makes exhaustive search impossible
* This highlights the importance of algorithms that can effectively navigate multimodal landscapes

## Score Reduction Analysis

Score reduction percentage shows relative improvement:

| Dimension | Evolution Strategy | Gradient Descent |
| --- | --- | --- |
| 1D | 99.99% | 92.80% |
| 2D | 98.11% | 50.82% |
| 30D | 57.01% | 49.10% |

Table 4: Score Reduction Comparison Table

Key observations:

* ES achieves near-perfect reduction (99.99%) in 1D, completely finding the global minimum
* ES maintains high performance in 2D with 98.11% reduction compared to GD’s 50.82%
* Both algorithms face challenges in 30D, but ES still outperforms GD (57.01% vs 49.10%)
* The performance gap is most significant in 2D, where ES reduces the score nearly twice as effectively as GD

## Optimization Trajectories Analysis

The visualization of optimization trajectories provides additional insights:

1. **1D Trajectories:**
   * ES successfully finds the global minimum at x=0   
     (green end point and yellow star are at the same position in Image 6)
   * GD gets trapped in a local minimum at approximately x=-1.4 (Image 7)
   * This clearly demonstrates ES’s ability to escape local optima through its restart mechanism

A graph with blue lines and red dots

AI-generated content may be incorrect.

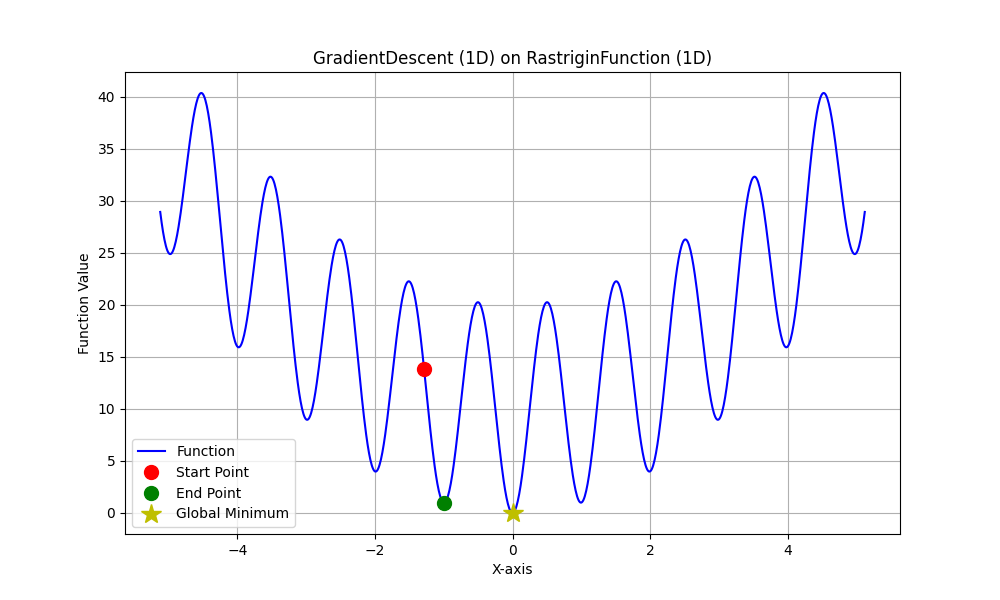


Image 7: Gradient Descent trajectory on 1D

Image 6: Evolution Strategy trajectory on 1D

1. **2D Trajectories:**
   * ES finds a position very close to the origin (Image 8), as shown by the nearby green end point
   * GD ends up in a more distant local minimum (Image 9)
   * *A screenshot of a computer generated graph

     AI-generated content may be incorrect.*The contour plots clearly show the multimodal landscape with its regular pattern of local minima (blue areas)

Image 8: Evolution Strategy trajectory on 2D Rastrigin function

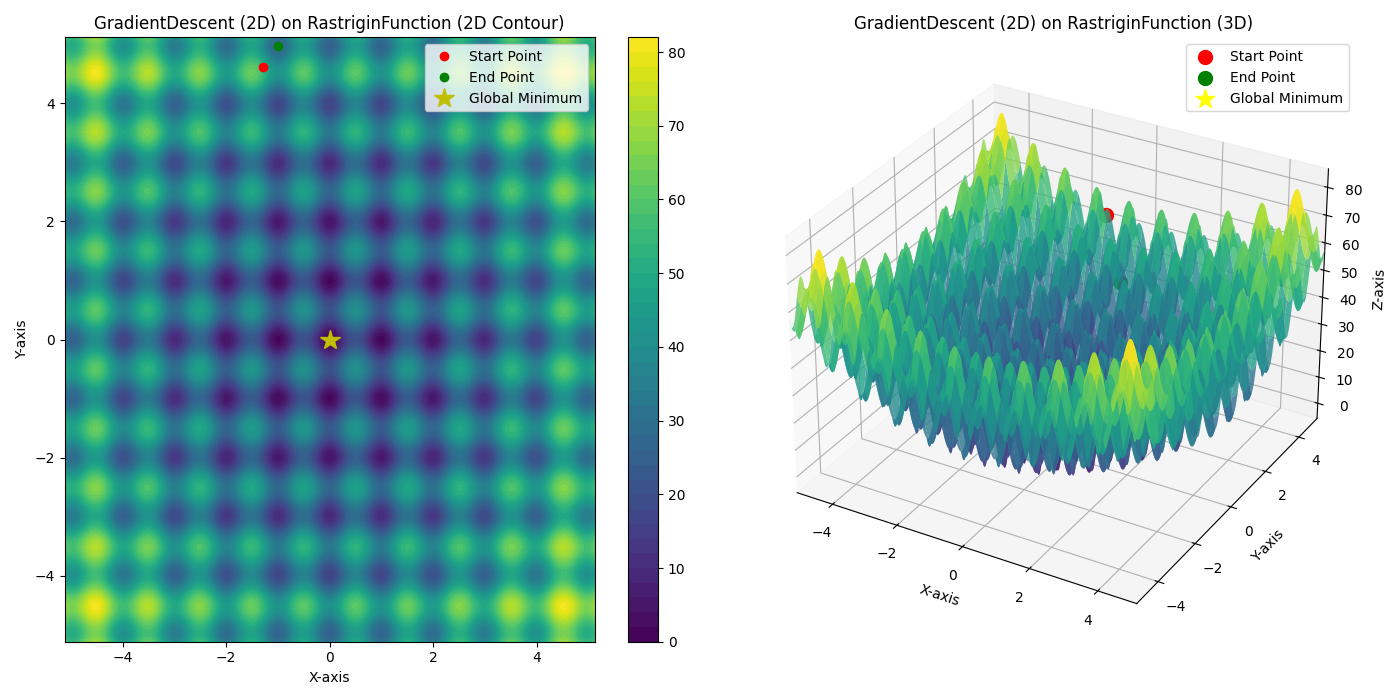
**

Image 9: Gradient Descent trajectory on 2D Rastrigin function

1. **30D Visualizations (Radar Charts):**
   * In the radar charts,
     + The center represents -5.12,
     + The yellow circle represents the negative value area,
     + The boundary of the yellow circle is 0 (global minimum for each dimension),
     + The outer edge represents 5.12
   * The black connecting lines in both plots show movement from start to end points for each dimension
   * ES solutions (Image 10)
     + Show more parameter values close to or cross into the yellow area (negative values), with several dimensions converging close to the optimal 0 value.
     + The distance Start and End points indicates algorithm discover more are.
   * GD solutions (Image 11)
     + Have more parameters stuck at higher positive values
     + The distance Start and End points indicates algorithm stuck in local minimums.
   * ES shows more effective movement toward optimal values across multiple dimensions

*A circular chart with red and green dots

AI-generated content may be incorrect.A circular chart with red and green dots

AI-generated content may be incorrect.*

Image 11: Gradient Descent parameter values

on 30D Rastrigin function

Image 10: Evolution Strategy parameter values

on 30D Rastrigin function

# Discussion

## Strengths and Limitations

**Evolution Strategy Strengths:**

* **Escape from local minima:** The stochastic nature and restart mechanism enable ES to escape local optima, as demonstrated in the 1D case where it found the global minimum.
* **Adaptive step size:** The 1/5 success rule automatically adjusts the mutation strength based on the local landscape, balancing exploration and exploitation.
* **Computational efficiency:** ES demonstrated significantly faster execution times across all dimensions, with particularly impressive advantages in higher dimensions.
* **Consistent performance:** The algorithm maintained high score reduction percentages across different dimensions compared to GD.
* **No gradient information required:** ES works effectively without requiring gradient information, making it suitable for non-differentiable or noisy functions.

**Evolution Strategy Limitations:**

* **Parameter sensitivity:** The initial sigma value and adaptation constants can influence performance.
* **Stochastic nature:** Results may vary between runs due to the probabilistic components.
* **Diminishing returns in high dimensions:** While better than GD, ES still struggles to find the global optimum in 30D.

**Gradient Descent Strengths:**

* **Simplicity:** The algorithm is straightforward to implement and understand.
* **Effective for smooth, unimodal functions:** In simpler landscapes, GD can efficiently converge to optima.
* **Deterministic:** Given the same starting point and parameters, GD will always produce the same result.

**Gradient Descent Limitations:**

* **Local optima traps:** As shown in the 1D case, GD easily gets trapped in local optima.
* **Requires gradient information:** The need to calculate gradients limits its application to differentiable functions.
* **Computational cost:** GD showed significantly longer execution times in higher dimensions.
* **Inconsistent performance:** Score reduction varied widely across dimensions.

## Comparison with Baseline Algorithm

The comparison between (1+1)-ES and Gradient Descent revealed several important insights:

1. **Performance in multimodal landscapes:**
   * ES consistently outperformed GD in the Rastrigin function across all dimensions.
   * The performance gap was most dramatic in 1D and 2D, where ES approached or reached the global minimum.
   * This demonstrates ES’s superior ability to navigate complex, multimodal functions.
2. **Scalability with dimension:**
   * Both algorithms faced challenges as dimensionality increased.
   * ES maintained better relative performance in higher dimensions, showing better scalability.
   * In 30D, ES achieved approximately 15.6% better final score with 16.8x faster execution.
3. **Computational efficiency:**
   * ES demonstrated significantly faster execution times across all dimensions.
   * ES efficiency (score reduction per second) was dramatically higher: 14.8x in 1D, 5.0x in 2D, and 20.0x in 30D.
4. **Effectiveness in finding the global optimum:**
   * ES was able to find the true global minimum in the 1D case with an MSE of just 0.000005.
   * In higher dimensions, both algorithms found local optima, with ES consistently finding better solutions.
5. **Robustness across problem instances:**
   * ES showed more consistent performance across different dimensionalities.
   * ES’s restart mechanism proved effective, with the console output showing multiple successful restarts.
   * GD’s effectiveness declined more dramatically with increasing dimensions.

These results demonstrate why evolutionary algorithms are often preferred for complex, multimodal optimization problems. The combination of stochastic search, adaptive parameters, and the ability to make large jumps in the search space gives ES a significant advantage over traditional gradient-based methods in these challenging landscapes.

# Conclusion

This study implemented and evaluated a (1+1)-Evolution Strategy with 1/5 Success Rule for optimizing the Rastrigin function, comparing its performance against Gradient Descent across different dimensionalities. The research objectives were successfully achieved, providing insights into the relative strengths of evolutionary approaches in multi-modal optimization.

The results clearly demonstrate the advantages of evolutionary approaches for complex optimization problems with multiple local optima. While gradient-based methods can be effective for simpler, unimodal functions, they often become trapped in local optima when faced with more complex landscapes.

The efficiency gap was particularly striking: ES achieved better results in a fraction of the time required by GD. This confirms that for multimodal problems like the Rastrigin function, Evolution Strategies offer not just better solutions but more computational efficiency.

Future work could explore more advanced ES variants like CMA-ES (Covariance Matrix Adaptation Evolution Strategy), which might provide even better performance in high-dimensional spaces, or investigate hybrid approaches that combine the strengths of both evolutionary and gradient-based methods.

This study successfully implemented and compared Grid Search and PSO for optimizing MLP hyperparameters for wine quality classification. PSO demonstrably outperformed Grid Search, achieving higher test accuracy (0.6238 vs. 0.5977) by identifying a superior set of hyperparameters. The objectives were met, providing valuable insights into the effectiveness of using PSO for neural network tuning, especially in navigating complex search spaces and handling class imbalance implicitly through finding better generalizing parameters.

# References

Eiben, AE & Smith, JE 2013, *Introduction to Evolutionary Computing*, Springer Science & Business Media.