DSA4212:
Natural Evolution Strategies for Blackbox Optimization

August 27, 2025

Group Composition:

1. Albin Erik Liljefors, A0307006U, e1384720@u.nus.edu
2. Jonathan Hvitved, A0293747M, e1345518@u.nus.edu
3. Alejandro Trigueros Alonso, A0293701H, e1345472@u.nus.edu

Contents

1 Introduction

Natural Evolution Strategies (NES) is a sub-set of algorithms belonging to the family of Evolution Strategies (ES). ES are a class of black-box optimization algorithms that utilize a heuristic, inspired by natural selection, to guide the next search direction. In every iteration, a new population of weight vectors are perturbed and evaluated against some objective function. Parameters with the best evaluation are recombined to form the population of the next generation. By repeating this procedure over multiple generations, any objective function can be optimized [SHC+17].

The NES algorithms are characterized by how they update the search distribution. From the parameterized search distribution, a set of search points are used and evaluated to adeptly capture the local structure of the fitness function[WSG+14]. With the samples, NES produce a search gradient, w.r.t to the parameters, towards higher expected fitness. With the search gradient, the algorithm takes a step in a gradient ascent along the natural gradient. The natural gradient is a second order method that prevents oscillations, undesired convergence and negative effects from the chosen parameterization. This process is re-iterated until convergence. In the family of NES-algorithim they all operate on the same foundation, but differ in type of distribution and gradient approximation method used. [WSG+14]

2 Theory of Natural Evolution Strategies (NES)

2.1 Core Principles of NES

Natural Evolution Strategies (NES) is a framework for black-box optimization that leverages the natural gradient to iteratively update a parameterized search distribution. The search distribution generates samples, which are evaluated using a fitness function. NES computes a search gradient that maximizes the expected fitness under the current distribution.

The gradient estimation employs the "log-likelihood trick," allowing the expected fitness gradient to be expressed in terms of the log-derivatives of the probability density function. This is defined as:

$$\nabla_\theta J(\theta) = \mathbb{E}_\theta \left[f(z) \nabla_\theta \log \pi(z|\theta) \right],$$

where θ are the parameters of the search distribution, $f(z)$ is the fitness function, and $\pi(z|\theta)$ is the density function of the search distribution. The natural gradient step incorporates the Fisher information matrix, $F(\theta)$, ensuring parameter updates are invariant to reparametrization:

$$\theta \leftarrow \theta + \eta F^{-1}(\theta) \nabla_\theta J(\theta),$$

where η is the learning rate. This approach enhances convergence stability and mitigates oscillations in optimization [WSG+14].

NES utilizes techniques such as fitness shaping, which ranks and normalizes fitness values to ensure robustness to scale transformations, and adaptive sampling, which adjusts learning rates

dynamically. These features collectively make NES well-suited for high-dimensional and multi-modal optimization problems.

2.2 Variants of NES

Natural Evolution Strategies (NES) encompasses a family of algorithms that differ in their choice of search distribution, gradient estimation methods, and implementation details. The key variants include:

- **Exponential NES:** This variant uses a multivariate Gaussian distribution parameterized by its mean and covariance matrix. The covariance matrix can be fixed, diagonal, or fully adaptive, with updates determined by the natural gradient [WSG+14].
- **Separable NES:** To improve scalability in high-dimensional optimization, Separable NES assumes a diagonal covariance matrix, reducing the computational cost of estimating and updating the natural gradient. This approach is particularly effective for problems where variables are independent or weakly correlated [WSG+14].
- **Covariance Matrix Adaptation NES:** This variant adapts the full covariance matrix during optimization, enabling NES to capture correlations between variables. While computationally intensive, it provides superior performance for complex, multimodal fitness landscapes [WSG+14].

These variants illustrate NES's flexibility in tailoring the search distribution to specific optimization challenges, balancing computational efficiency and precision as required.

2.3 Mathematical Formulation of NES

The search gradient is defined as the sampled gradient of expected fitness. Using the search gradient, NES-algorithms update the parameters of the search distribution, restricted to any distribution with existing derivatives of log-density w.r.t its parameters[WSG+14].

Let be the parameters with density function $\rho(\mathbf{z}, \theta)$ and fitness function $f(\mathbf{z})$, the expected fitness under the search direction is given by

$$F(\theta) = \mathbb{E}_\theta[f(\mathbf{z}] = \int f(\mathbf{z})\rho(\mathbf{z}, \theta)d\mathbf{z} \tag{1}$$

Using the log-likelihood trick and re-writing

$$\nabla_\theta F(\theta) = \nabla_\theta \int f(\mathbf{z})\rho(\mathbf{z},\theta)d\mathbf{z} \tag{2}$$

$$= \int f(\mathbf{z})\nabla_\theta \rho(\mathbf{z},\theta)d\mathbf{z} \tag{3}$$

$$= \int f(\mathbf{z})\nabla_\theta \rho(\mathbf{z},\theta)\frac{\rho(\mathbf{z},\theta)}{\rho(\mathbf{z},\theta)}d\mathbf{z} \tag{4}$$

$$= \int \Big[f(\mathbf{z})\nabla_\theta \log \rho(\mathbf{z},\theta)\Big]\rho(\mathbf{z},\theta)d\mathbf{z} \tag{5}$$

$$= \mathbb{E}\Big[f(\mathbf{z})\nabla_\theta \log \rho(\mathbf{z},\theta)\Big] \tag{6}$$

An estimate of the search gradient is thus given by the samples as

$$\nabla_\theta F(\theta) \approx \frac{1}{n}\sum_i^n f(\mathbf{z_i})\nabla_\theta \log \rho(\mathbf{z_i},\theta) \tag{7}$$

with population size n. With a gradient ascent procedure this provides a search direction in the space of search distributions. The update of search distribution is given by

$$\theta \leftarrow \theta + \gamma \nabla_\theta F(\theta). \tag{8}$$

This plain search gradient update can be both unstable and unsatisfying, requiring an extension that makes the updates invariant with respect to the parameterization used[WSG+14]. To resolve these issues, the natural gradient tries to remove any dependence on the used parameterization. This can be done by utilizing a more natural distance between probability distributions, such as the Kullback-Leibler divergence. The natural gradient is then given according to [WSG+14] as the solution to the constrained optimization problem

$$\max_{\delta\theta} F(\theta + \delta\theta) \approx F(\theta) + \delta\theta^T \nabla_\theta F, \tag{9}$$

$$\text{s.t} D(\theta + \delta\theta||\theta) = \varepsilon, \tag{10}$$

for $\lim \delta\theta \to 0$

$$D(\theta + \delta\theta||\theta) = \frac{1}{2}\delta\theta^T \mathbf{F}(\theta)\delta\theta \tag{11}$$

where

$$\mathbf{F}(\theta) = \int \rho(\mathbf{z},\theta)\nabla_\theta \log \rho(\mathbf{z},\theta) \log \rho(\mathbf{z},\theta)^T d\mathbf{z} \tag{12}$$

$$= \mathbb{E}\Big[\nabla_\theta \log \rho(\mathbf{z},\theta) \log \rho(\mathbf{z},\theta)^T\Big] \tag{13}$$

is the Fisher information matrix of the given parametric family of search distributions. The solution to equation 9 is given by [WSG+14] as

$$\mathbf{F}\delta\theta = \beta\nabla_\theta F \tag{14}$$

, If $\mathbf{F}$ is invertible this leads to the natural gradient as

$$\tilde{\nabla}_\theta F = \mathbf{F}^{-1}\nabla_\theta F(\theta) \tag{15}$$

utilizing the natural gradients, the parameters can be updated without using the steepest ascent.

$$\theta \leftarrow \theta + \alpha\mathbf{F}^{-1}\nabla_\theta F. \tag{16}$$

where both $\mathbf{F}$ and F are to be computed by the log-derivatives.

2.4 Application Potential of NES in Optimization Problems

Natural Evolution Strategies have gained recognition for handling optimization problems that traditional methods struggles with. With sampling-based gradient estimation, NES can operate effectively without explicit gradient calculations. NES have shown strong performance in high-dimensional and multidimensional optimisation problems. Additionally, NES have provided significant performance in machine learning and mathematics [NO22]

.

In hyperparameter tuning, high-dimensional optimization is required to tune numerous parameters. Utilizing NES sampling-based approach can be highly efficient as it navigates large, complex landscapes without the need to differentiate the objective function. Nomura and Ono (2022) have shown that NES can be a practical alternative to gradient-based optimization. By adopting a natural gradient, they showed how to efficiently search across a high-dimensional parameter space to optimize neural network configurations [SHC+17].

2.5 Challenges and Limitations of NES

Despite its strengths, NES faces certain limitations. The reliance on sampling for gradient estimation makes NES computationally intensive, particularly for high-dimensional problems where larger populations may be required for stable convergence. Additionally, the necessity of computing and inverting the Fisher information matrix introduces further complexity, particularly in cases where this matrix is large or ill-conditioned. Regularization techniques, such as adding small diagonal terms to the Fisher matrix, are often employed to mitigate numerical instability.

Another challenge lies in the sensitivity to hyperparameter choices, such as learning rate and population size. Poorly chosen hyperparameters can lead to suboptimal exploration or slow convergence. Furthermore, NES may struggle with optimization landscapes featuring extremely narrow valleys or large flat regions, where the natural gradient's effectiveness can be limited without tailored adjustments to the search distribution [WSG+14].

3 Method

In this projected we have implemented versions of natural evolution strategies to optimise three different problems of high relevance in the field of machine learning and applied mathematics.

3.1 General Training Framework

The NES training process consists of generating a set of samples in each generation and assigning higher weights to those with superior fitness. This weighting approach ensures that high-fitness samples have a larger influence on the parameter updates, effectively guiding the search process in each iteration. Samples are drawn from a Gaussian distribution, denoted as $\mathbf{z_i} \sim \pi(w, \sigma)$, where the parameters of the distribution evolve over generations.

For each NES variation, the log-derivatives are computed as follows so that the gradient estimate and Fisher matrix can be formed from the sampled points. This allows for the estimation of the natural gradient to guide parameter updates. The specific calculations may vary based on the objective function, with details provided in subsequent sections.

3.2 Rosenbrock function

Because of its significance as a benchmark function for optimization problems, we implemented a program to investigate the convergence for a d-dimensional rosenbrock function. The fitness function was evaluated as,

$$f(x) = \sum 100 \times (x_{i+1} - x_i^2)^2 + (1 - x_i)^2 \tag{17}$$

where x represents each samples in every generation.

To simplify implementation while ensuring efficiency and robustness, the following details were incorporated:

1. **Sample Generation**: New samples are generated by perturbing the current solution, w, with Gaussian noise, where the noise's standard deviation is defined by σ.

2. **Fitness Normalization**: Fitness values are normalized to make updates invariant to monotonic transformations, ensuring stability across generations regardless of the function's scale.

3. **Adaptive Sampling for Learning Rate**: An adaptive sampling technique adjusts the learning rate based on feedback from the natural gradient, allowing the algorithm to dynamically fine-tune its convergence speed.

4. **Exponential Decay of σ**: The standard deviation, σ, is updated through exponential decay, ranging between predefined maximum and minimum values, to gradually reduce noise as the solution approaches optimality.

3.3 Neural Network

The Natural Evolution Strategies (NES) algorithm is applied to optimize the parameters of a neural network for a regression task. The networks consist of fully connected layers with weights and biases optimized directly through NES.

Fitness Function. The fitness function is defined as the negative mean squared error (MSE) between the network's predictions and the true function (the function we will try to model):

$$y = \sin(\pi x_1) + \cos(\pi x_2),$$

where the inputs $x_1, x_2 \in [-1, 1]$.

Fitness Shaping. Fitness shaping is implemented to improve stability and convergence by transforming raw fitness scores into utility values. This transformation ranks the fitness scores and computes utilities as:

$$u_i = \max\left(0, \log\left(\frac{\text{population size}}{2} + 1\right) - \log(\text{rank}_i + 1)\right),$$

where u_i are normalized to have a mean of zero, ensuring that selection pressure is balanced and reducing the risk of premature convergence.

Standard Deviation Updates. The standard deviation (σ) of the Gaussian sampling distribution is updated iteratively using the formula:

$$\sigma \leftarrow \sigma \exp\left(\frac{\text{learning rate}}{2} \cdot \nabla_\sigma J\right),$$

as proposed in [WSG+14]. To ensure stability, σ is clipped to remain within predefined bounds $[\sigma_{\text{min}}, \sigma_{\text{max}}]$.

Robustness Measures. To improve numerical stability, the implementation includes:

- Regularization of the Fisher information matrix by adding a small diagonal term.
- Clipping of σ within predefined bounds to avoid divergence or stagnation during updates.

3.4 Sphere-10

A classic convex optimization benchmark:

$$f(x) = -\sum x_i^2, \quad x \in \mathbb{R}^{10}$$

The implementation includes adaptive features from Section 3.2:

- Fitness shaping through normalization: $(f - \mu)/(\sigma + \epsilon)$
- Natural gradient computation via Fisher Information Matrix
- Adaptive learning rate sampling to improve convergence stability
- Exponential sigma decay: $\sigma = \sigma_{\text{min}} + (\sigma_{\text{max}} - \sigma_{\text{min}})e^{-kt}$

3.5 Flame Performance Optimization:

A 2D engineering problem optimizing flame performance through nozzle parameters (center position, width). For this computationally expensive problem, we incorporated CMA-ES style adaptations:

- Evolution path tracking: $p = (1 - c)p + \sqrt{c(2 - c)}\nabla$
- Adaptive step size based on path length:

$$\sigma \leftarrow \sigma \times \exp\left(\frac{\|p\| - E\|p\|}{d \times E\|p\|}\right)$$

$$E\|p\| = \sqrt{2} \times \sqrt{\text{dim}} \quad \text{for expected path length}$$

- Higher initial σ to encourage exploration

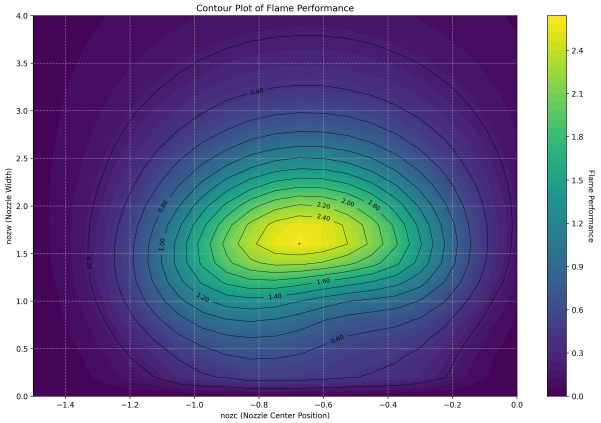

Figure 1: Contour plot of flame performance as a function of nozzle center position and width.

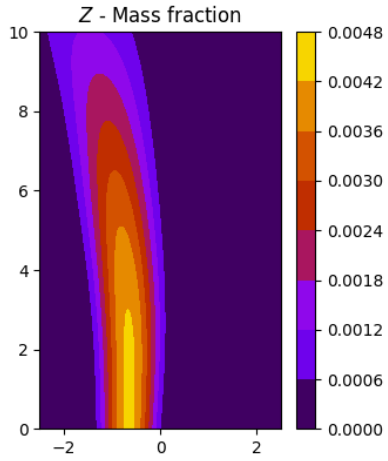

Figure 2: Mass fraction distribution of Z component across the domain.

This implementation combines elements from subsection 2.1 (Natural Gradient) and subsection 2.2 (CMA-NES variant), while adding problem-specific adaptations for the engineering application.

3.6 Performance Benchmark

3.6.1 Rosenbrock

The experiments on the Rosenbrock function were conducted with the following configurations:

- **Dimensions**: `[2, 5, 10, 25]`
- **Population Sizes**: `[50, 500]`
- **Maximum Generations**: `100,000`
- **Learning Rate**: `0.001`
- **Sigma**: `[0.1, 0.0001]`

These parameters were selected to capture performance variations as the complexity of the objective function increases with dimension, allowing the algorithm to adapt across configurations. The variation in population sizes provided flexibility in exploration, while the learning rate and sigma values were initialized to balance convergence speed with robustness.

To investigate the performance of NES with varying dimension and population sizes, the following key metrics were saved for further analysis:

1. **Best Solution:** The best set of parameters, guided by their evaluation on the fitness function in subsequent generations.
2. **Convergence:** Fitness evaluation in each generation.
3. **Relative Error History:** Relative error between the true solution and the approximate.

3.6.2 Neural Network

The performance benchmark for the neural network optimization consists of evaluating NES across the following configurations:

- **Simple architecture:** [2, 10, 1]
- **Complex architecture:** [2, 10, 20, 10, 1]
- **Population sizes:** 50 and 500

A uniform training set of 100 samples is drawn from the input space $[-1, 1]$ to define the optimization problem. The NES algorithm tracks the MSE across generations, visualizes true vs. predicted outputs, and generates error heatmaps to evaluate its optimization behavior. Each configuration's sampling is informed by dynamically updated Gaussian parameters (μ and σ), ensuring adaptability during the optimization process.

Grid-based sampling over the input space is used to evaluate prediction accuracy and error distribution for the neural networks. This setup provides insights into the interplay between NES parameters and the complexity of the optimization landscape.

3.6.3 Sphere-10

Parameter	NES A	NES B	GD
Dimensions	[2, 5, 10, 25]	[2, 5, 10, 25]	[2, 5, 10, 25]
Population	[50, 500]	[50, 500]	N/A
Max Iterations	5000	5000	5000
Learning Rate	0.01	0.001	0.001
Initial Sigma	0.1	0.001	N/A
Error Tolerance	10^{-5}	10^{-5}	10^{-5}

Table 1: Experimental configurations for Sphere-10 function optimization. NES A prioritizes fast convergence while NES B and GD aim for higher precision.

The configurations were designed to explore NES's versatility: Configuration A prioritizes convergence speed while Configuration B aims for higher precision, allowing direct comparison with GD's performance.

3.6.4 Flame Optimization

Parameter	NES	GD (Finite Diff.)
Learning Rate	0.01	0.01
Population/Step Size	50	10^{-4}
Max Iterations	100	200
Path Learning Rate (c)	0.1	N/A
Sigma Damping (d)	1.0	N/A

Table 2: Comparison setup for flame performance optimization experiments.

For the flame optimization, CMA-ES adaptations were incorporated to handle the irregular performance landscape typical in engineering applications. Both experiments tracked convergence rate, computational overhead, and final solution quality against gradient-based approaches.

4 Results and Discussion

4.1 Rosenbrock

This section examines the convergence performance of the NES implementation for the Rosenbrock function. The experiments were conducted with the following configurations of parameters.

- Dimensions: `[2, 5, 10, 25]`
- Population: `[50, 500]`

- Maximum Generations: $100\ \ 000$
- Learning rate: 0.001
- Sigma: 0.001

These parameters were chosen to highlight performance differences as the objective function complexity increases, with varying population sizes to allow greater flexibility in exploration. The learning rate and sigma were initialized to balance efficiency with robustness, ensuring stable convergence across configurations.

4.1.1 Convergence Analysis of the Objective Function

The log-log convergence plot highlights exponential improvements and variations in parameter evaluations throughout training. In each generation, the updated set of parameters is evaluated against the objective function, demonstrating a gradual convergence towards zero.

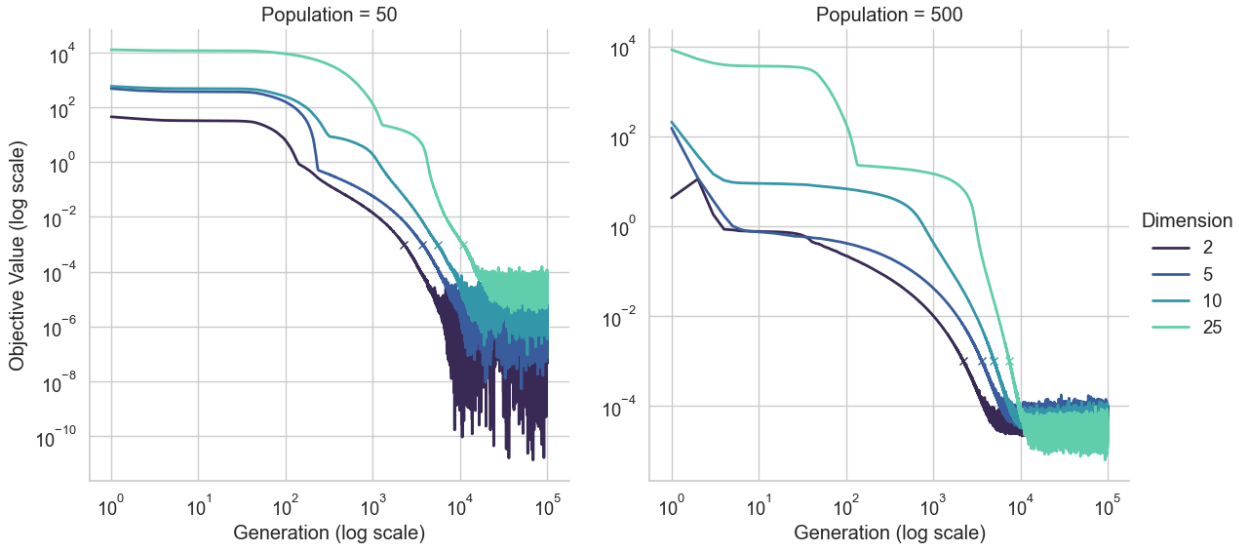

Figure 3: Convergence of NES on the Rosenbrock function over generations. The log-log plot shows the parameter evaluation on the objective function for different dimensions and population sizes.

Objective function convergence: Convergence order depends on the dimension, with dimensions 2, 5, 10, and 25 converging in that sequence. Larger populations (500) lead to earlier convergence across all dimensions as can be seen in fig 3 and table 3, indicating that NES benefits from a broader exploration set, improving convergence speed.

4.1.2 Convergence Analysis of Relative Error

The linear-log convergence plot provides an analysis of the relative error between each parameter update and the true solution. The relative error is calculated using the L^2 norm as:

$$\text{Relative Error} = \frac{\|\text{True Solution} - \text{Approximate Solution}\|_2}{\|\text{True Solution}\|_2}$$

This metric captures the proportional distance between the current parameter set and the optimal solution, highlighting the algorithm's progress in reducing error over time. The linear-log scale allows for detailed observation of early convergence behavior, showcasing how quickly the NES algorithm reduces error across generations.

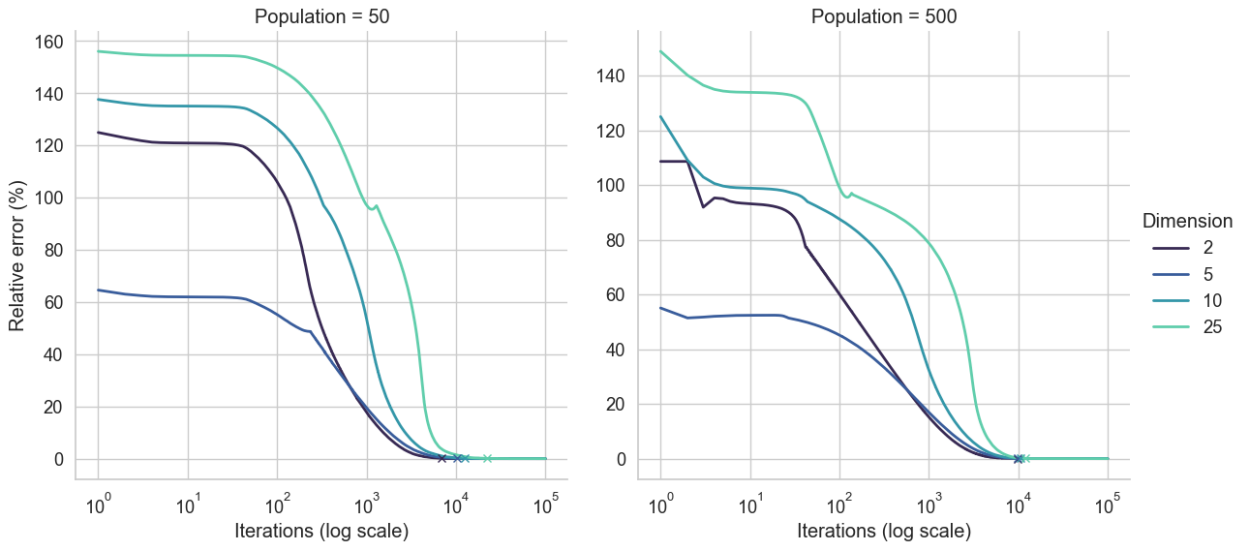

Figure 4: Convergence of NES on the Rosenbrock function over generations, shown as relative error.

Relative Error Convergence: Dimension 5 consistently begins with the lowest relative error, and dimension 2 typically converges fastest. However, with a population of 500, dimension 5 converges first, showing an inverse dependency of convergence speed on population size. For instance, dimension 2 converges around 30% slower with a population of 500, whereas dimension 25 converges 50% faster.

These findings suggest that NES tuning should consider dimension and population size to optimize convergence efficiency.

4.1.3 Threshold

To determine when the solution is sufficiently accurate, we used threshold values for both the objective function and relative error, set at 0.001, 0.1%, respectively. These thresholds were chosen to allow the program to potentially terminate early once the solution reached an acceptable level of precision, avoiding unnecessary exploration in further generations. Table 3 summarizes the

generation count at which each dimension and population size first met these thresholds, providing insight into how quickly the NES algorithm achieves convergence under different conditions.

Table 3: Generation at Thresholds for Objective Value and Relative Error

Dimension	**Population**	**Objective Value** < 0.001	**Relative Error** $< 0.1\%$**)**
2	50	2278	6980
5	50	3709	10408
10	50	5530	12483
25	50	10814	22062
2	500	2210	9804
5	500	3615	9730
10	500	4974	10546
25	500	7480	12140

4.2 Neural Networks

4.2.1 Performance Analysis

Figures 5, 6, and 7 summarize the performance of NES across the four configurations. Key observations include:

- **[2, 10, 1] with Population 50:** This configuration achieves low and stable MSE quickly (Figure 5), producing accurate predictions (Figure 6) with uniformly low errors across the input space (Figure 7). Its simplicity aligns well with NES's capabilities, offering a robust and efficient solution.
- **[2, 10, 1] with Population 500:** A larger population improves NES's exploration, resulting in slightly better long-term accuracy and reduced errors in regions of high curvature. However, the computational cost outweighs the marginal performance gains.
- **[2, 10, 20, 10, 1] with Population 50:** This setup performs the worst, with unstable convergence and high error concentrations. The small population is insufficient for effective NES optimization in the high-dimensional parameter space, resulting in poor generalization.
- **[2, 10, 20, 10, 1] with Population 500:** The larger population stabilizes NES slightly, but the high-dimensional architecture remains inefficient for this relatively simple target function. Errors remain widespread, and the overall performance is suboptimal.

These results highlight that NES performs best with simpler architectures, such as [2, 10, 1], where it can effectively balance exploration and exploitation. Overly complex architectures introduce optimization challenges, particularly with small populations, and fail to leverage NES's strengths.

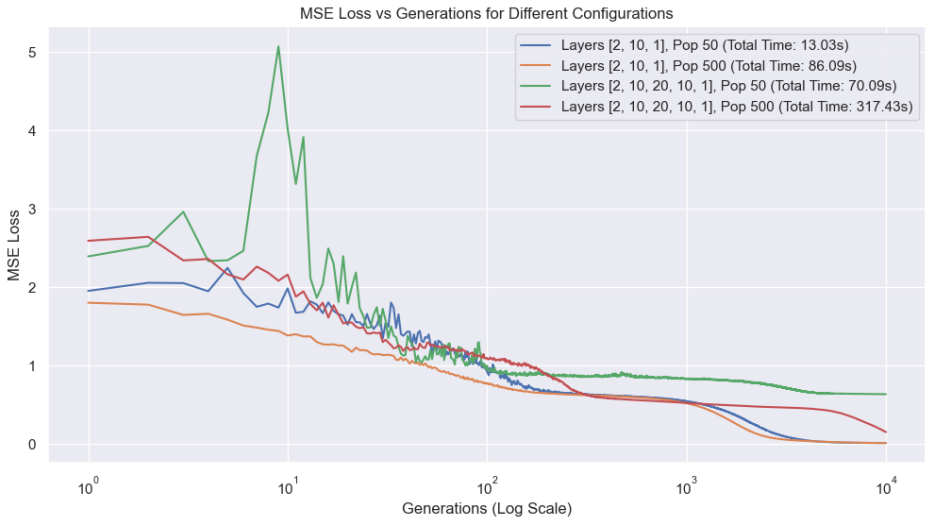

Figure 5: MSE Loss vs Generations for Different Configurations.

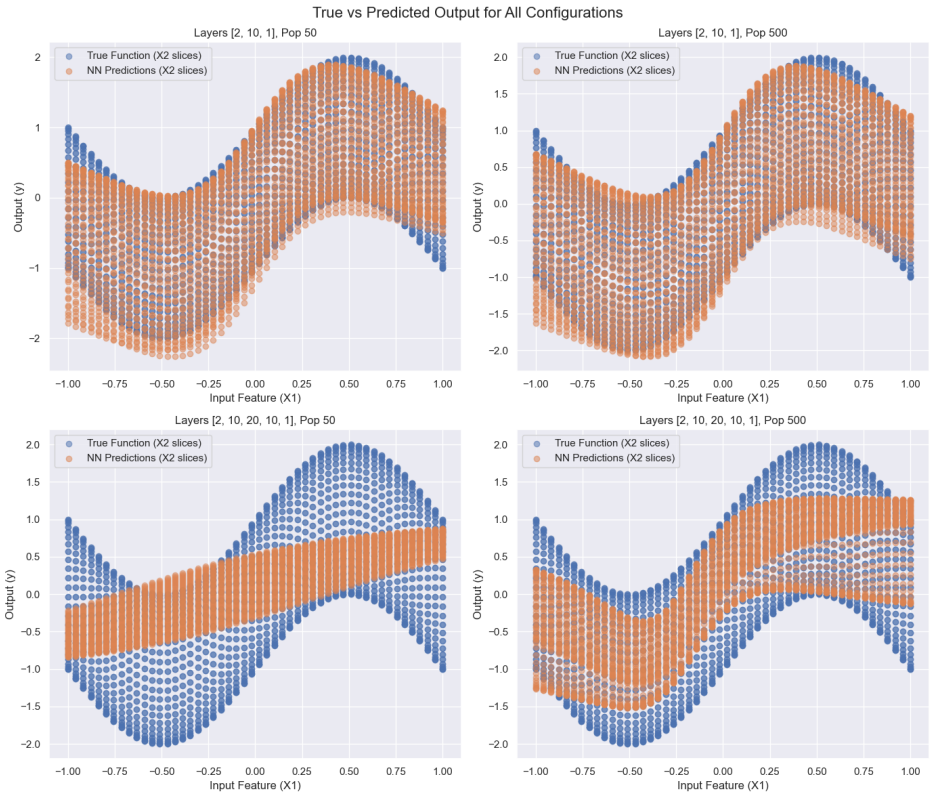

Figure 6: True vs Predicted Output for All Configurations. Each subplot corresponds to one configuration, comparing the true sampling function (blue) and the network predictions (orange).

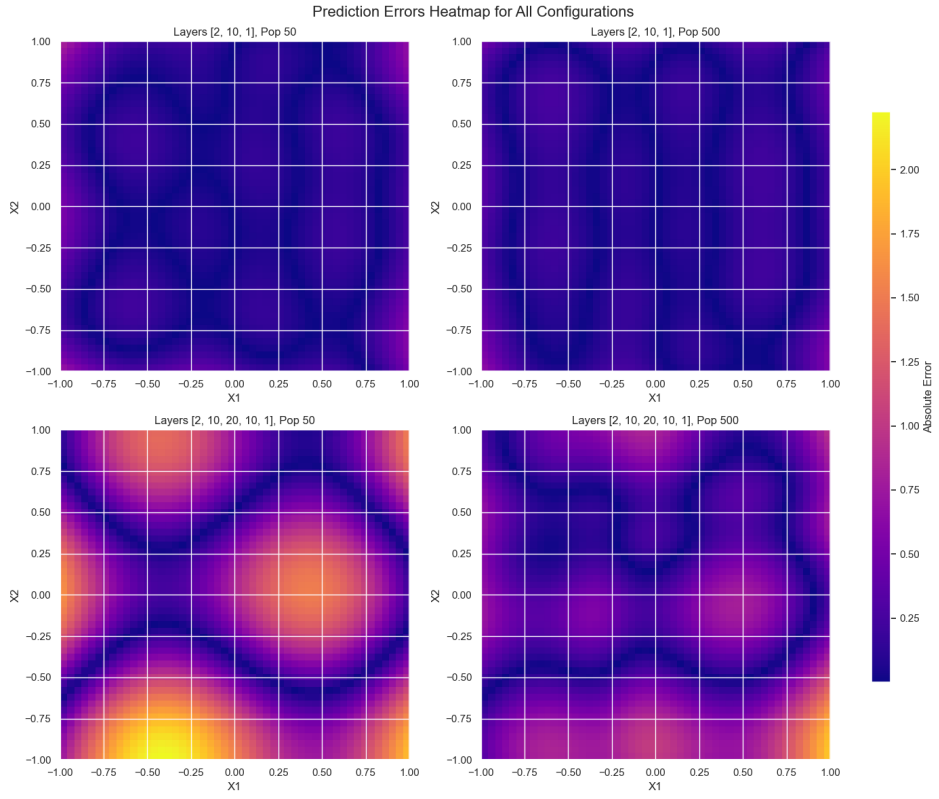

Figure 7: Prediction Errors Heatmap for All Configurations. Each subplot shows the absolute error distribution across the input space for a given configuration.

4.2.2 Discussion

The results underline the importance of matching NES parameters to the problem's complexity:

- **Architecture Simplicity:** Simpler architectures like [2, 10, 1] allow NES to efficiently optimize the parameter space, delivering low error and stable convergence. For the given target function, additional complexity provides no benefit and hinders optimization.
- **Population Size:** While larger populations improve parameter space exploration, their benefits diminish in simpler setups. For complex architectures, larger populations stabilize optimization but cannot compensate for the inefficiencies caused by overparameterization.
- **Target Function Simplicity:** The simplicity of the target function exacerbates the mismatch between NES and complex architectures, leading to instability and poor performance.

Overall, NES demonstrates strong performance when paired with simpler architectures and appropriately sized populations. For this problem, [2, 10, 1] with Population 50 provides the best tradeoff between speed, accuracy, and computational cost.

4.3 Sphere-10

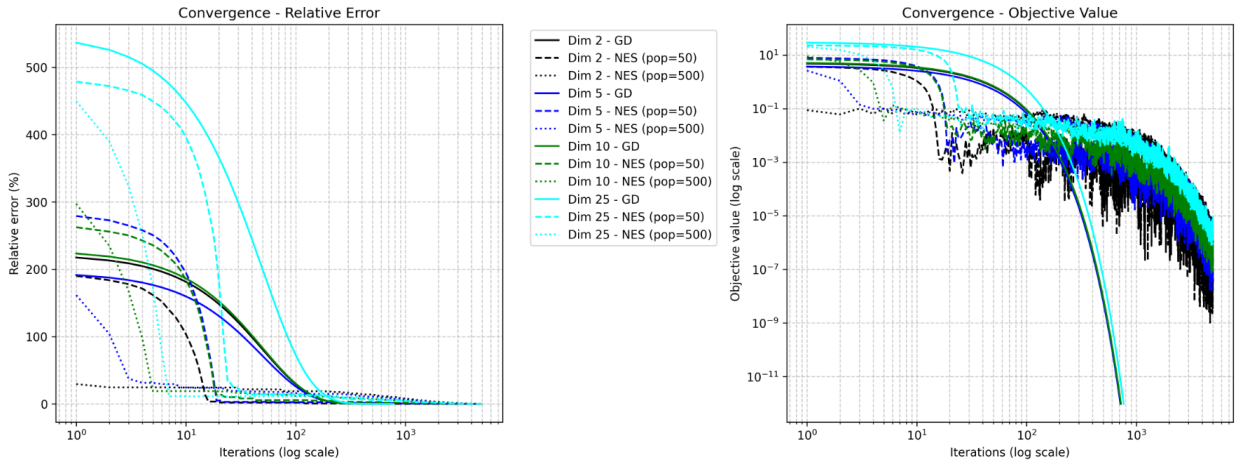

Figure 8: Convergence plots for Sphere-10 function optimization showing relative error (left) and objective value (right) for NES and Gradient Descent across different dimensions. The plot compares NES with small and large population sizes (50 and 500) against GD in terms of convergence rate and objective value achieved.

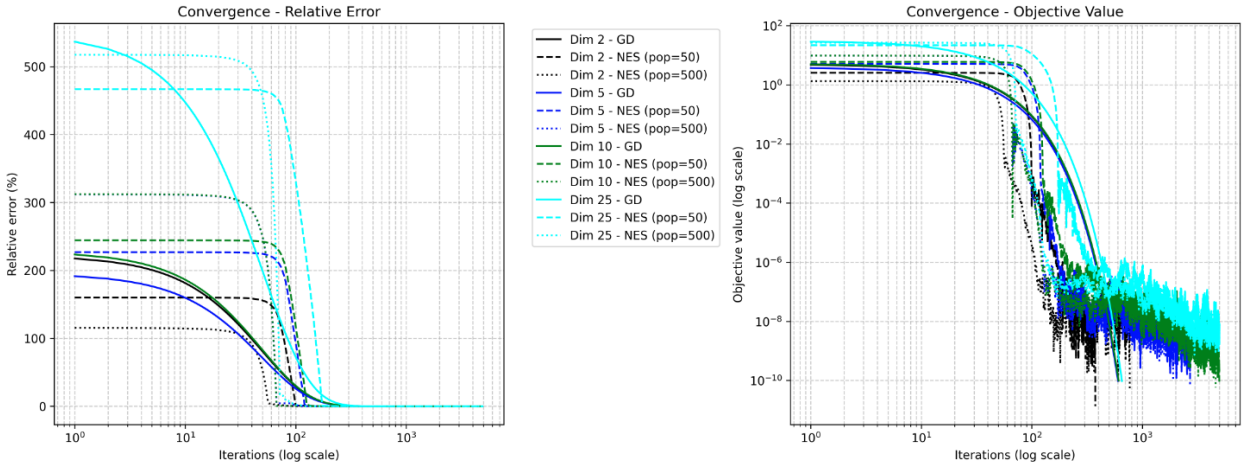

Figure 9: Detailed comparison of convergence for Sphere-10 function optimization with reduced learning rate and sigma (Configuration B). This figure highlights the improved accuracy achieved by both NES and GD with more conservative parameter settings, particularly for higher dimensions.

4.4 Sphere-10 Function Optimization

We compared NES against gradient descent on the Sphere-10 function across dimensions [2, 5, 10, 25] and population sizes [50, 500]. Two parameter configurations were tested:

Config A: Learning rate = 0.01, $\sigma = 0.1$
Config B: Learning rate = 0.001, $\sigma = 0.001$

Method	Config	Dim=2 Error	Dim=10 Error	Time (s)
GD	A	9.86×10^{-6}	9.93×10^{-6}	0.67
NES (pop=50)	A	3.19×10^{-5}	2.75×10^{-4}	8.18
NES (pop=50)	B	3.67×10^{-6}	9.22×10^{-6}	1.19

Table 4: Error and time comparison for NES and GD on Sphere-10 function.

4.5 Key Findings

- **Config B achieved better accuracy** (10^{-10} vs 10^{-6}) with lower computational cost (1.2s vs 8.2s).
- **Smaller populations (50)** excelled in lower dimensions.
- **Larger populations (500)** only benefited high-dimensional cases ($d = 25$).
- **Parameter tuning** is crucial for NES performance.

These results demonstrate that NES can match GD's precision while maintaining black-box optimization advantages.

4.6 Flame Performance

The flame optimization problem tests NES's ability to handle a complex engineering system with two parameters: nozzle center position (nozc) and nozzle width (nozw). The goal is to maximize flame performance by finding the optimal nozzle parameters.

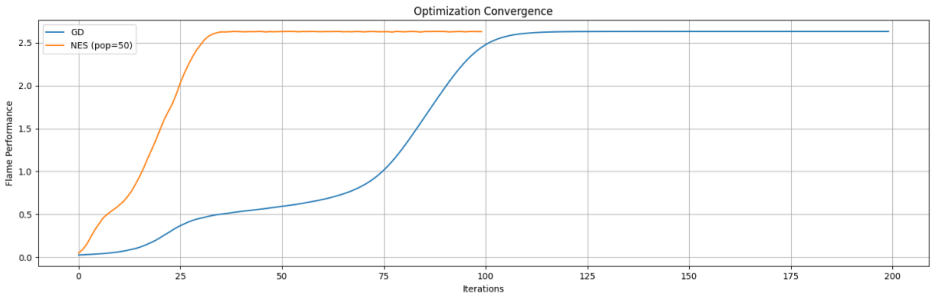

Figure 10: Optimization convergence plot comparing NES (population=50) and gradient descent for flame performance optimization.

The convergence comparison between NES (population=50) and gradient descent reveals:

1. Faster initial convergence: NES reaches near-optimal performance in 25 iterations

2. Earlier stabilization: NES converges to its final value around iteration 30

3. Gradient descent requires approximately 100 iterations to reach similar performance levels

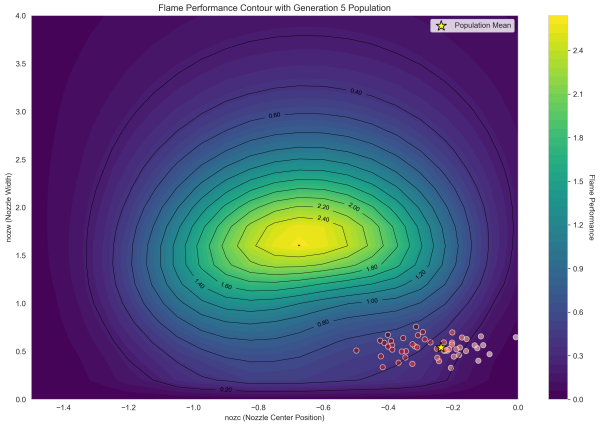

Figure 11: Generation 5 Population: Initial exploration of parameter space with wide spread of population points.

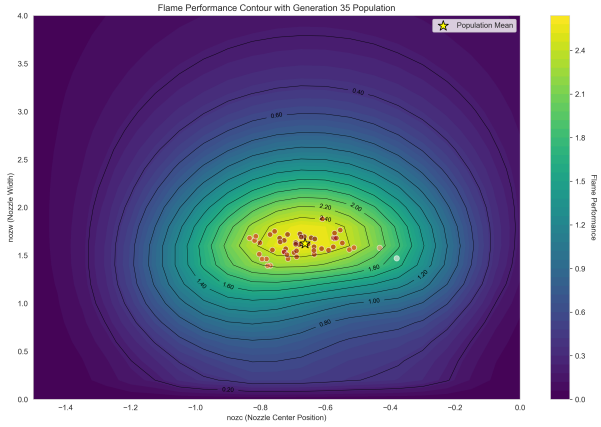

Figure 12: Generation 35 Population: Convergence near optimal region with clustered population points.

The flame performance contour plot illustrates:

1. **Optimal region:** Peak performance occurs at $\mathrm{nozw} \approx 1.6$ and $\mathrm{nozc} \approx -0.7$

2. **Search behavior:** Population points (shown in red) demonstrate how NES explores the parameter space

3. **Clear convergence:** The algorithm successfully identifies the global maximum region (yellow peak)

Performance Metrics:

Method	Time (s)	Final Performance
NES	250.12	2.6210
GD	43.24	2.6316

Table 5: Comparison of computation time and final performance for NES and gradient descent on the flame optimization problem.

While NES converges in fewer generations, its computational cost is 6x higher than gradient descent. For this well-behaved problem, both methods reached optimal performance. Although NES's global exploration capabilities could prove valuable for more complex engineering problems, computational constraints limited further investigation.

References

[NO22] Masahiro Nomura and Isao Ono. Fast moving natural evolution strategy for high-dimensional problems, 2022.

[SHC+17] Tim Salimans, Jonathan Ho, Xi Chen, Szymon Sidor, and Ilya Sutskever. Evolution strategies as a scalable alternative to reinforcement learning, 2017.

[WSG+14] Daan Wierstra, Tom Schaul, Tobias Glasmachers, Yi Sun, Jan Peters, and Jürgen Schmidhuber. Natural evolution strategies. *Journal of Machine Learning Research*, 15(27):949–980, 2014.