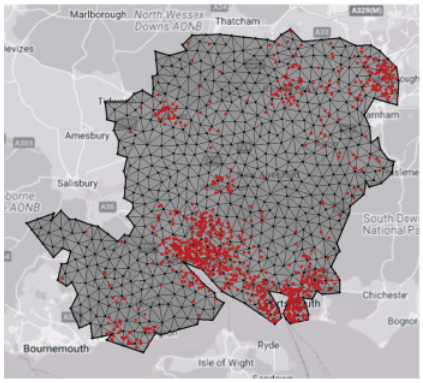
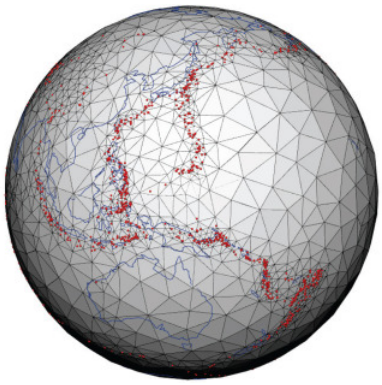
Projet de Recherche (PRe)

Major: Applied mathematics

Academic Year: 2024-2025

Advanced optimization for physics-informed statistical learning





Non-confidential

Author : Arnaud PELISSIER
Promotion : 2026
ENSTA Paris Supervisor: Andrea SIMONETTO
Host Organization Supervisor : Laura Maria SANGALLI
From: 26/05/2025 **To**: 24/08/2025
Host Organization: Politecnico di milano, Department of Mathematics

Intentionally Left Blank

1 Abstract

This report focuses on the analysis of the performance of numerous optimization methods on the problem of density estimation. Given a list of points on a manifold, sampled from some unknown distribution, we wish to reconstruct the density function. The method developed by [13] solves this problem by discretising the domain and modeling the density as some high dimensional vector, and chosing the density that minimizes a certain error 4. This method require high dimensional minimization of the error, and the calibration of two smoothing parameters by minimization of the cross-validation error.

The contribution of this report is the folowing:

- Efficient implementation of multiple optimization methods, using the most recent analysis of the performance of these algorithms.
- The analysis of the performance of these methods for low dimensional optimization against SciPy [9].
- The analysis of the performance of these methods in the context of nonparametric space density estimation, using [13], on simulated and on real world data.
- The analysis of the performance of multiple low dimensional optimization methods for parameter calibration for space-time density estimation.

Key words: Optimization, Statistics, Density estimation

2 Acknowledgment

INSERT Acknowledgment, Laura Maria Sangalli, Alessandro Palumo, Andrea Simonetto

Contents

3 Introduction

Density estimation is an important tool in statistics as it can be used for visualization purposes ["and is a starting point for classification and regression]. The input of a density estimation problem is a list of samples observed from any kind of unknown physical or anthropological process. For example, the space-time distribution of crimes or road accidents in a city, the position of earthquakes in the world 1. The output is an estimation of the real density of the process over the whole domain.

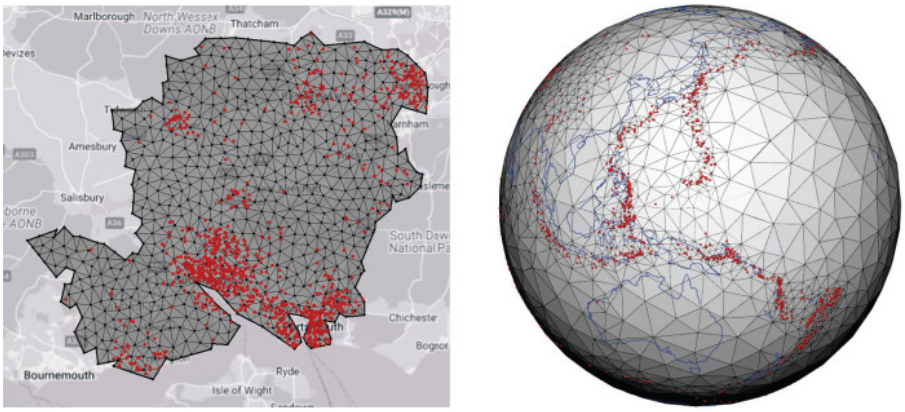


Figure 1: Discretization and point samples of infections in the region of Southampton, Hampshire county, UK (left) and earthquakes in earth's surface (right)

[10] proposes a nonparametric approach where the density is chosen by minimizing the log-likelyhood of the data. Moreover a penalization term is used to favor smooth densities. This approach raises the challenge of the numerical representation of the function. [10] uses finite elements on a discretized planar domain, and uses differential operators for the penalazation term. [12] extended this method on 2D shaped embedded in a 3D space such as a sphere which enable the density estimation process to be performed on a wider wariety of cases including worldwide geographical data. Its contribution is the use of differential geometry to analyse the problem, and the use of the Laplace-Beltrami operator, which generalize the laplacian for Rimannian manifolds and is used for the penalization term. [13] analyses various theoretical properties of the previous method and extends it the space-time data by discretizing using finite elements in space and B-splines in time. The work [14] extends this method to linear network which is usefull [REPHRASE] to analyse phenomena such as road accidents on a city's road network.

These methods selects a density that minimizes of the penalized likelyhood error, and thus require the use of optimization method on domains that often have thousands, to tens of thousand dimensions. In this context it is crutial to develop fast and efficient algorithms. Moreover, in the context of space-time regresion, two parameters are used to control the penalization term, which adds a calibration step where the expensive to compute cross-validation error must be minimized over a 2D domain. The aim of this work is to efficiently implement and analyse the performance of multiples optimization methods in the context of the nonparametric density estimation method described by [13] and [14].

EXPLAINATION OF OLD R PACKAGE VS NEW

The current C++ package is divided into a main package and a core package. The core

package contains basic functionalities such as finite element solver for second-order linear elliptic boundary value problems, linear and non-linear system solvers. It also includes unconstrained nonlinear optimization methods, which are the focus of this report's analysis. The main library implements the density estimation mentionned above as well as other models, as it is only a part of a wider set of models based on statistical learning from the fdaPDE project. INCLUDE OTHER METHODS

First, we consider multiple optimization methods and derive an efficient implementation of these in our C++ API 5. In A, we benchmark their performance against state-of-the-art implementations. We then analyse the strenghts and weaknesses of each method in the context of a high dimensional problem: the nonparametric density estimation over complicated domains using the method developed in [13].

4 Density estimation model

4.1 Mathematical aspect

Let $\mathcal{D} \subset \mathbb{R}^d$ be the spacial domain of interest and $[0,T], T \in \mathbb{R}^+$ its temporal domain. For our purpose, $d \in \{2,3\}$ and $\mathcal{D}$ is a compact domain. The unknown distribution is $f : \mathcal{D} \times [0,T] \mapsto \mathbb{R}^+$ and the points sampled from it are denoted by $\{(\overrightarrow{p_i}, t_i), i = 1..N\}$ with $\forall i = 1..N, \overrightarrow{p_i} \in \mathcal{D}$ and $t_i \in [0,T]$.

The method does not assume any prior form for the density f, instead we consider every possible densities $\{f : \mathcal{D} \times [0,T] \mapsto \mathbb{R}^+$ such that $\forall t \in [0,T], \int_{\mathcal{D}} f(\overrightarrow{p}, t) d\overrightarrow{p}) = 1\}$ and find the one that minimizes some error.

In practice, we use the log-density $g := \log f$ to get rid of the constrain of positivity of f and we add a penalization term to force the total weight of the density to be 1. Thus the error $L(g)$ is given by:

$$L(g) := -\frac{1}{n} \sum_{i=1}^{n} g(\overrightarrow{p_i}, t_i) + \int_{\mathcal{D}_T} e^{g(\overrightarrow{p_i}, t_i)} \overrightarrow{dp} dt + R(g, \overrightarrow{\lambda})$$

Where:

- $-\frac{1}{n} \sum_{i=1}^{n} g(\overrightarrow{p_i}, t_i)$ is the negative log-likelyhood
- $\int_{\mathcal{D}_T} e^{g(\overrightarrow{p_i}, t_i)} \overrightarrow{dp}$ Enforces the constraint that $\forall t \in [0,T], \int_{\mathcal{D}} f(\overrightarrow{p}, t) d\overrightarrow{p} = 1$
- $R(g, \overrightarrow{\lambda})$ is a smoothing term that becomes larger as g gets rougher. $\overrightarrow{\lambda} \in \mathbb{R}^{+,2}$ are the smoothing parameters.

The roughness penality term is defined as followed:

$$R(g; \overrightarrow{\lambda}) := \lambda_D \int_{D_T} (\Delta_D g(\mathbf{p}, t))^2 \, d\mathbf{p} \, dt + \lambda_T \int_{D_T} \left(\frac{\partial^2 g(\mathbf{p}, t)}{\partial t^2} \right)^2 d\mathbf{p} \, dt.$$

. The proof that second term does indeed forces g to be a log-density can be found in Appendix A of [13].

4.2 Numerical aspect

INSERT [DISCRETIZATION PROCESS: MESH AND OTHER DETAILS LIKE MANIFOLD AND LINEAR NETWORKS] HERE

The model as is take two meta parameters $\lambda_{\mathcal{D}}$ and λ_T which needs to be calibrated. This calibration is performed by minimizing the K fold cross-validation error, which is computer by spliting the dataset into K lists of points and deriving $(\text{Train}_i, \text{Test}_i), i = 1..K$ subsets. For each pair of train and test sets, $L(g)$ is minimized with the train set and its negative log likelyhood is computed on the test set. The K fold cross validation error is the average of these negative log-likelyhood errors. The algorithm works as folowed:

Algorithm 1 K-Fold Cross-Validation

Require: Dataset $D = \{(\overrightarrow{p_i}, t_i), i = 1..N\}$, number of folds K
Ensure: Average negative log-likelihood
1: Split the dataset D into K roughly equally-sized lists of points: $\{D_1, D_2, \ldots, D_K\}$
2: **for** $k \leftarrow 1$ to K **do**
3: Define $D_{\text{train}} \leftarrow D \setminus D_k$ ▷ Training set
4: Define $D_{\text{val}} \leftarrow D_k$ ▷ Validation set
5: Minimize $L(g)$ using D_{train}
6: $\text{err}_k \leftarrow -\frac{1}{|D_{\text{Test}}|} \sum_{(\overrightarrow{p},t)\in D_{\text{Test}}}^{n} g(\overrightarrow{p}, t)$ ▷ Negative log-likelyhood
7: **end for**
8: **return** $\frac{1}{K} \sum_{k=1}^{K} \text{err}_k$ ▷ Average negative log-likelihood

5 Optimization methods

In this section, we introduce the optimization methods that will be used in the rest of the report. for each of these, we will discuss their known trenghts and weaknesses and derive an efficient implementation.

We will denote $f : \mathbb{R}^d \mapsto \mathbb{R}$ the objective function that needs to be minimized. Here, $d \in \mathbb{N} - \{0\}$ is the dimension of the problem. Most of the algorithm considered works by generating some sequence $\{x_k\}_{k\in\mathbb{N}}$ where x_0 is a starting point that for our purose is arbitrary, but is to be chosen by the user of the algorithm depending on the optimization problem. Moreover, we will denote $y_k = \nabla f(x_{k+1}) - \nabla f(x_k)$ the difference of gradients at each step and $s_k = x_{k+1} - x_k$ the difference between each step.

FOR EACH OF THE FOLOWING, ADD SOME BIBLIOGRAPHY REF FOR: WHICH VERSION USED (OPTIM BOOK FOR ALL OF THEM) AND FOR THE HISTORY OF THESE (ORIGINAL PAPER + POTENTIALLY THE DIFFERENT PAPER WE USED)

5.1 Algorithms

5.1.1 Wolfe line search

The LBFGS and Conjugate gradient algorithms that we will analyse works by generating a sequence $x_k \in \mathbb{R}$ such that the sequence of objective evaluation at x_k, $f(x_k)$ this formula $x_{k+1} = x_k + \alpha p_k$. Where p_k is the descent direction that is computed by the algorithm.

However, the step size α needs to be chosen carfully to guaranty a sufficient decrease of the objective value.

1. **Sufficient Decrease (Armijo) Condition:**

$$f(\mathbf{x}_k + \alpha_k \mathbf{d}_k) \leq f(\mathbf{x}_k) + c_1 \alpha_k \nabla f(\mathbf{x}_k)^T \mathbf{d}_k$$

   where $0 < c_1 < 1$.

2. **Curvature Condition:**

$$\nabla f(\mathbf{x}_k + \alpha_k \mathbf{d}_k)^T \mathbf{d}_k \geq c_2 \nabla f(\mathbf{x}_k)^T \mathbf{d}_k$$

   where $c_1 < c_2 < 1$.

5.1.2 L-BFGS

"L-BFGS" stands for "Low-memory, Broyden–Fletcher–Goldfarb–Shanno". This method is a so-called "Quasi-Newton" method. Newton's methods work by generating a sequence x_k that converges to a local minima of the objective function. At each step, the objective function is approximated by a quadratic form which can be minimized formally. If $g : \mathbb{R}^d \mapsto \mathbb{R}$ is a quadratic function, and the current point in the sequence is $x_{curr} \in \mathbb{R}^d$, then the minimum of g is at

$$x_{\min} = -(\nabla^2 f(x_{curr}))^{-1} \times \nabla g(x_{curr})$$

. Thus,

$$x_{k+1} = x_k - H_f(x_k)^{-1} \nabla f(x_k)$$

.

However, Newton's methods involve the computation of the Hessian of the objective function $H(x_k)$ which becomes expensive for high dimensional problems, the BFGS method mitigates this issue by using an approximation of the Hessian B_k which is faster to compute.

$$B_{k+1} = B_k + \frac{y_k y_k^T}{y_k^T s_k} - \frac{B_k s_k s_k^T B_k}{s_k^T B_k s_k}$$

While faster, the BFGS method still store $d \times d$ values which can get expensive for high dimensional problems. If the mesh of the domain has $n \in \mathbb{N}$ vertices, then the size of the Hessian has $n \times n$ terms which consume a large amount of memory in the case of density estimation where the domains can have thousands of vertices. The L-BFGS mitigates this issue by approximating only the term $B_f(x_k)^{-1} \nabla f(x_k)$ using two loops. The full algorithm can be found at [4].

5.1.3 Nelder-Mead

The Nelder-Mead algorithm is a gradient-free optimization method. It works by keeping track of a simplex (a polygon of $N + 1$ vertices in N dimensions, the simplest possible polygon for a given dimension). At each iteration, the vertex corresponding to the worst objective function value gets moved according to some rules 2 using 4 parameters $(\alpha, \beta, \gamma, \delta)$, such that the measure of the simplex gets smaller. The stopping criterion used is std $< \epsilon$. Where std is the standard

deviation of the objective value on the vertices. One important aspect of this algorithm is the choise for the initial simplex, given an initial point x_0.

This method was first described in Nelder and Mead's original article [1], but it has multiple well-known issues which can affect its performance. [8] raises the issue of the sensitivity of the method to the choice of the initial simplex and compares multiple formula for the initial simplex. Using these results, we decide to use the initial simplex described in [6]. Additionally, the method does not scale well with the dimentionnality of the problem. [6] suggest adapting the parameters ($\alpha = 1, \beta = 2, \gamma = \frac{1}{2}, \delta = \frac{1}{2}$) used to compute the next simplex, to ($\alpha = 1, \beta = 1 + \frac{2}{N}, \gamma = 0.75 - \frac{1}{2N}, \delta = 1 - \frac{1}{N}$).

For the stopping criterion, we used $\sqrt{v} \leq \epsilon$ where v is the empirical variance of the objective values at each vertex.

5.1.4 Genetic algorithms

THE INITIAL METHOD + OUR MODIFICATION + PROBLEM ENCOUNTERED
INSERT INTRODUCTION OF GENETIC ALGORITHM
INSERT GENETIC OPERATORS
INSERT THE GENETIC OPERATORS WE USE
INSERT NICE GRAPH
INSERT SPECIAL VIGNETTE IN APPENDIX FOR TESTING
[11]

5.1.5 Conjugate gradient

HISTORY OF THE METHOD
INSERT MATHEMATICS OF THE METHOD
INSERT ALGORITHM SEE APPENDIX
INSERT MULTIPLE FORULAS PRP, FR, PR
INSERT SOME EXAMPLE OF EXPLOSIONS IN ONE FIGURE AND ONE TABLE (SMALL)
INSERT THE SPECIAL WOLFE LINE SEARCH WHEN FINISHED

5.2 Performance analysis

In this section, we compare the performance of LBFGS 5.1.2 with a memory size of 30, Nelder-Mead 5.1.3 and Conjugate gradient 5.1.5 against the implementation from the package SciPy [9]. We use multiple challenging functions that feature non-convexity, multiple local minima, and far-away global minima. This way, we can more precisly find the strenghts and weaknesses of each method.

We used the standard benchmark functions Rosenbrock, and Sphere functions. Additionally, as suggested in [5], we added Rastrigin Schwefel and SchafferF6 functions. The formula and features of each functions are shown in 2, and their graph in 2D can are shown in 10. The minimum value of each of these functions is 0.

We tested each pair of (optimizer method, objective function) on 30 initial points, sampled from a gaussian distribution around $\overrightarrow{0}$ (except for rosenbrock, the distribution was centerd around the point $[-1.2, 1]$, a usual starting point), the initial points are shown in 11.

Name	Formula	Argmin	Features
Sphere	$f(x) = \sum_{i=1}^{n} x_i^2$	$\overrightarrow{0}$	Simple convex function.
Rosenbrock	$f(x, y) = (1 - x)^2 + 100(y - x^2)^2$	$\begin{pmatrix} 1 \\ 1 \end{pmatrix}$	Non-convex.
Rastrigin	$f(x) = An + \sum_{i=1}^{n}[x_i^2 - A\cos(2\pi x_i)]$	$\overrightarrow{0}$	Non-convex, many local minima.
Schwefel	$f(x) = 418.9829n - \sum_{i=1}^{n} x_i \sin(\sqrt{\lvert x_i \rvert})$	$\begin{pmatrix} 418.9829 \\ \vdots \\ 418.9829 \end{pmatrix}$	Non-convex, many local minima. The global minimum is distant from other local minima.
Schaffer F6	$f(x, y) = 0.5 + \frac{\sin^2(\sqrt{x^2+y^2}) - 0.5}{(1+0.001(x^2+y^2))^2}$	$\overrightarrow{0}$	Non-convex, many local minima. The algorithms must climb and descent many peaks to attain global minima at 0

Figure 2: List of benchmark functions

The parameters used for the optimization process are the folowing:

- Tolerance $\epsilon = 1 \times 10^{-5}$
- Maximum number of iterations: 500
- Initial step size $\times 10^{-2}$

TODO USE SAME VAR NAME AS THE WOLFE LINE SEARCH PART.Additionnaly, together with the LBFGS and Conjugate Gradient methods, we used the Wolfe line search algorithm with $\alpha = 10^{-4}$, $\beta = 0.9$.

5.2.1 LBFGS30

PLOTS, ALG SPECIFIC PARAMETERS, REF TO APPENDIX TABLE AND CONCLUSION

5.2.2 Nelder-Mead

PLOTS, ALG SPECIFIC PARAMETERS, REF TO APPENDIX TABLE AND CONCLUSION

5.2.3 Conjugate gradient

PLOTS, ALG SPECIFIC PARAMETERS, REF TO APPENDIX TABLE AND CONCLUSION

5.2.4 Genetic algorithms

PLOTS, ALG SPECIFIC PARAMETERS, REF TO APPENDIX TABLE AND CONCLUSION

5.3 Conclusion

We observe that the compute time of the `fdaPDE` implementation is significantly lower than the `scipy` one, as expected by the difference in programing langages used. Moreover, the benchmark show performance on part with current state-of-the-art implementation of each algorithms.

DEVELOP THIS CONCLUSION

6 Simulations

EXPLAIN THE METHODOLOGY: MINIMIZATION OF THE CV ERROR, COMP WITH THE REAL DENSITY

6.1 Gaussian distribution on a square in 2D

WHAT IS THE DOMAIN
- WHAT IS THE DISTRIBUTION
- WHAT ARE THE PARAMETERS USED
- RESULTS IN NICE FIGURES FOR SPACE ONLY (OUR RES AND CRAN)
- RESULTS IN NICE FIGURES FOR SPACE AND TIME (OUR RES AND CRAN)
- INSERT TABLE WITH —REAL - APPROX—
- CONCLUSION

6.2 Decreasing distribution on a Horseshoe-shaped domain

WHAT IS THE DOMAIN
- WHAT IS THE DISTRIBUTION
- WHAT ARE THE PARAMETERS USED
- RESULTS IN NICE FIGURES FOR SPACE ONLY (OUR RES AND CRAN)
- RESULTS IN NICE FIGURES FOR SPACE AND TIME (OUR RES AND CRAN)
- INSERT TABLE WITH —REAL - APPROX—
- CONCLUSION

6.3 Kent distribution in a sphere embedded in 3D

WHAT IS THE DOMAIN
- WHAT IS THE DISTRIBUTION
- WHAT ARE THE PARAMETERS USED
- RESULTS IN NICE FIGURES FOR SPACE ONLY (OUR RES AND CRAN)
- RESULTS IN NICE FIGURES FOR SPACE AND TIME (OUR RES AND CRAN)
- INSERT TABLE WITH —REAL - APPROX—
- CONCLUSION

7 Case study

DO THE SAME

7.1 Distribution of infections on a 2D map

WHAT IS THE DOMAIN
- WHAT IS THE DATA
- WHAT ARE THE PARAMETERS USED
- RESULTS IN NICE FIGURES FOR SPACE ONLY (OUR RES AND CRAN)
- RESULTS IN NICE FIGURES FOR SPACE AND TIME (OUR RES AND CRAN)
- COMPARE THE RESULTS OF CRAN AND C++

7.2 Distribution of earthquakes

WHAT IS THE DOMAIN
- WHAT IS THE DATA
- WHAT ARE THE PARAMETERS USED
- RESULTS IN NICE FIGURES FOR SPACE ONLY (OUR RES AND CRAN)
- RESULTS IN NICE FIGURES FOR SPACE AND TIME (OUR RES AND CRAN)
- COMPARE THE RESULTS OF CRAN AND C++

7.3 Accidents on a linear network

WHAT IS THE DOMAIN
- WHAT IS THE DATA
- WHAT ARE THE PARAMETERS USED
- RESULTS IN NICE FIGURES FOR SPACE ONLY (OUR RES AND CRAN)
- RESULTS IN NICE FIGURES FOR SPACE AND TIME (OUR RES AND CRAN)
- COMPARE THE RESULTS OF CRAN AND C++

- tolerance: 1×10^{-5}
- step proposal: 1×10^{-2}
- max iteration count: 500

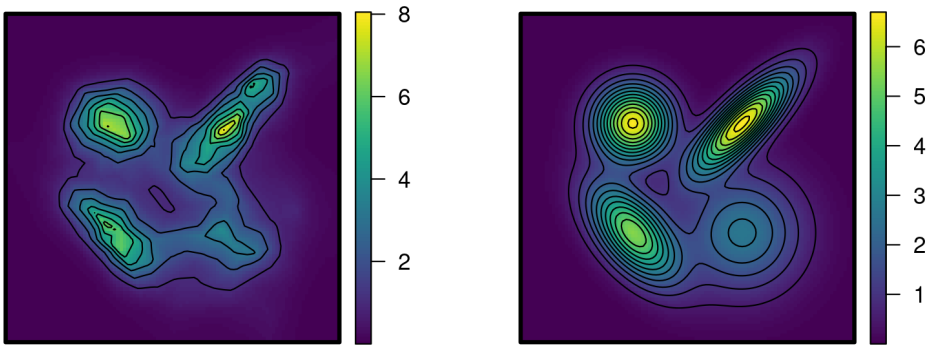


Figure 3: Density generated by L-BFGS30 (left) VS real density (right)

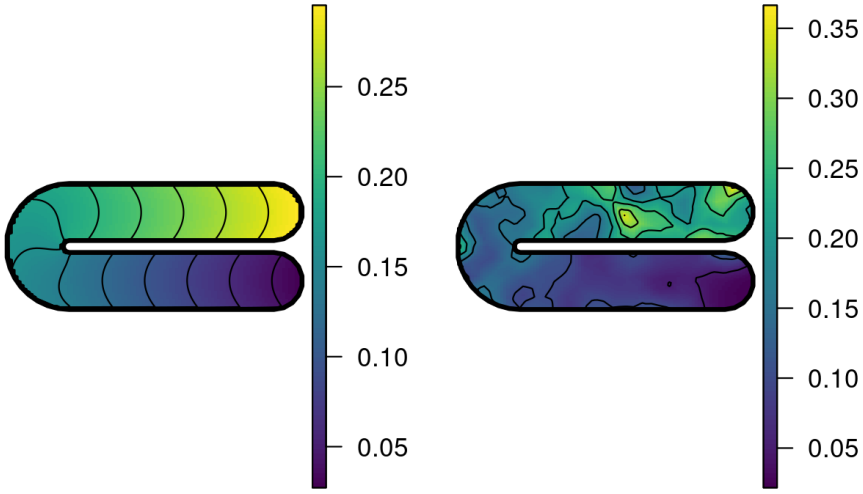


Figure 4: Density generated by L-BFGS30 (left) VS real density (right)

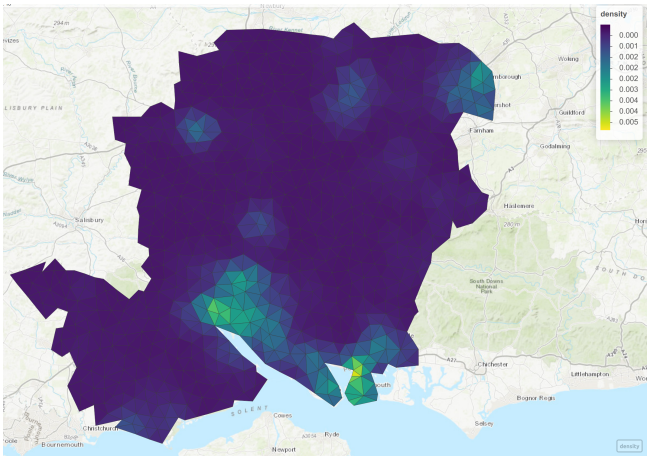


Figure 5: Density generated by L-BFGS30 (left) VS real density (right)

- cv size: 5

INSERT TABLE OF CPP VS R CV ERROR INSERT REAL DISTRIBUTION VS COMPUTED DISTRIBUTION INSERT PLOT OF DENSITIES OVER DOMAIN

8 Conclusion

Lorem ipsum dolor sit ame... [4, 11, 8, 1, 13, 5, 6, 10, 7, 2, 3]

References

[1] J. A. Nelder and R. Mead. "A Simplex Method for Function Minimization". In: *The Computer Journal* 7.4 (Jan. 1965), pp. 308–313. ISSN: 0010-4620. DOI: 10.1093/comjnl/7.4.308. eprint: https://academic.oup.com/comjnl/article-pdf/7/4/308/1013182/7-4-308.pdf. URL: https://doi.org/10.1093/comjnl/7.4.308.

[2] Jonathan R Shewchuk. *An Introduction to the Conjugate Gradient Method Without the Agonizing Pain*. Tech. rep. USA, 1994.

[3] M. Bessaou and P. Siarry. "A genetic algorithm with real-value coding to optimize multimodal continuous functions". In: *Structural and Multidisciplinary Optimization* 23.1 (Dec. 2001), pp. 63–74. ISSN: 1615-1488. DOI: 10.1007/s00158-001-0166-y. URL: https://doi.org/10.1007/s00158-001-0166-y.

[4] Wright Nocedal. *Numerical Optimization*. Springer Series in Operations Research and Financial Engineering. Springer New York, NY, 2006. ISBN: 978-0-387-30303-1.

[5] Johannes M. Dieterich and Bernd Hartke. *Empirical review of standard benchmark functions using evolutionary global optimization*. 2012. arXiv: 1207.4318 [cs.NE]. URL: https://arxiv.org/abs/1207.4318.

[6] Fuchang Gao and Lixing Han. "Implementing the Nelder-Mead simplex algorithm with adaptive parameters". In: *Computational Optimization and Applications* 51 (May 2012), pp. 259–277. DOI: 10.1007/s10589-010-9329-3.

[7] Iztok Fajfar, Árpád Bűrmen, and Janez Puhan. "The Nelder–Mead simplex algorithm with perturbed centroid for high-dimensional function optimization". In: *Optimization Letters* 13.5 (July 2019), pp. 1011–1025. ISSN: 1862-4480. DOI: 10.1007/s11590-018-1306-2. URL: https://doi.org/10.1007/s11590-018-1306-2.

[8] Simon Wessing. "Proper initialization is crucial for the Nelder–Mead simplex search". In: *Optimization Letters* 13.4 (June 2019), pp. 847–856. ISSN: 1862-4480. DOI: 10.1007/s11590-018-1284-4. URL: https://doi.org/10.1007/s11590-018-1284-4.

[9] Pauli Virtanen et al. "SciPy 1.0: Fundamental Algorithms for Scientific Computing in Python". In: *Nature Methods* 17 (2020), pp. 261–272. DOI: 10.1038/s41592-019-0686-2.

[10] Federico Ferraccioli et al. "Nonparametric Density Estimation Over Complicated Domains". In: *Journal of the Royal Statistical Society Series B: Statistical Methodology* 83.2 (Feb. 2021), pp. 346–368. ISSN: 1369-7412. DOI: 10.1111/rssb.12415. eprint: https://academic.oup.com/jrsssb/article-pdf/83/2/346/49321224/jrsssb_83_2_346.pdf. URL: https://doi.org/10.1111/rssb.12415.

[11] Sourabh Katoch, Sumit Singh Chauhan, and Vijay Kumar. "A review on genetic algorithm: past, present, and future". In: *Multimedia Tools and Applications* 80.5 (Feb. 2021), pp. 8091–8126. ISSN: 1573-7721. DOI: 10.1007/s11042-020-10139-6. URL: https://doi.org/10.1007/s11042-020-10139-6.

[12] Eleonora Arnone et al. "A roughness penalty approach to estimate densities over two-dimensional manifolds". In: *Computational Statistics & Data Analysis* 174 (2022), p. 107527. ISSN: 0167-9473. DOI: https://doi.org/10.1016/j.csda.2022.107527. URL: https://www.sciencedirect.com/science/article/pii/S0167947322001074.

[13] Blerta Begu et al. “A nonparametric penalized likelihood approach to density estimation of space–time point patterns”. In: *Spatial Statistics* 61 (2024), p. 100824. ISSN: 2211-6753. DOI: https://doi.org/10.1016/j.spasta.2024.100824. URL: https://www.sciencedirect.com/science/article/pii/S2211675324000150.

[14] Simone Panzeri et al. “Spatio-temporal intensity estimation for inhomogeneous Poisson point processes on linear networks: A roughness penalty method”. In: *Spatial Statistics* 69 (2025), p. 100912. ISSN: 2211-6753. DOI: https://doi.org/10.1016/j.spasta.2025.100912. URL: https://www.sciencedirect.com/science/article/pii/S221167532500034X.

A Benchmark tables

Method	Iters	x_diff	f_diff	duration
sphere_2d (fdaPDE)	16.0 ±2.4	2.59e-03 ±1.87e-03	1.01e-05±1.56e-05	2ns±4ns
sphere_2d (scipy)	50.1 ±3.8	3.32e-06 ±1.05e-06	1.21e-11±8.12e-12	846µs±223µs
sphere_10d (fdaPDE)	175.2 ±31.2	6.08e-03 ±8.91e-04	3.77e-05±1.16e-05	34ns±13ns
sphere_10d (scipy)	500.0 ±0.0	5.94e-01 ±3.29e-01	4.57e-01±5.40e-01	6ms±157µs
sphere_30d (fdaPDE)	500.0 ±0.0	1.46e-01 ±1.07e-01	3.24e-02±7.05e-02	253ns±57ns
sphere_30d (scipy)	500.0 ±0.0	3.71e+00 ±7.61e-01	1.43e+01±5.82e+00	6ms±154µs
rastrigin_2d (fdaPDE)	41.6 ±9.1	1.44e+00 ±8.85e-01	2.85e+00±3.57e+00	3ns±0ns
rastrigin_2d (scipy)	39.1 ±3.9	2.92e+00 ±1.14e+00	9.82e+00±7.24e+00	716µs±56µs
rastrigin_10d (fdaPDE)	496.5 ±9.2	4.10e+00 ±9.41e-01	1.83e+01±8.30e+00	135ns±28ns
rastrigin_10d (scipy)	500.0 ±0.0	6.43e+00 ±1.19e+00	4.67e+01±1.73e+01	11ms±332µs
rastrigin_30d (fdaPDE)	500.0 ±0.0	8.84e+00 ±1.43e+00	1.27e+02±3.31e+01	819ns±186ns
rastrigin_30d (scipy)	500.0 ±0.0	1.07e+01 ±1.43e+00	2.24e+02±3.76e+01	19ms±464µs
schwefel_2d (fdaPDE)	41.7 ±6.3	5.77e+02 ±1.10e+02	6.70e+02±9.43e+01	4ns±0ns
schwefel_2d (scipy)	55.1 ±5.4	6.10e+02 ±9.59e+01	6.65e+02±7.42e+01	1ms±121µs
schwefel_10d (fdaPDE)	479.4 ±19.2	1.39e+03 ±1.14e+02	3.31e+03±2.58e+02	140ns±32ns
schwefel_10d (scipy)	500.0 ±0.0	1.38e+03 ±1.11e+02	3.33e+03±2.26e+02	13ms±576µs
schwefel_30d (fdaPDE)	500.0 ±0.0	2.35e+03 ±9.50e+01	1.05e+04±3.68e+02	557ns±106ns
schwefel_30d (scipy)	500.0 ±0.0	2.38e+03 ±1.12e+02	1.08e+04±4.60e+02	23ms±480µs
rosenbrock (fdaPDE)	76.3 ±27.0	3.71e-03 ±3.65e-03	1.37e-05±2.26e-05	5ns±2ns
rosenbrock (scipy)	105.8 ±10.9	2.09e-06 ±1.12e-06	6.48e-12±1.05e-11	1ms±176µs
schaffer_f6 (fdaPDE)	13.8 ±2.6	6.91e+00 ±4.62e+00	5.54e-02±5.79e-02	2ns±0ns
schaffer_f6 (scipy)	53.9 ±4.0	6.49e+00 ±3.59e+00	4.74e-02±4.18e-02	954µs±95µs

Table 1: Nelder-Mead method, scipy vs fdaPDE

Method	Iters	x_diff	f_diff	duration
sphere_2d (fdaPDE)	1.0 ±0.0	5.06e-14 ±7.68e-14	1.91e+00±1.93e+00	0ns±0ns
sphere_2d (scipy)	1.9 ±0.3	1.29e-08 ±1.90e-08	5.16e-16±1.70e-15	388µs±233µs
sphere_10d (fdaPDE)	1.0 ±0.0	6.60e-13 ±3.26e-13	1.01e+01±4.21e+00	2ns±0ns
sphere_10d (scipy)	2.0 ±0.0	3.64e-07 ±2.90e-07	2.13e-13±3.20e-13	425µs±45µs
sphere_30d (fdaPDE)	1.0 ±0.0	3.93e-12 ±1.95e-12	3.02e+01±8.94e+00	5ns±0ns
sphere_30d (scipy)	2.0 ±0.0	4.14e-06 ±2.41e-06	2.27e-11±2.40e-11	716µs±15µs
rastrigin_2d (fdaPDE)	17.6 ±58.7	2.72e+00 ±8.31e-01	8.09e+00±4.49e+00	61ns±274ns
rastrigin_2d (scipy)	5.5 ±1.5	2.70e+00 ±8.82e-01	8.09e+00±4.75e+00	836µs±237µs
rastrigin_10d (fdaPDE)	23.8 ±37.3	6.55e+00 ±1.51e+00	4.53e+01±2.06e+01	434ns±1µs
rastrigin_10d (scipy)	7.5 ±2.2	6.37e+00 ±1.19e+00	4.21e+01±1.51e+01	1ms±514µs
rastrigin_30d (fdaPDE)	19.1 ±22.2	1.10e+01 ±1.90e+00	1.24e+02±4.46e+01	1µs±3µs
rastrigin_30d (scipy)	9.8 ±1.7	1.06e+01 ±1.43e+00	1.16e+02±3.05e+01	10ms±2ms
schwefel_2d (fdaPDE)	7.1 ±1.7	6.08e+02 ±1.05e+02	6.50e+02±8.36e+01	5ns±2ns
schwefel_2d (scipy)	5.5 ±1.7	6.12e+02 ±1.19e+02	6.46e+02±8.51e+01	840µs±291µs
schwefel_10d (fdaPDE)	12.4 ±2.7	1.38e+03 ±1.12e+02	3.29e+03±2.90e+02	74ns±18ns
schwefel_10d (scipy)	9.1 ±3.3	1.38e+03 ±1.07e+02	3.31e+03±2.26e+02	3ms±1ms
schwefel_30d (fdaPDE)	14.9 ±2.4	2.38e+03 ±1.13e+02	1.02e+04±3.69e+02	854ns±235ns
schwefel_30d (scipy)	9.5 ±1.7	2.38e+03 ±1.11e+02	1.02e+04±3.65e+02	16ms±2ms
rosenbrock (fdaPDE)	196.5 ±215.3	4.39e-04 ±1.90e-05	4.04e-08±4.88e-09	426ns±515ns
rosenbrock (scipy)	33.8 ±3.6	4.61e-04 ±4.21e-04	1.31e-07±1.70e-07	3ms±378µs
schaffer_f6 (fdaPDE)	3.9 ±0.5	6.49e+00 ±3.59e+00	4.74e-02±4.18e-02	2ns±1ns
schaffer_f6 (scipy)	3.5 ±0.6	6.70e+00 ±3.57e+00	4.94e-02±4.36e-02	507µs±98µs

Table 2: LBFGS30

Method	Iters	x_diff	f_diff	duration
sphere_2d	136.9 ±13.9	1.79e-03 ±1.02e-03	4.19e-06±4.28e-06	499ns±97ns
sphere_10d	178.1 ±5.8	4.02e-03 ±8.47e-04	1.69e-05±7.29e-06	2µs±241ns
sphere_30d	276.9 ±21.9	4.75e-01 ±3.83e-01	3.68e-01±5.62e-01	9µs±959ns
rastrigin_2d	221.4 ±13.7	4.72e-01 ±5.61e-01	5.31e-01±6.78e-01	1µs±134ns
rastrigin_10d	260.7 ±5.6	2.74e+00 ±6.32e-01	7.93e+00±3.75e+00	5µs±350ns
rastrigin_30d	353.3 ±25.9	1.03e+01 ±1.83e+00	1.11e+02±4.02e+01	15µs±2µs
schwefel_2d	178.6 ±12.2	4.59e+02 ±3.32e+02	2.02e+02±1.78e+02	967ns±94ns
schwefel_10d	216.8 ±5.5	1.40e+03 ±2.26e+02	1.95e+03±3.95e+02	3µs±638ns
schwefel_30d	338.0 ±26.3	2.60e+03 ±2.12e+02	6.80e+03±6.54e+02	18µs±2µs
rosenbrock	253.9 ±14.3	1.04e+00 ±3.85e+00	1.10e+00±5.83e+00	966ns±125ns
schaffer_f6	140.5 ±10.7	5.96e+00 ±4.76e+00	4.21e-02±6.39e-02	723ns±124ns

Table 3: Genetic algorithm, Binary tournament selectioon, gaussian mutation

Method	Iters	x_diff	f_diff	duration
sphere_2d	139.8 ±12.2	1.90e-03 ±1.23e-03	5.07e-06±6.09e-06	414ns±73ns
sphere_10d	183.5 ±5.7	3.99e-03 ±9.66e-04	1.68e-05±8.45e-06	1µs±318ns
sphere_30d	359.6 ±8.4	2.37e+01 ±6.27e+00	6.00e+02±3.07e+02	7µs±1µs
rastrigin_2d	221.9 ±13.3	7.46e-01 ±5.59e-01	8.62e-01±7.27e-01	856ns±153ns
rastrigin_10d	264.7 ±5.5	3.51e+00 ±8.11e-01	1.30e+01±5.88e+00	3µs±550ns
rastrigin_30d	467.4 ±10.3	1.23e+02 ±3.67e+01	1.65e+05±9.23e+04	14µs±1µs
schwefel_2d	179.8 ±13.3	4.42e+02 ±3.29e+02	1.67e+02±1.63e+02	701ns±158ns
schwefel_10d	220.3 ±6.9	1.36e+03 ±2.19e+02	1.61e+03±3.18e+02	3µs±506ns
schwefel_30d	392.5 ±8.6	2.70e+03 ±2.59e+02	5.99e+03±6.28e+02	16µs±1µs
rosenbrock	227.5 ±19.0	1.25e-01 ±8.72e-02	4.97e-03±5.79e-03	683ns±115ns
schaffer_f6	141.8 ±12.9	6.59e+00 ±4.69e+00	4.84e-02±6.37e-02	647ns±98ns

Table 4: Genetic algorithm, Binary tournament selectioon, crossover mutation and gaussian mutation

Method	Iters	x_diff	f_diff	duration
sphere_2d	140.3 ±11.9	1.71e-03 ±8.99e-04	3.71e-06±3.53e-06	883ns±202ns
sphere_10d	182.1 ±5.6	3.72e-03 ±8.24e-04	1.45e-05±6.52e-06	2µs±377ns
sphere_30d	327.1 ±11.1	4.67e+00 ±2.55e+00	2.81e+01±3.12e+01	8µs±2µs
rastrigin_2d	223.7 ±12.6	4.86e-01 ±5.79e-01	5.64e-01±7.24e-01	1µs±374ns
rastrigin_10d	262.9 ±6.5	2.98e+00 ±5.40e-01	9.22e+00±2.87e+00	4µs±1µs
rastrigin_30d	431.9 ±18.5	2.88e+01 ±1.73e+01	9.36e+03±1.74e+04	13µs±3µs
schwefel_2d	178.2 ±14.1	4.89e+02 ±2.95e+02	3.07e+02±1.79e+02	678ns±88ns
schwefel_10d	220.1 ±6.0	1.47e+03 ±1.85e+02	2.36e+03±3.54e+02	2µs±281ns
schwefel_30d	379.5 ±9.2	2.52e+03 ±1.79e+02	7.49e+03±5.15e+02	18µs±3µs
rosenbrock	273.1 ±9.8	8.89e-01 ±1.02e+00	2.27e-01±4.75e-01	1µs±246ns
schaffer_f6	99.9 ±19.9	1.33e+02 ±9.60e+01	4.57e-01±9.38e-02	494ns±151ns

Table 5: Genetic algorithm, Rank selection, gaussian mutation

Method	Iters	x_diff	f_diff	duration
sphere_2d	140.8 ±14.5	1.85e-03 ±9.41e-04	4.26e-06±3.72e-06	790ns±206ns
sphere_10d	252.5 ±35.2	2.97e-01 ±4.39e-01	2.75e-01±6.43e-01	3µs±1µs
sphere_30d	371.6 ±6.3	4.85e+01 ±8.78e+00	2.43e+03±9.09e+02	9µs±1µs
rastrigin_2d	226.1 ±12.4	8.12e-01 ±5.23e-01	9.29e-01±6.88e-01	1µs±178ns
rastrigin_10d	326.5 ±35.5	5.23e+00 ±2.09e+00	7.08e+01±2.47e+02	5µs±879ns
rastrigin_30d	481.5 ±8.7	2.53e+02 ±5.50e+01	6.72e+05±2.74e+05	10µs±2µs
schwefel_2d	180.8 ±12.9	4.43e+02 ±3.52e+02	1.61e+02±1.50e+02	1µs±246ns
schwefel_10d	294.3 ±37.0	1.41e+03 ±2.27e+02	1.62e+03±3.24e+02	4µs±970ns
schwefel_30d	405.9 ±7.8	2.70e+03 ±2.83e+02	9.81e+03±4.84e+03	17µs±1µs
rosenbrock	255.5 ±16.0	4.02e-01 ±1.48e-01	4.46e-02±2.91e-02	1µs±318ns
schaffer_f6	84.2 ±29.8	2.35e+02 ±1.87e+02	4.71e-01±7.09e-02	530ns±190ns

Table 6: Genetic algorithm, Rank selectioon, crossover mutation and gaussian mutation

B Algorithms

Algorithm 2 Nelder-Mead Algorithm

Require: Objective function f, initial simplex vertices $\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_{n+1}$, convergence tolerance ϵ

Ensure: Simplex vertex $\mathbf{x}_0$ that minimizes f

Order the vertices such that $f(\mathbf{x}_1) \leq f(\mathbf{x}_2) \leq \cdots \leq f(\mathbf{x}_{n+1})$
repeat
    Compute the centroid $\mathbf{x}_0 = \frac{1}{n} \sum_{i=1}^{n} \mathbf{x}_i$
    Compute the reflection point $\mathbf{x}_r = \mathbf{x}_0 + \alpha(\mathbf{x}_0 - \mathbf{x}_{n+1})$
    if $f(\mathbf{x}_1) \leq f(\mathbf{x}_r) < f(\mathbf{x}_n)$ **then**
        Replace $\mathbf{x}_{n+1}$ with $\mathbf{x}_r$
    else if $f(\mathbf{x}_r) < f(\mathbf{x}_1)$ **then**
        Compute the expansion point $\mathbf{x}_e = \mathbf{x}_0 + \gamma(\mathbf{x}_r - \mathbf{x}_0)$
        if $f(\mathbf{x}_e) < f(\mathbf{x}_r)$ **then**
            Replace $\mathbf{x}_{n+1}$ with $\mathbf{x}_e$
        else
            Replace $\mathbf{x}_{n+1}$ with $\mathbf{x}_r$
        end if
    else
        Compute the contraction point $\mathbf{x}_c = \mathbf{x}_0 + \rho(\mathbf{x}_{n+1} - \mathbf{x}_0)$
        if $f(\mathbf{x}_c) < f(\mathbf{x}_{n+1})$ **then**
            Replace $\mathbf{x}_{n+1}$ with $\mathbf{x}_c$
        else
            Shrink the simplex towards $\mathbf{x}_1$
        end if
    end if
    Order the vertices such that $f(\mathbf{x}_1) \leq f(\mathbf{x}_2) \leq \cdots \leq f(\mathbf{x}_{n+1})$
until $\frac{1}{n+1} \sum_{i=1}^{n+1} (f(\mathbf{x}_i) - f(\mathbf{x}_0))^2 < \epsilon$
return $\mathbf{x}_1$

C Figures

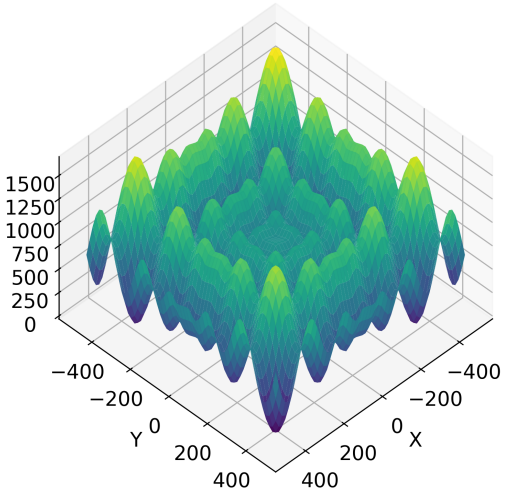


Figure 6: Schwefel

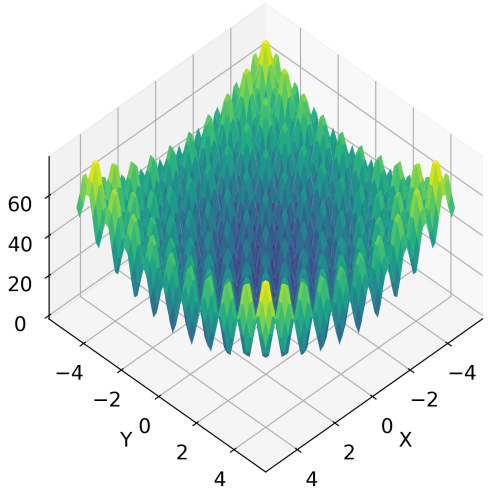


Figure 7: Rastrigin

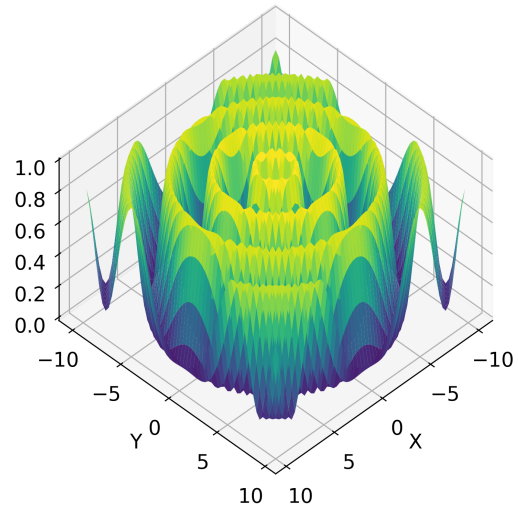


Figure 8: Schaffer F6

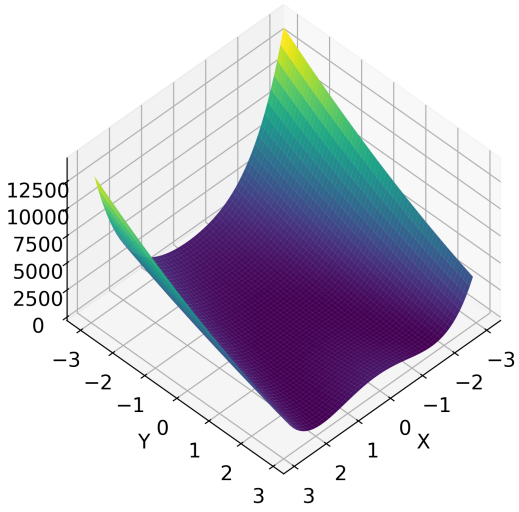
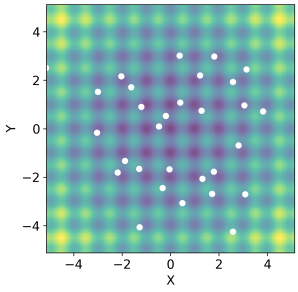
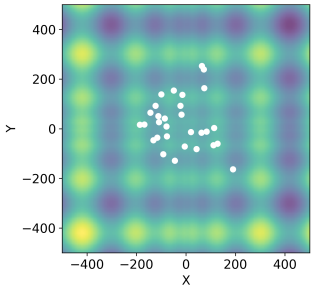


Figure 9: Rosenbrock

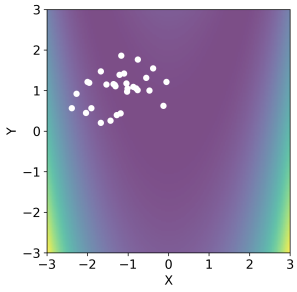
Figure 10: 2D graphs of the benchmark functions



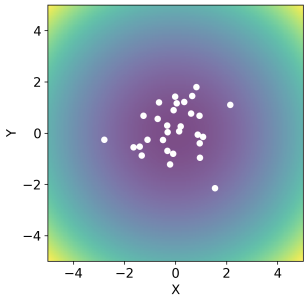
(a) Rastrigin



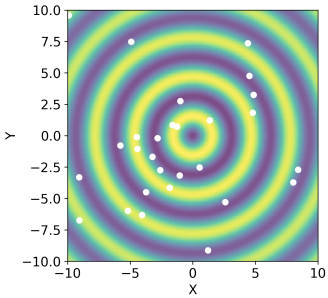
(b) Schwefel



(c) Rosenbrock

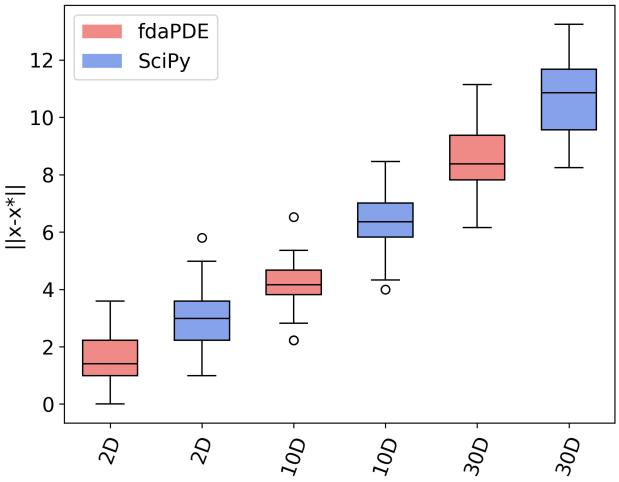


(d) Sphere function

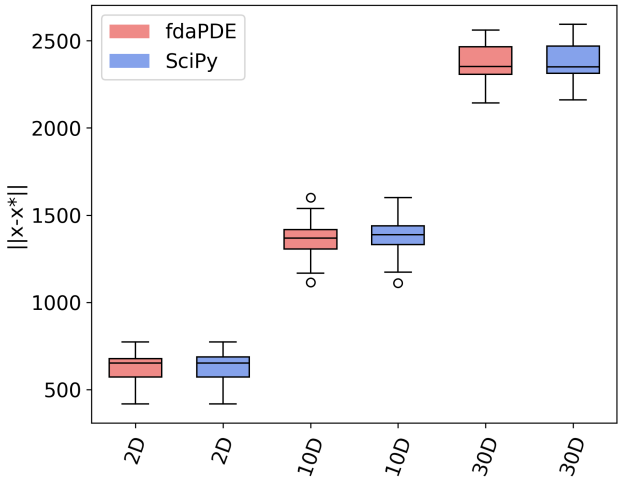


(e) Schaffer F6

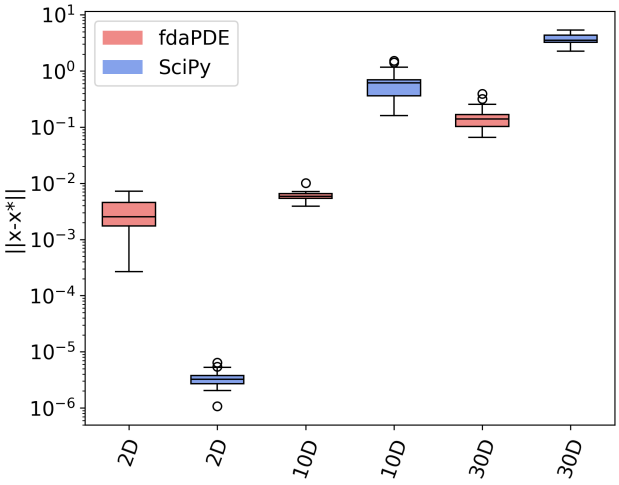
Figure 11: Distribution of the 30 initial points for every test function



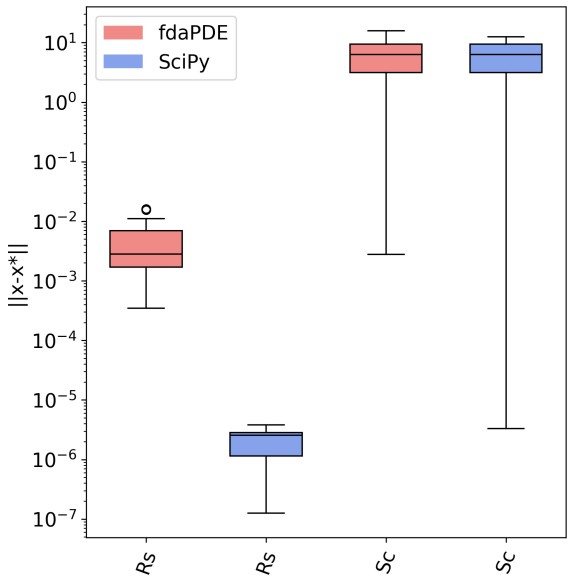
(a) Rastrigin



(b) Schwefel function

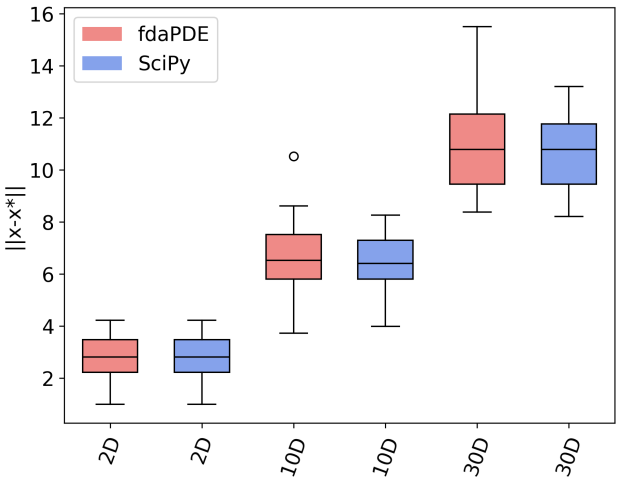


(c) Sphere function

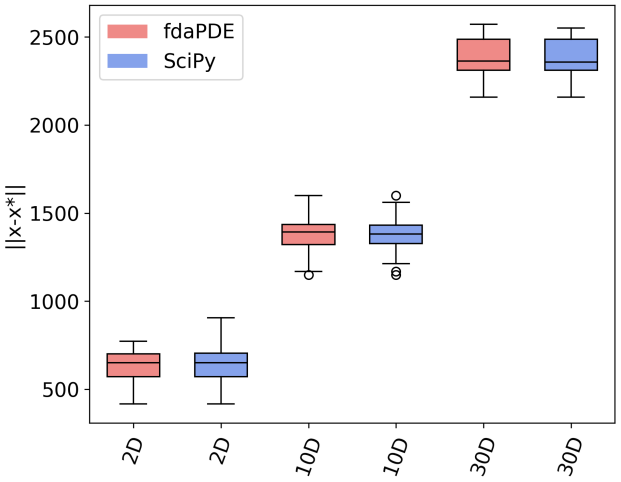


(d) Rosenbrock "Rs" (Left) Schaffer "Sc" (Right)

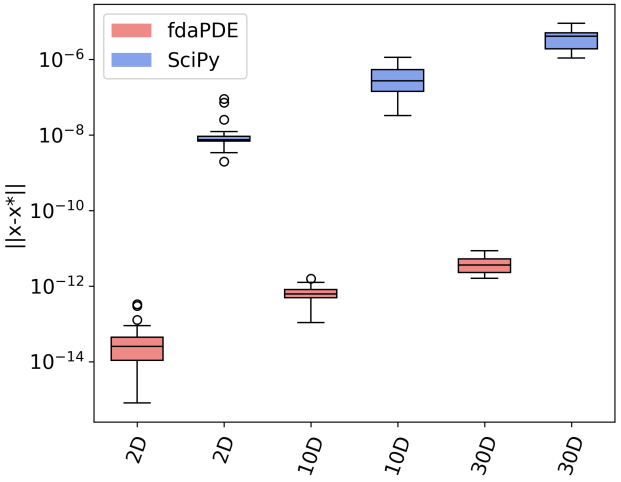
Figure 12: Convergence of the Nelder-Mead method, fdaPDE vs scipy



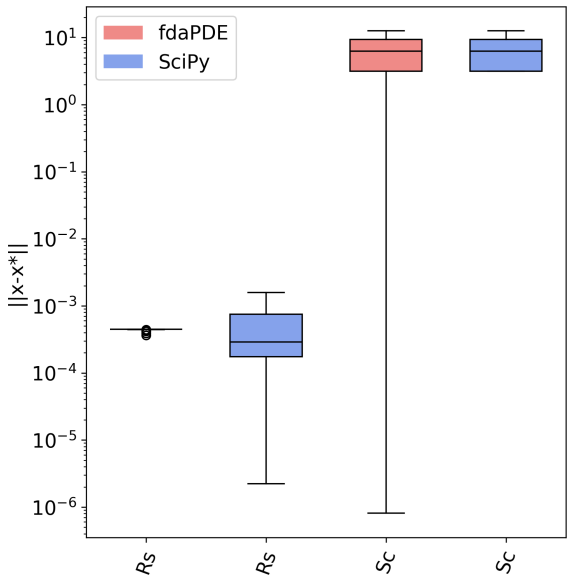
(a) Rastrigin



(b) Schwefel function



(c) Sphere function



(d) Rosenbrock "Rs" (Left) Schaffer "Sc" (Right)

Figure 13: Convergence of the LBFGS method with a memory size of 30, fdaPDE vs scipy