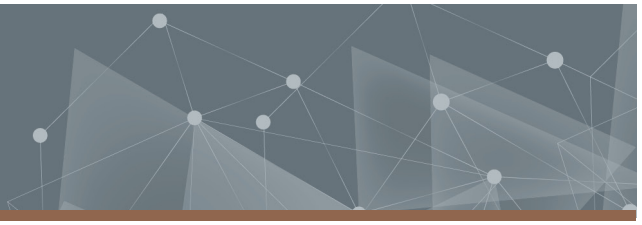
CHALMERS
UNIVERSITY OF TECHNOLOGY



Interacting particle systems for constrained optimization

Master's thesis in Physics

Jonathan Borsander

DEPARTMENT OF MATHEMATICAL SCIENCES

CHALMERS UNIVERSITY OF TECHNOLOGY
Gothenburg, Sweden 2025
www.chalmers.se

Master's thesis 2025

Interacting particle systems for constrained optimization

JONATHAN BORSANDER

CHALMERS
UNIVERSITY OF TECHNOLOGY

Department of Mathematical Sciences
Chalmers University of Technology
Gothenburg, Sweden 2025

Interacting particle systems for constrained optimization
JONATHAN BORSANDER

Supervisor: Akash Sharma, Department of Mathematical Sciences
Examiner: Axel Ringh, Department of Mathematical Sciences

Master's Thesis 2025
Department of Mathematical Sciences
Chalmers University of Technology
SE-412 96 Gothenburg
Telephone +46 31 772 1000

Cover: Python plot showing 1000 particle swarm optimization particles search and locate the global minimum of the Eggholder function.

Typeset in LaTeX
Printed by Chalmers Reproservice
Gothenburg, Sweden 2025

Interacting particle systems for constrained optimization
Jonathan Borsander
Department of Mathematical Sciences
Chalmers University of Technology

Abstract

This thesis investigates two approaches to constrained optimization: particle swarm optimization (PSO) and a second-order Kalman-Langevin method. Both techniques are formulated in continuous time as stochastic differential equations (SDEs) and are implemented using numerical discretization schemes. Constraints are enforced through a reflection mechanism applied at the domain boundaries. The performance of the PSO algorithm is first evaluated on the Cross-in tray and Eggholder benchmark functions, where it demonstrates reliable convergence to global minima despite the highly non-convex landscapes. The Kalman-Langevin method is primarily assessed on the Rastrigin function, exhibiting robust performance in the presence of numerous local minima.

To assess the applicability of these methods to practical problems, both algorithms are subsequently applied to the task of determining the optimal placement of a single gold atom on a gold surface, a problem characterized by a computationally expensive potential energy surface. The results indicate that both PSO and the Kalman-Langevin approach are effective in this setting, highlighting their generalizability beyond standard test functions. Furthermore, the parameter configurations identified during benchmark tuning are found to be transferable to the real-world application. These findings suggest that interacting particle systems governed by SDEs, such as PSO and Kalman-Langevin dynamics, constitute promising frameworks for addressing constrained optimization problems.

Keywords: Constrained optimization, Interacting particle systems, Particle swarm optimization, Kinetic Kalman-Langevin.

Acknowledgements

Finishing this thesis has been a journey full of learning, challenges, and plenty of support from the people around me. I definitely didn't do this alone. First, I want to thank my supervisor, Akash Sharma, for your guidance, patience, and for always being there when things got tricky.

I'm also grateful to my examiner, Axel Ringh, for your thoughtful feedback and for helping me navigate the complicated process of writing a master's thesis.

A big thank you to my family for all the love and support and to my friends for making sure I had some fun along the way.

Finally, I want to thank my university for providing a great environment to learn and grow during these years.

Jonathan Borsander, Gothenburg, June 2025

Contents

1

Introduction

The amount of data available in different contexts continues to grow, and in order to navigate and extract insights from this overwhelming amount of information, one must rely on mathematical models and computational techniques. In particular, methods for optimization and sampling are often used, and these methods have evolved significantly from simple linear regression for inference and gradient descent for optimization to more sophisticated approaches capable of handling complex scenarios. In this project, we focus on two key methods: particle swarm optimization with constraints and the kinetic Kalman-Langevin stochastic differential equation with constraints. These models are stochastic differential equations governing interacting particle systems. In general, methods based on interacting particle systems have found recent attraction [1, 2, 3, 4, 5, 6, 7, 8] due to improved capabilities in exploring the hyper-surface of objective functions.

Particle swarm optimization is a popular gradient-free optimization method proposed in [9]. An SDE-based formulation of particle swarm optimization is suggested by [10]. We adopt the same formulation and to incorporate constraints, we add a reflection term, following the method outlined in [11] (see also [12] for numerical implementation). In addition to SDE-driven particle swarm optimization, we also test the performance of kinetic Kalman-Langevin dynamics with constraints. Kinetic Kalman-Langevin dynamics has been investigated for sampling purposes in [3]. Here, we consider constrained version of kinetic Kalman Langevin dynamics for the purpose of constrained optimization. We test both methods on benchmark functions for constrained optimization, where we also tune hyperparameters to study their performance. Finally, we show how these approaches can be applied to a problem in material science, namely finding the optimal placement of a single Au-atom on top of an Au-surface via minimization of the potential energy.

1.1 Notation

In this section, notation and operations frequently used in the report will be explained. For $\mathbf{x}, \mathbf{y} \in \mathbb{R}^d$, $\langle x, y \rangle \in \mathbb{R}$ is the scalar product between the two vectors $\mathbf{x}$ and $\mathbf{y}$. For a matrix M the transpose will be written as $M^\top$. In this project,

the constraints we consider will be of the form that the decision variables need to belong to some set $G \subset \mathbb{R}^n$. To this end, let ∂G denote the boundary of G, defined as $\partial G = \bar{G} \backslash G$, and let $I_{\partial G}(X(t))$ denote the indicator function of the boundary, i.e. a function that is 1 on the boundary of the domain G and 0 otherwise. Time dependence will be written in two different formulations: continuous, and discrete. Continuous time dependence will be written with parenthesis as $f(t)$, $t \in [0, T]$, and discrete time formulation will be written with subscript $k = 0, ..., n$, where n is the total number of steps taken to get to the total runtime of T.

2

Background

This chapter introduces the most fundamental concepts used in the thesis. We begin with an overview of constrained optimization and common methods for enforcing constraints, such as projection and penalty methods. Next, we present stochastic differential equations, which describe our models in continuous time, and the Euler-Maruyama scheme, which is used to discretize these equations and express them in discrete time. Finally, we discuss the discretization of the reflection way of enforcing constraints within the numerical schemes.

2.1 Constrained optimization

In constrained optimization restrictions are imposed on the decision variables of an objective function, specifying the permissible values these variables can take. Given a set of variables $x_1, x_2, \ldots, x_n \in \mathbb{R}$, the goal is to minimize an objective function $f(\mathbf{x})$ subject to m constraints. Without loss of generality, any inequality constraint of the form $a \leq b$ can be rewritten as $a - b \leq 0$, so all constraints can be expressed as $g_i(x_1, x_2, \ldots, x_n) \leq 0$, for $i = 1, 2, \ldots, m$. Thus, a constrained optimization problem can be formulated as [13]:

$$\min_{x_1,x_2,\ldots,x_n} \quad f(x_1, x_2, ..., x_n), \quad \text{such that} \quad \begin{array}{l} g_1(x_1, x_2, ..., x_n) \leq 0 \\ g_2(x_1, x_2, ..., x_n) \leq 0 \\ \vdots \\ g_m(x_1, x_2, ..., x_n) \leq 0. \end{array} \tag{2.1}$$

We will also use the notation $\mathbf{g}$ to denote the vector-valued function whose components are the individual functions $g_i(\mathbf{x})$. With this notation, problem (2.1) can be written as $\min f(x)$ subject to $\mathbf{g}(\mathbf{x}) \leq 0$.

There are several different reasons why one might want to impose constraints on variables. It could be to match some real life limitation we know to be true, for example that energies must be non-negative, or it could be to adhere to some legal requirement. Some places where constrained optimization is commonplace is in:

- Engineering design (e.g., maximizing structural strength while limiting weight and cost)
- Finance (e.g., maximizing return with risk and budget constraints)
- Machine learning (e.g., ensuring probabilities sum to one)
- Resource allocation and logistics (e.g., distributing resources without exceeding capacity)

In this project, the constraint imposed on the objective function is a boundary constraint, where variables are restricted to remain within predefined lower and upper bounds. This constraint ensures that all particles consistently explore only the permitted area of the search space.

2.2 Common constraint enforcement methods

With the knowledge of constraints a natural question that arises is how to choose to enforce these constraints during optimization. There are several different ways to ensure that the particles stay within the allowed search region throughout the process. The most common ways of enforcing constraints for gradient descent based methods are projection and penalty methods.

Projection Methods

Since time is discretized when working with numerical solutions, it is possible for a particle to move outside the allowed search domain in a single time step. Projection methods check after each gradient descent step if the new point lies outside the search domain G. If the particle did in fact leave G, it is projected onto the closest point within the feasible region. This guarantees that all particles satisfy the constraints at every step. The projected gradient method can be written as [14]

$$\mathbf{x}_{k+1} = \mathrm{Proj}_G(\mathbf{x}_k - \eta \nabla f(\mathbf{x}_k)), \tag{2.2}$$

where the projection operator is defined by

$$\mathrm{Proj}_G(\mathbf{y}) = \arg\min_{\mathbf{z} \in G} \|\mathbf{z} - \mathbf{y}\|_2. \tag{2.3}$$

Projection is particularly effective for convex constraint sets, where the projection operation is well-defined and computationally efficient. However, for more complex feasible regions, the projection step can become challenging or computationally expensive.

Penalty Methods

Penalty methods transform the constrained optimization problem

$$\min_{\mathbf{x}\in G} f(\mathbf{x})$$

into an unconstrained problem by adding penalty terms for constraint violations:

$$\min_{\mathbf{x}} f(\mathbf{x}) + \lambda \sum_{i=1}^{m} P(g_i(\mathbf{x})) \tag{2.4}$$

where $P : \mathbb{R} \to \mathbb{R}^+$ is a penalty function and $\lambda > 0$ is a penalty parameter. A common choice is [15]:

$$P(g_i(\mathbf{x})) = \begin{cases} 0 & \text{if } g_i(\mathbf{x}) \leq 0 \\ g_i(\mathbf{x})^2 & \text{otherwise.} \end{cases} \tag{2.5}$$

2.3 SDE and Euler-Maruyama scheme

Both optimization methods considered in this thesis, Particle swarm optimization (PSO) with constraint and second order (kinetic) Kalman-Langevin with constraint, are formulated in continuous time as stochastic differential equations (SDEs). By taking inspiration from the Euler-Maruyama scheme, they can be expressed in discrete time formulation. In this thesis the process will be the following: formulate the dynamics as SDEs and then discretize them using this Euler-Maruyama scheme based process in order to obtain implementable numerical methods. Equations in continuous time formulation will be written as SDEs which have the general form

$$dX(t) = b(t, X(t))dt + \sigma(t, X(t))dw(t), \tag{2.6}$$

where b is the drift term, and σ is the diffusion coefficient. This notation by itself has no meaning and is used as shorthand notation for the meaningful integral formulation

$$X(t) = X_0 + \int_0^t b(s, X(s))ds + \int_0^t \sigma(s, X(s))dw(s), \tag{2.7}$$

for some position X at time t. For the purpose of implementing the methods numerically, these continuous-time SDEs are discretized using the Euler-Maruyama scheme. The discrete-time update rule takes the form

$$X_{k+1} = X_k + b(t_k, X_k)h + \sigma(t_k, X_k)\sqrt{h}\,\xi_{k+1}, \tag{2.8}$$

where k indexes the discrete time steps, h is the step-size, $t_{k+1} = t_k + h$, and ξ_{k+1} is a standard normal random variable, i.e., $\xi_{k+1} \sim \mathcal{N}(0_d, I_d)$, meaning a normal distribution centered around the origin with a standard deviation of 1 in each dimension.

The step-size h is determined by the total runtime T and the number of steps n via $h = T/n$, with $h \in (0, 1)$.

For methods involving both position and velocity, such as PSO and kinetic Kalman-Langevin dynamics, the state of each particle is described by both its position X and its velocity V. In these cases, the update for the position at each time step can be written as

$$X_{k+1} = X_k + hV_k, \tag{2.9}$$

where V_k represents the velocity of the particle at step k. The velocity itself is updated according to the specific dynamics of the method.

2.4 Discretization of the reflection enforcement method for rectangular constraint

In this project, the enforcement of boundary constraints is handled via reflection. Since both optimization methods are formulated in continuous time using SDEs, the reflection mechanism must also be discretized to be compatible with the Euler-Maruyama scheme.

At each discrete time step in the Euler-Maruyama scheme, the position of a particle is updated according to equation (2.8). After such an update, it is possible that the new position X_{k+1} lies outside the feasible domain G.

To enforce the constraint, if $X_{k+1} \notin G$, the particle is reflected at the boundary ∂G. The reflection is performed by moving the particle to the intersection point on the boundary and inverting the component of its velocity that is normal to the boundary. For a rectangular domain, this reflection can be implemented coordinate-wise: For each coordinate j, the allowed interval is $[0, L^j]$, where $L^1 = L_x$ and $L^2 = L_y$. If the updated position $\hat{X}^j_{k+1} = X^j_k + hV^j_k$ leaves this interval, the time to reach the boundary is

$$\tau^j = \frac{\partial G^j - X^j_k}{V^j_k},$$

where the boundary is $\partial G^j = L^j$ if $V^j_k > 0$ and $\partial G^j = 0$ if $V^j_k < 0$. The particle is first moved to the boundary, the corresponding velocity component is reflected, and then the particle continues for the remaining time step with the new velocity:

$$X^j_{k+1} = X^j_{\tau,k+1} + (h - \tau^j)(-V^j_k),$$

where $X^j_{\tau,k+1} = X^j_k + \tau^j V^j_k$ is the intersection point at the boundary. For other domains, such as circular regions, the reflection is performed by calculating the intersection point on the boundary ∂G and reflecting the update direction across the normal to the boundary at the point of intersection. The mathematical details of this procedure are given in Section 3.3.

2.5 Criterion for success

The performance will be evaluated on various benchmark functions, and optimal hyperparameters will be determined by maximizing the success rate of finding global optima. A run will be deemed to be a success if there is a particle $X^* \in G$ within a Euclidean distance of 0.2 from the global minimum X_{min}

$$||X_{min} - X^*|| < 0.2, \tag{2.10}$$

and thus the success rate will be the number of runs where this condition was fulfilled divided by the total number of runs performed.

3

Particle swarm optimization with constraints

3.1 Introduction

The first method which is investigated is the application and tuning of particle swarm optimization with constraint on different benchmark test functions.

In our framework, PSO, consisting of N-particles, has the dynamics of the i:th particle described by its position $X^i(t)$ and velocity $V^i(t)$ at time t. The dynamics of each particle are governed by a system of SDEs, which are specified in detail in equation (3.9). The SDE contains four terms: an interaction term that looks like

$$-\int_0^t \lambda(X^i(s) - \bar{X}(s))ds, \tag{3.1}$$

where $\lambda > 0$ and $\bar{X}(s)$ denotes the weighted average of all particles at a time t which is defined as

$$\bar{X}(t) = \sum_{j=1}^{N} \frac{X^j(t)e^{-\alpha U(X^j(t))}}{\sum_{l=1}^{N} e^{-\alpha U(X^l(t))}}, \tag{3.2}$$

where $\alpha > 0$.

For brevity, we denote

$$\bar{X}(t) = \sum_{j=1}^{N} X^j(t)w^j, \tag{3.3}$$

where

$$w^j := \frac{e^{-\alpha U(X^j(t))}}{\sum_{l=1}^{N} e^{-\alpha U(X^l(t))}}. \tag{3.4}$$

Note that as the parameter α increases and tends to infinity, more and more weight is assigned to the position $X^j(t)$ with the current smallest corresponding objective function value $U(X^j(t))$, leading to this weighted average of positions approaching $\min U(X^j(t))$, the best particle method.

The second term of the equations of motion for PSO is the friction term given by

$$-\int_0^t \gamma V^i(s)ds, \tag{3.5}$$

where γ is a hyperparameter to be tuned. Third is the noise term accounting for the exploration by the particles. This term is given by

$$\int_0^t \sigma(s)dB^i(s), \tag{3.6}$$

where $B^i(t)$ is a Brownian motion and $\sigma(t)$ is a non-increasing function that decides how much the particles should explore. A first choice of sigma used in implementation is

$$\sigma(t) = \begin{cases} 1, & \text{if } t < t_0, \\ 10^{-5}, & \text{if } t \geq t_0, \end{cases} \tag{3.7}$$

where t_0 is a time which has to be determined, meaning exploration will be valued a lot before t_0 and not at all after. The final term handling the imposed constraint of being bounded in domain $X(t) \in G, G \subset \mathbb{R}^d$ is given by

$$-2\left\langle n(X^i(t)), V^i(t)\right\rangle n(X^i(t))I_{\partial G}(X(t)), \tag{3.8}$$

where $n(\mathbf{x})$ is a unit outward normal at $\mathbf{x} \in \partial G$, $\langle n(\mathbf{x}, \mathbf{v})\rangle$ is the projection of the velocity onto this normal and by multiplying this with the normal vector the orientation is regained.

The equations of motion for particle swarm optimization are obtained by adding the interaction, friction, exploration, and constraint terms and this can be written in continuous time formulation as

$$\begin{aligned} X^i(t) &= X^i(0) + \int_0^t V^i(s)ds \\ V^i(t) &= V^i(0) - \lambda\int_0^t (X^i(s) - \bar{X}(s))ds - \gamma\int_0^t V^i(s)ds + \int_0^t \sigma(s)dB^i(s) \\ &\quad - 2\sum_{0<s\leq t} \langle n(X^i(s)), V^i(s)\rangle n(X^i(s))I_{\partial G}(X^i(s)), \end{aligned} \tag{3.9}$$

where λ and γ are hyper-parameters which are tuned via testing in a range and evaluated in Section 3.3. Note that the PSO method is gradient-free.

3.2 Implementation

To use these SDEs for numerical simulations they have to be discretized. The discretization is based on Euler-Maruyama scheme. We take a uniform partition of $[0, T]$ with step-size h as $0 = t_0 <, \ldots, t_n = T$ with $h = T/n$. We denote our numerical scheme with $(X_k, V_k)_{k=0}^n$. We symbolically write our numerical scheme in equation (3.10) and then further explain the algorithm below.

$$\begin{aligned} X_{k+1}^i &= X_k^i + V_k^i h, \\ V_{k+1}^i &= V_k^i - \lambda(X_{k+1}^i - \bar{X}_{k+1})h - \gamma V_k^i h + \sigma_k \sqrt{h}\xi_{k+1}^i + \Delta R_k, \end{aligned} \tag{3.10}$$

note that $\sigma_k = \sigma(t_k)$, and ΔR_k is the reflective term in discrete time formulation which we elaborate later. The discretization of the reflective term requires some explanation since, unlike the continuous case, we can no longer rely on a boundary indicator function, as particles may step outside the domain G in a single time step. Instead, we define a different procedure to handle the reflections at the boundary. First, a candidate for the new position is calculated as any other via the velocity vector: $\hat{X}_{k+1}^i = X_k^i + hV_k^i$. We then check if $\hat{X}_{k+1}^i$ lies outside the domain G. If it is, we apply the following reflection procedure:

- Find the intersection point $X_{\tau,k+1}^i$ on the boundary ∂G, where $\tau \in (0, h)$ is the fraction of the time step at which the particle would cross the boundary.
- Calculate the reflected velocity $\hat{V}_k^i$ using the normal vector at the intersection point (see Step 6 in Algorithm 1).
- Update the position by moving along the reflected trajectory for the remaining fraction of the time step, ensuring the total distance traveled is one that corresponds to a step size of h.

If instead the position did not exceed the bounds of G, it is accepted without reflection.

The algorithm can be expressed in more detail as follows:

Algorithm 1 Discrete Reflection Procedure for Boundary Handling

1: Compute candidate position: $\hat{X}_{k+1}^i = X_k^i + hV_k^i$
2: **if** $\hat{X}_{k+1}^i \notin G$ **then**
3: Find $\tau \in (0, h)$ and $X_{\tau,k+1}^i \in \partial G$ such that:
4: $X_{\tau,k+1}^i = X_k^i + \tau V_k^i$
5: Calculate reflected velocity:
6: $\hat{V}_k^i = V_k^i - 2\langle n(X_{\tau,k+1}^i), V_k^i \rangle n(X_{\tau,k+1}^i)$
7: Update position:
8: $X_{k+1}^i = X_{\tau,k+1}^i + (h - \tau)\hat{V}_k^i$
9: **else**
10: $X_{k+1}^i = \hat{X}_{k+1}^i$
11: $\hat{V}_k^i = V_k^i$
12: **end if**
13: Update velocity for next timestep:
14: $V_{k+1}^i = \hat{V}_k^i - \lambda(X_{k+1}^i - \bar{X}_{k+1})h - \gamma \hat{V}_k^i h + \sigma_k \xi_{k+1}^i \sqrt{h}$

3.3 Numerical illustration

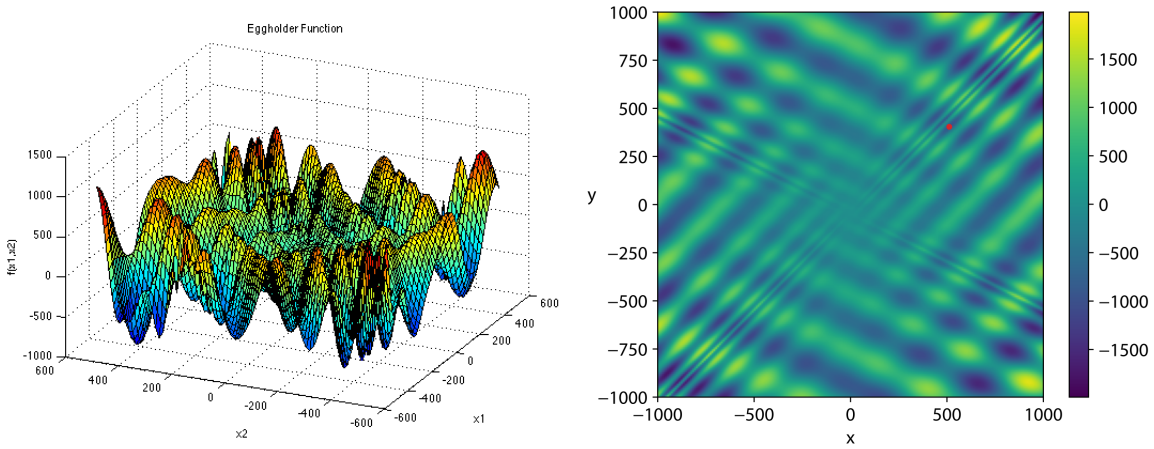
During the tuning process, we consider two test-functions in order to evaluate and compare the performance of different hyperparameter configurations. Since the equations of motion for PSO are gradient free it allows for benchmark functions that are non-differentiable. Two different test functions are used: Eggholder function, and Cross-in tray function.

Eggholder The Eggholder function is defined as

$$U(x,y) = -(y+47)\sin\sqrt{\left|\frac{x}{2}+(y+47)\right|} - x\sin\sqrt{|x-(y+47)|}, \tag{3.11}$$

where $x, y \in [-512, 512]$, with a global minimum $U(512, 404.2319) = -959.6407$

The Eggholder function is shown in Figure 3.1.



(a) Surface plot of the Eggholder function. Image from [16], used under CC BY-SA 4.0 license.

(b) Contour plot of the Eggholder function, with the global minimum shown as a red dot. Image by Nschloe [17], used under CC BY-SA 4.0 license.

Figure 3.1: Surface plot and contour plot of the Eggholder function, one of the two test functions used during tuning of the method in this research project. The function has a global minimum $f(512, 404.2319) = -959.6407$ shown as a red dot in the contour plot.

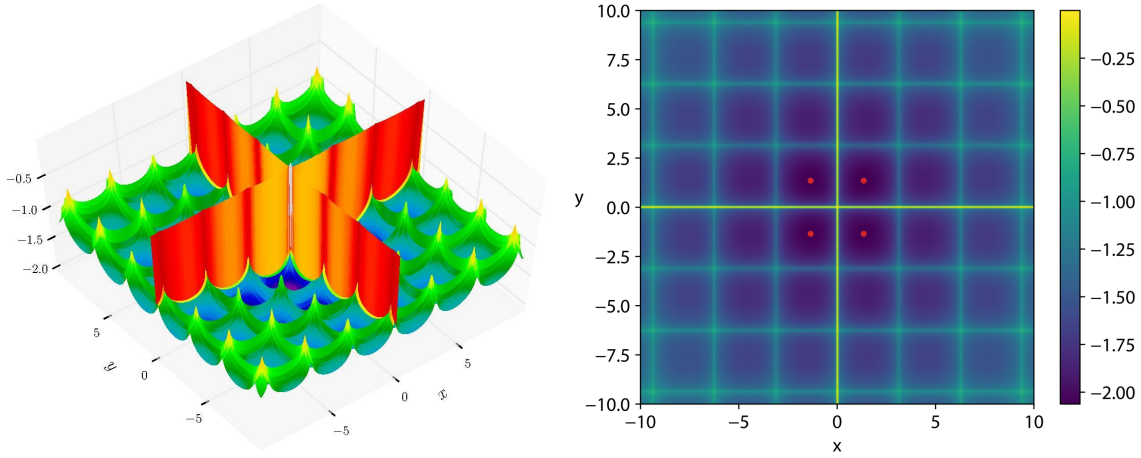
The constraint is implemented differently for the two test functions. For the Eggholder function, the search domain is defined as $-512 \leq x, y \leq 512$. The global minimum of the function is located at $(512, 404.2)$, which is approximately 652 units from the origin. Consequently, if a circular constraint were to be applied, the radius would need to be at least 653 to encompass the global minimum, and thus expanding the total search domain. However, such an expansion could potentially introduce additional global minima, thereby altering the problem landscape. To avoid this issue, a

square constraint is employed instead. Under this approach, when a particle reaches the boundary of the domain, the relevant component of its velocity is reversed, effectively reflecting the particle back into the feasible region. The mathematical detail of enforcing the square constraint can be seen in Section 2.4.

Cross-in tray The Cross-in-tray function is defined as

$$U(x,y) = -0.0001 \left[\left| \sin x \sin y \exp\left(\left| 100 - \frac{\sqrt{x^2+y^2}}{\pi} \right| \right) \right| + 1 \right]^{0.1} \tag{3.12}$$

with four global minima $\text{Min} = \begin{cases} U\left(1.34941, -1.34941\right) & = -2.06261 \\ U\left(1.34941, 1.34941\right) & = -2.06261 \\ U\left(-1.34941, 1.34941\right) & = -2.06261 \\ U\left(-1.34941, -1.34941\right) & = -2.06261 \end{cases}$, and it is shown in Figure 3.2.



(a) Surface plot of the Cross in tray function. Image from [18], used under CC BY-SA 3.0 license.

(b) Contour plot of the Cross-in-Tray function, with global minima shown as red dots. Image by Nschloe, used under CC BY-SA 4.0 license [19].

Figure 3.2: Surface plot and contour plot of the Cross-in tray function, the other of the two test functions used during tuning of the method in this research project. The function has four global minima $f(\pm 1.34941, \pm 1.34941) = -2.06261$ shown as a red dot in the contour plot.

For the Cross-in tray function a circular constraint with radius r is used since the global minima are comfortably inside the known search domain of $-10 \leq x, y \leq 10$.

When a particle exits the domain G, reflection requires enforcing the boundary condition. For a circular domain $G = \{(x,y) : x^2 + y^2 < r^2\}$ with boundary $\partial G = \{(x,y) : x^2 + y^2 = r^2\}$, the parameter τ, which is the distance the particle

travels before intersecting the boundary, can be found by solving the quadratic equation:

$$(b_1^2 + b_2^2)\tau^2 + 2(a_1b_1 + a_2b_2)\tau + (a_1^2 + a_2^2 - r^2) = 0, \tag{3.16}$$

where a_1 and a_2 are the initial coordinates of the particle X_k^i, and b_1 and b_2 are the components of its velocity V_k^i. This equation follows directly from the requirement that the particle lies on the boundary at time τ:

$$(a_1 + \tau b_1)^2 + (a_2 + \tau b_2)^2 = r^2. \tag{3.13}$$

The particle's motion must satisfy

$$\begin{aligned} &\mathbf{X} + h\mathbf{V} \in G^C, \\ &\mathbf{X} + \tau\mathbf{V} \in \partial G, \quad \tau \in (0, h), \end{aligned}$$

where $\mathbf{X}$ and $\mathbf{V}$ denote the position and velocity vectors, respectively.

Once τ is calculated, the outward normal vector at the boundary needs to be determined. For a spherical or circular domain, this normal is simply given by the normalized position vector of the particle at the boundary, $X_{\tau,k+1}^i$. Thus, the outward normal for particle i at time $t = kh + \tau$ is:

$$n(X_{\tau,k+1}^i) = \frac{X_{\tau,k+1}^i}{||X_{\tau,k+1}^i||}. \tag{3.14}$$

3.3.1 PSO performance on Cross-in Tray

During the tuning process a point of interest is finding the optimal configuration of the hyperparameters for the different benchmark functions. Optimal in this case was defined as the configuration which maximized the success rate defined in equation (2.10). The amount of weight assigned to the particle with the lowest corresponding function value is governed by the α hyperparameter. By fixing α and running a primitive parameter search, testing values of $\lambda = 1.0, 3.25, 5.5, 7.75, 10$, and $\gamma = 0.1, 0.2, ..., 1$ and testing each configuration only once we can get an idea of how the success rate depends on λ and γ. Then by redoing the same search with a different α will give a glimpse of the impact α has. Testing a single run per configuration on the Cross-in tray function with $\alpha = 10$ fixed we obtain the results in Table 3.1.

Although a single run per configuration is a much too small sample size to draw any meaningful conclusion, this indicates that for the configuration of $\lambda = 10, \gamma = 1$ there is at the very least a possibility of find the global minimum. Now, redoing the same search with $\alpha = 100$, assigning even more weight to the points which are closest to the global minimum we get the results shown in Table 3.2.

Now with a higher α the amount of configurations leading to a successful run has increased noticeably. More so than any insight on the configurations of λ and γ

Table 3.1: Results of parameter search for λ and γ pairs with $\alpha = 10$. 'X' indicates failure, 'O' indicates success. The total runtime was $T = 5$, exploring for $t_0 = 3$, the step-size was $h = 0.01$ and the number of particles was $N = 50$, σ was as defined in equation (3.7).

λ	γ									
	0.1	0.2	0.3	0.4	0.5	0.6	0.7	0.8	0.9	1.0
1.0	X	X	X	X	X	X	X	X	X	X
3.25	X	X	X	X	X	X	X	X	X	X
5.5	X	X	X	X	X	X	X	X	X	X
7.75	X	X	X	X	X	X	X	X	X	X
10.0	X	X	X	X	X	X	X	X	O	O

Table 3.2: Results of parameter search for λ and γ pairs with $\alpha = 100$. 'X' indicates failure, 'O' indicates success. The total runtime was $T = 5$, exploring for $t_0 = 3$, the step-size was $h = 0.01$ and the number of particles was $N = 50$, σ was as defined in equation (3.7).

λ	γ									
	0.1	0.2	0.3	0.4	0.5	0.6	0.7	0.8	0.9	1.0
1.0	X	X	X	X	X	X	X	O	O	O
3.25	X	X	X	X	X	X	O	O	X	X
5.5	X	X	X	O	O	O	O	O	O	X
7.75	O	X	O	O	X	O	O	O	O	O
10.0	X	X	X	X	X	O	O	O	O	O

this indicates that having a higher α value greatly increases the chance of success. This is investigated more thoroughly by increasing the runs to 100 and taking the percentage of successful runs as the success rate. By fixing $\gamma = 1$ and testing λ values ranging from 1 to 10 with α fixed to either 10 or 100 we get results shown in Table 3.3.

These results show that having a higher α value is significantly more impactful than the specific choice of λ, with the exception of $\lambda = 3$, which exhibits an unexpectedly low success rate. The cause of this anomaly is unclear and was not investigated further in this study. The rest of the results suggest that the dependence on α dominates to such an extent that setting α to a larger value becomes the most straightforward way to increase the success rate. However, as α approaches infinity, the weighted average effectively assigns nearly all the weight to the particle with the lowest corresponding function value, making it equivalent to simply using the best particle. Therefore, using a weighted average and using the largest α possible without causing overflow in the code becomes redundant, as we can achieve the same outcome by directly adopting a "best particle" approach, thereby eliminating α dependence altogether.

The impact of the λ and γ parameters can be seen by varying them and plotting the final positions. The resulting plots can be seen in Figure 3.3.

Table 3.3: Success rates for 100 runs with varying λ values with $\gamma = 1$. The total runtime was $T = 5$, exploring for $t_0 = 3$, the step-size was $h = 0.01$ and the number of particles was $N = 50$, σ was as defined in equation (3.7).

λ	Success Rate [$\alpha = 10$]	Success Rate [$\alpha = 100$]
1.0	0.0 %	87.0 %
2.0	2.0 %	99.0 %
3.0	26.0 %	5.0 %
4.0	25.0 %	100.0 %
5.0	19.0 %	97.0 %
6.0	2.0 %	95.0 %
7.0	4.0 %	99.0 %
8.0	0.0 %	98.0 %
9.0	1.0 %	74.0 %
10.0	29.0 %	98.0 %

The global minima are marked with yellow stars and the red dots are individual particles. There are 1000 particles shown in Figure 3.3 for visual clarity, which is more than necessary for a search domain of this size. One can directly see the impact the friction parameter γ has, as the particle distribution becomes much denser with increasing γ, indicating faster convergence. The λ parameter appears to affect the shape of the distribution; with smaller λ, the interaction strength between particles is lower.

In the case of large λ and small γ, the particles are spread out and do not collapse to a single point, but the distribution centers near one of the global minima. If λ is smaller and γ is 1, the particles still collapse, but the centering is less accurate. This suggests that λ influences how well the particles orient towards the best found minimum, which is reasonable since λ is tied to the interaction term in the equations of motion. Meanwhile, the friction parameter γ mainly affects how quickly the swarm collapses.

Although Figure 3.3 uses 1000 particles, it is not clear that such a large number is necessary to find a global minimum of the Cross-in tray function. To investigate this, the number of particles was varied ($N = 3, 5, 10, 20, 50$), with each case tested 1000 times. The success rate was calculated as the proportion of runs where the global minimum was found within a Euclidean distance of 0.2. The results are shown in Table 3.4.

Table 3.4: Success rates for 1000 runs with varying numbers of particles (N). The total runtime was $T = 5$, exploring for $t_0 = 3$, the step-size was $h = 0.01$ and the number of particles was $N = 50$, σ was as defined in equation (3.7), $t_0 = 3$, $\lambda = 10$ and $\gamma = 1$.

N	3	5	10	20	50
Success Rate	60.30%	89.90%	99.40%	100.00%	100.00%

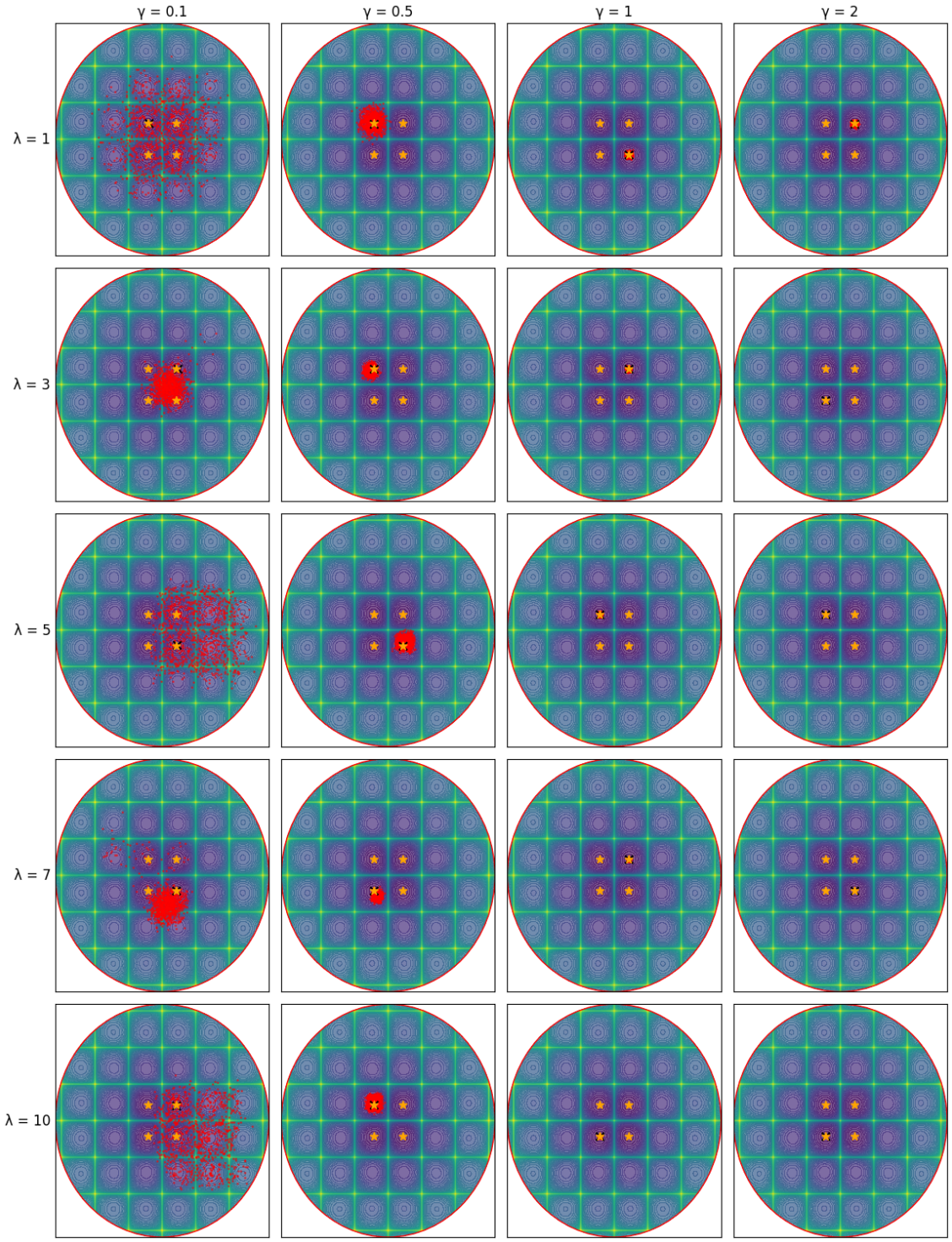


Figure 3.3: Plot visualizing the impact of γ and λ has on particles' behaviour for the Cross-in tray function. In each plot the global minima are marked with yellow stars, the best particle with a black cross, and each red dot is a particle. There are 1000 particles in each plot and the parameters being tested are $\gamma = 0.1, 0.5, 1, 2$ and $\lambda = 1, 3, 5, 7, 10, 15$. Each simulation ran for a total runtime of $T = 10$, exploring for $t_0 = 3$, with a stepsize $h = 0.01$.

The particles are instantiated uniformly in the square $-7 \leq x, y \leq 7$, so that no particles are initialized outside the circular constraint of radius $r = 10$. Here we can see that the 1000 particles used to visualize the impact of γ and λ in Figure 3.3 are several orders of magnitude larger than what is actually required for the method to reliably find the global minimum of the Cross-in tray function.

Since we are evaluating an interacting particle method, the results in Table 3.4 make it difficult to actually test which hyperparameter configuration is optimal when using many particles. With as few as 20 particles, the success rate already reaches 100%, so increasing N further doesn't provide any additional information—almost any reasonable configuration will succeed. This means that for the Cross-in tray function tuning the hyperparameters to find the optimal configuration will be difficult since the global minimum is very commonly found regardless of the configuration. Therefore it is more fitting to tune these parameters on the Eggholder function instead.

3.3.2 PSO performance on Eggholder function

To find the optimal hyperpparameters for the Eggholder function a more extensive parameter search is carried out, testing each configuration 1000 times and calculating the success rate as the share of these 1000 runs which found the global minimum. The first parameter search, whose results can be seen in Table 3.5, tests values of λ and γ that are spaced quite far apart from each other. This initial, broader search serves the purpose of identifying promising regions in the parameter space, providing a general direction for a subsequent, more focused search to fine-tune the optimal values for the method. For these parameter searches the total runtime T was set to 10, with a time step of $h = 0.01$, the exploration time parameter t_0 in σ was set to 7 and each configuration was tested 1000 times.

Table 3.5: Preliminary parameter search results for 1000 runs of different λ and γ combinations for PSO on the Eggholder function. The total runtime was $T = 10$, exploring for $t_0 = 7$, the step-size was $h = 0.01$ and the number of particles was $N = 1000$, σ was as defined in equation (3.7).

λ	γ				
	0.1	1	5	10	15
$\lambda = 0.1$	0.0%	0.0%	0.0%	0.0%	0.0%
$\lambda = 1$	0.1%	3.4%	0.0%	0.0%	0.0%
$\lambda = 5$	0.1%	34.0%	0.5%	0.0%	0.0%
$\lambda = 10$	0.0%	19.4%	3.1%	0.1%	0.1%
$\lambda = 15$	0.0%	21.4%	1.4%	0.4%	0.0%

The results from this first parameter search show that γ values around 1 should be investigated more thoroughly and that λ appears to show promise for several values ranging between 5 and 15. The second parameter search explores these combinations further, and the results are shown in Table 3.6.

Table 3.6: Success rates from 1000 runs of each different combination of λ and γ for fine tuning the parameters for the Eggholder function. The total runtime was $T = 10$, exploring for $t_0 = 7$, the step-size was $h = 0.01$ and the number of particles was $N = 1000$, σ was as defined in equation (3.7), $\lambda = 10$ and $\gamma = 1$.

λ	γ						
	0.8	1	1.25	1.5	1.75	2.0	2.25
$\lambda = 5$	13.5%	35.7%	60.7%	25.5%	12.3%	12.5%	15.9%
$\lambda = 8$	7.9%	18.6%	43.2%	67.5%	36.7%	19.4%	12.7%
$\lambda = 10$	9.0%	17.8%	54.1%	47.0%	74.4%	36.1%	18.1%
$\lambda = 12$	8.7%	17.5%	35.7%	52.5%	58.7%	71.6%	30.9%
$\lambda = 15$	10.2%	22.4%	29.8%	52.6%	45.2%	62.6%	71.7%

Upon closer examination, a distinct pattern emerges in the form of a diagonal band of high success rates. This band starts from the upper left of the table ($\lambda = 5$, $\gamma = 1.25$) and extends towards the lower right ($\lambda = 15$, $\gamma = 2.25$). The optimal configuration appears to be $\lambda = 10$ and $\gamma = 1.75$, yielding a peak success rate of 74.0%. However, other high-performing combinations (>70%) include $\lambda = 8$ with $\gamma = 1.5$, $\lambda = 12$ with $\gamma = 2.0$, and $\lambda = 15$ with $\gamma = 2.25$. This diagonal trend suggests a relationship between λ and γ where, if λ increases, γ can also be increased to maintain high performance. The performance rapidly decreases as the combinations of γ and λ moves away from this diagonal band in either direction.

The amount of reflections that occurred during the final run for each configuration can be seen in Table 3.7.

Table 3.7: Amount of reflections from all particles during the final 1000th run from the refined parameter search results for fine tuning the parameters for the Eggholder function

λ	γ						
	0.8	1	1.25	1.5	1.75	2.0	2.25
$\lambda = 5$	7347	2607	1902	2244	1306	1792	1153
$\lambda = 8$	6777	6156	4043	1643	2439	2035	1683
$\lambda = 10$	10614	5401	2919	2094	9362	9478	1823
$\lambda = 12$	8004	6670	5550	10346	3371	10270	2175
$\lambda = 15$	9053	9066	5003	11958	4013	2958	11539

Considering there were 1000 particles to build this statistic, we get that, for the optimal configuration of $\lambda = 10, \gamma = 1.75$, there are approximately 10 reflections per particle during the final run, motivating the inclusion of constraint when optimizing the function. One can see by looking at Tables 3.6 and 3.7 that configurations that are better at finding the global minimum also lead to more reflections, this is likely a consequence of the Eggholder global minimum being located right at the edge of the search domain. The amount of reflections is also high when the γ parameter is smaller, which is reasonable considering this corresponds to lower friction and therefore the particles keeping more of their momentum for longer.

The behavior of the method, when using the optimal parameters $\lambda = 10$ and $\gamma = 1.75$, while searching for the global minimum in the Eggholder function is shown in Figure 3.4. Here the iterations $0, 50, 100, 200, 300, 499$ are plotted, and since a step size of 0.01 is used, this corresponds to snapshot images at times $T = 0, 0.5, 1, 2, 3, 4.99$ during a run.

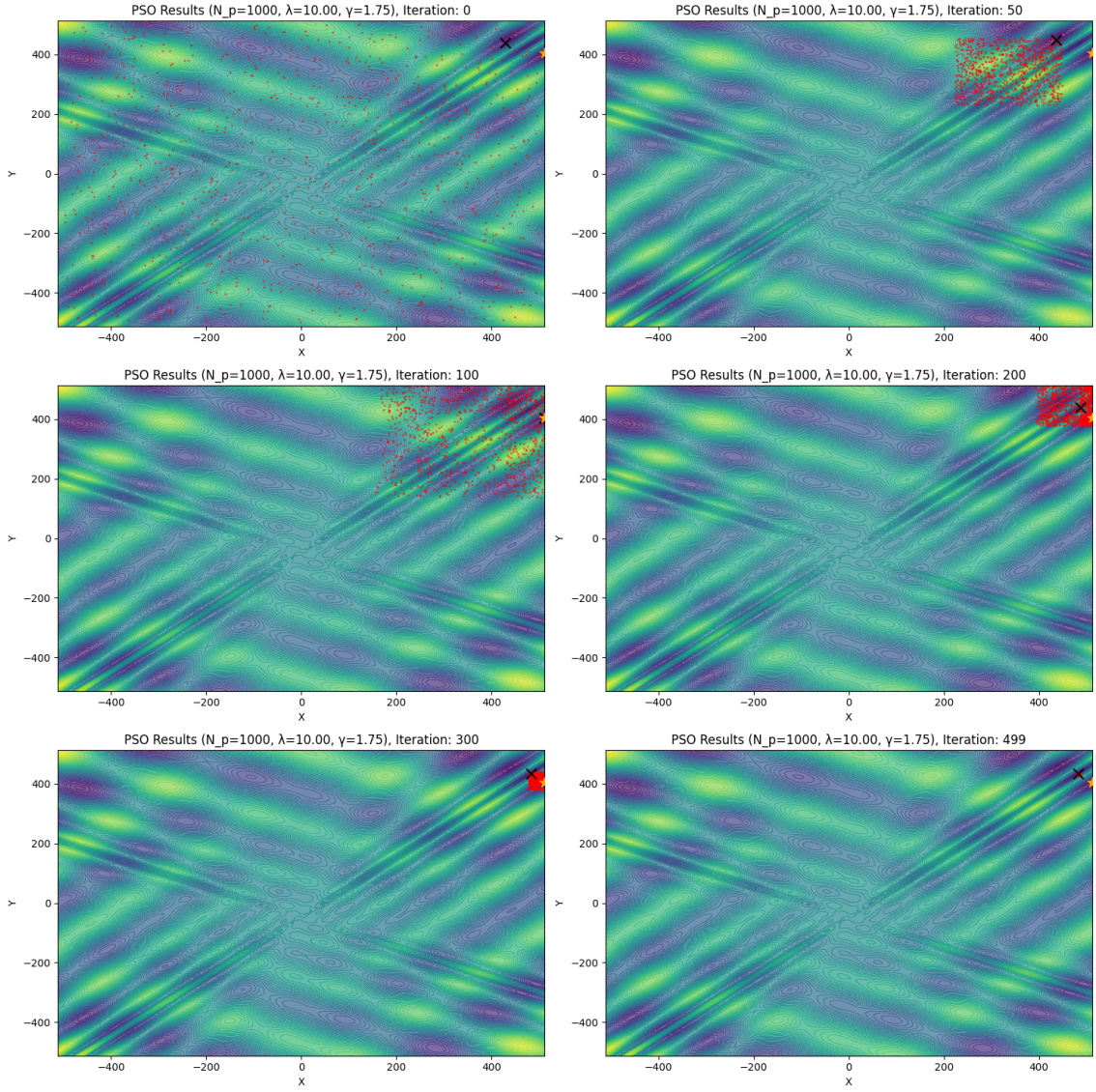


Figure 3.4: Plot visualizing how the particles behave at different points when searching for the global minimum of the Eggholder function. In each plot the global minima are marked with yellow stars, the best particle with a black cross, and each red dot is a particle. There are 1000 particles in each plot, particles are initialized in the square $-500 \leq x, y \leq 500$ and the plot occurs for iterations $0, 50, 100, 200, 300, 499$. Each simulation ran for a total runtime of $T = 5$ with a stepsize $h = 0.01$.

3.3.2.1 Discussion about friction

Something interesting is happening in between iterations 50 and 100 in Figure 3.4 which causes the particles to spread again after they have started converging to the best particle. This is investigated more closely by creating a new plot showing iterations in this interval. A plot showing iterations $50, 58, 64, 72, 80, 88$ can be seen in Figure 3.5.

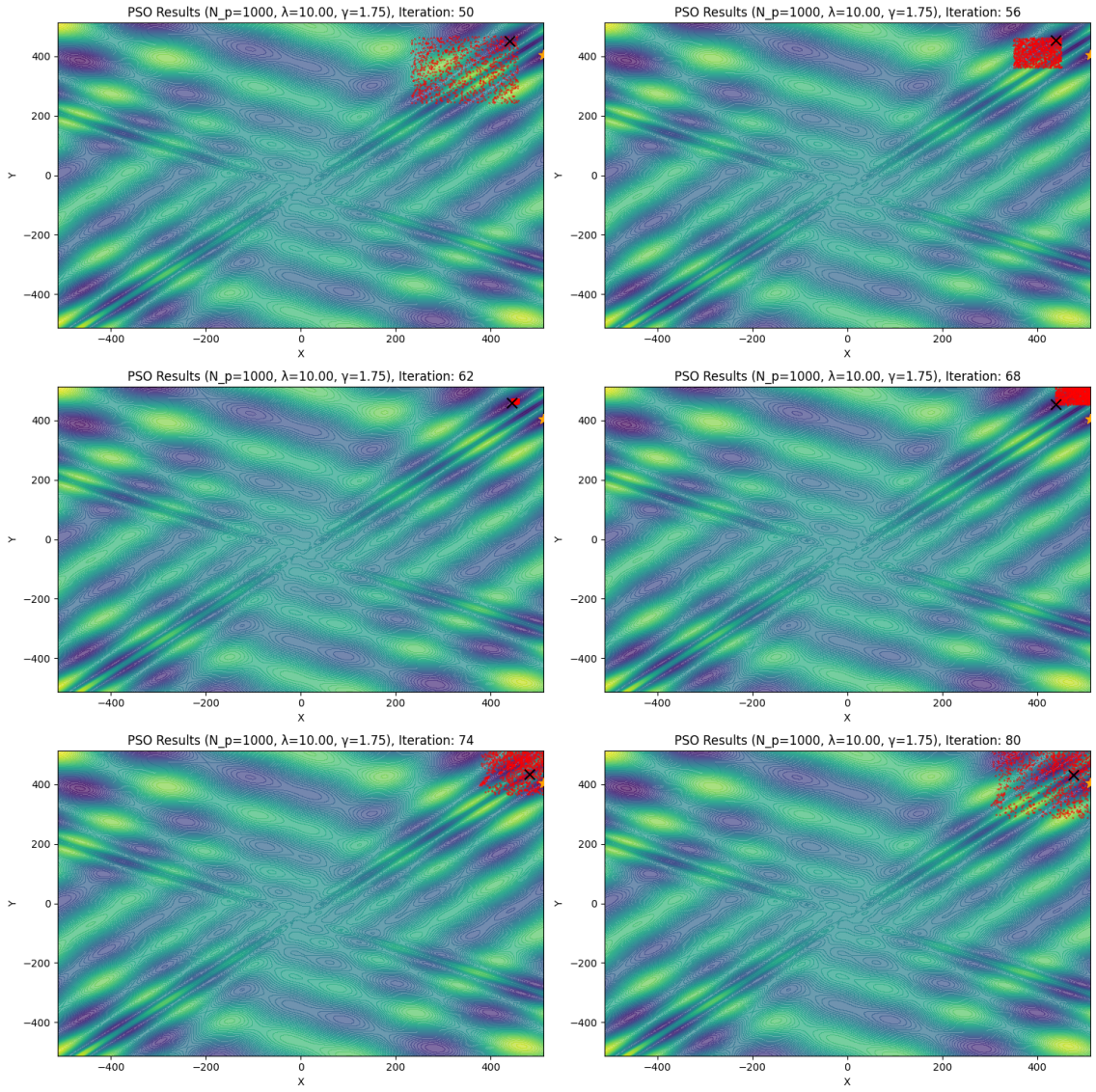


Figure 3.5: Plot visualizing how the particles behave when searching for the global minimum of the Eggholder function. In each plot the global minima are marked with yellow stars, the best particle with a black cross, and each red dot is a particle. There are 1000 particles in each plot, particles are initialized in the square $-500 \leq x, y \leq 500$ and the plot occurs for iterations $50, 56, 62, 68, 74, 80$ Each simulation ran for a total runtime of $T = 5$ with a stepsize $h = 0.01$.

The optimal configuration that was found from the parameter search resulted in

behavior where the particles appear to be converging to the best particle around iteration 62, when exploration is still in full effect, meaning $t < t_0$ and thus the magnitude of the noise term has not decreased yet. The particles then after reaching this point are not slowed down enough and are instead carried further by their momentum. In the run being showcased in the plots the particles then collide with the boundary and are reflected back, resulting in the spread of the particles seen in iteration 100 in Figure 3.4. If the particles' movements were not subject to this constraint, and no new minimum was found, then as they overshoot, their distance to the best particle would increase again. However, this increase would occur in the opposite direction. The particles will then oscillate harmonically around the global minimum until the friction term eventually slows the velocity down for the particles to converge. This behavior is a consequence of the size of the search domain being large, while the γ parameter being quite small to control the velocity of the particles. From the λ term in the equations of motion, large distances cause high velocities which are only counter balanced by braking from the friction term. If the friction increases to $\gamma = 4$, we would expect the duration of this harmonic behavior around the best minimum to shorten and for the particles to converge faster. A plot showcasing this behavior can be seen in Figure 3.6.

Note that this means that our implementation of the PSO method is, for large search domains, sensitive to the location of the best particle at initialization, as this is where the particles which are far away will rush toward. This behavior is not a bad thing and will not be corrected via increasing γ, instead the same optimal combination of $\lambda = 10, \gamma = 1.75$ will continue being used. Increasing γ will greatly hinder the particles' ability at exploring the search domain. There are other ways to handle λ so that the particles stay at their initial positions, utilizing the strength of uniform initialization, and then have the particles move to the most promising location after some time. This could be implemented by adding a time dependence to the λ parameter to have it start low, curbing the instant movement of the far particles towards the best one, and then increase the interaction strength as time goes on. However, this is not implemented in this project.

3.3.2.2 Discussion about noise

The result seen in Figure 3.4 is independent of whether noise is included or not. The same picture is recreated even when setting the noise parameter $\sigma(t) = 0 \quad \forall t \leq T$. This also means, unsurprisingly, that our noise parameter σ which, as defined in (3.7), has a maximum strength of 1, has virtually no impact. One might be surprised at the high success rate obtained for the method when the exploration of the particles is negligible. However, recalling that the particles were initialized uniformly within the square $-512 \leq x, y \leq 512$, their momentum does not need to carry them far. This is because the length of the harmonic oscillation exceeds the distance from the best particle to the global minimum, enabling reliable discovery. If instead initialized in a smaller domain, such as $-300 \leq x, y \leq 300$, particles—even when propelled to the domain's top-right corner—would lack sufficient momentum to traverse the

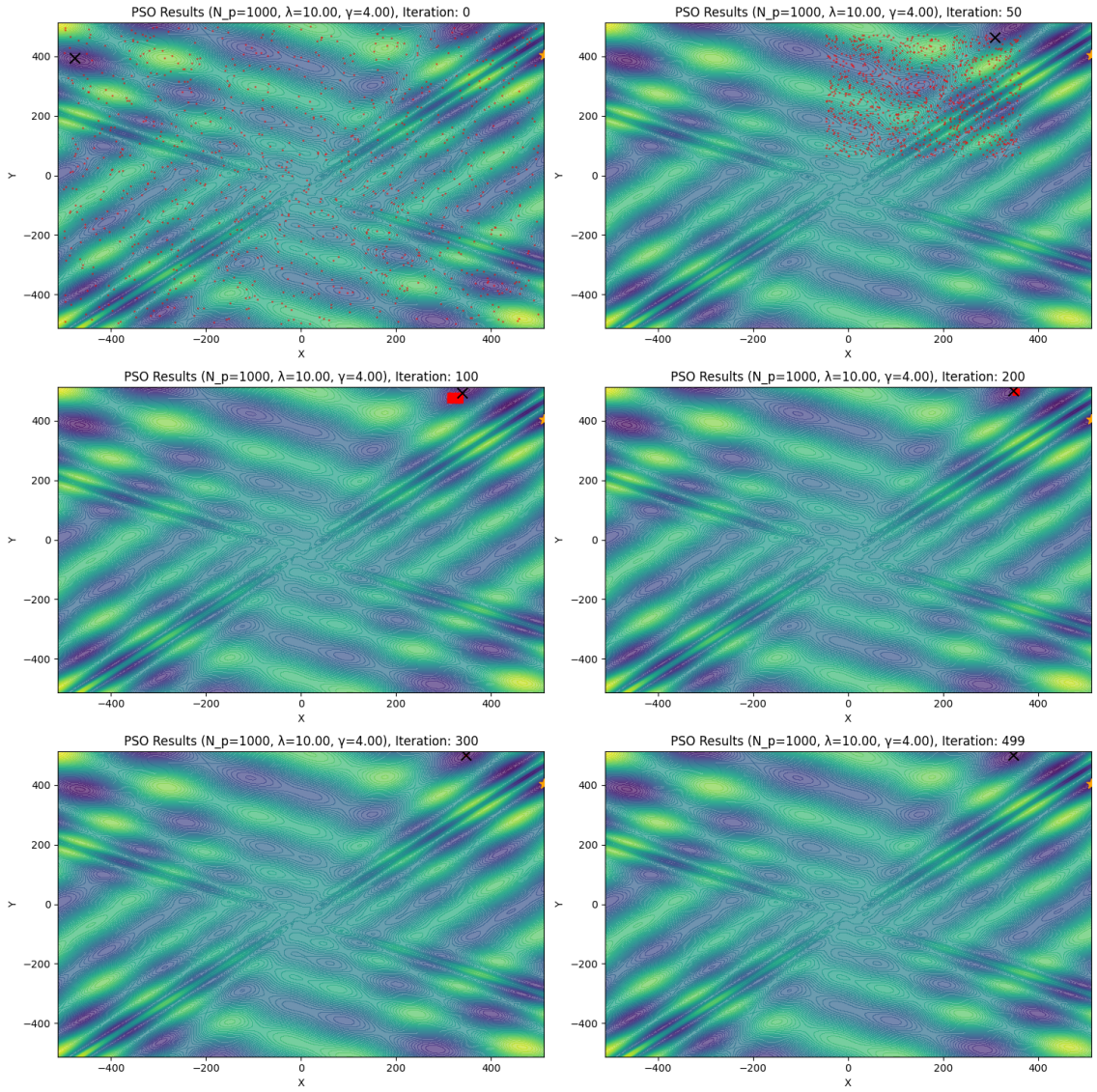


Figure 3.6: Plot visualizing how the particles behave when searching for the global minimum of the Eggholder function. In each plot the global minima are marked with yellow stars, the best particle with a black cross, and each red dot is a particle. There are 1000 particles in each plot, particles are initialized in the square $-500 \leq x, y \leq 500$ and the plot occurs for iterations $0, 50, 100, 200, 300, 499$. Each simulation ran for a total runtime of $T = 10$ with a stepsize $h = 0.01$. Here γ has been increased to 4.

$\approx$200 units required to reach the global minimum without noise. This is illustrated via a success rate investigation with results shown in Table 3.8.

Table 3.8: Success rates for different noise magnitudes ($|\sigma|$) when initialized in $-L \leq x, y \leq L$, for $L = 400, 300, 200, 100$. Other parameter values were $T = 10, h = 0.01, t_0 = 7, \lambda = 10, \gamma = 1.75$

L	$\lvert\sigma\rvert$		
	0	500	$\sqrt{2} \cdot 500$
L = 400	2.1%	7.60%	5.00%
L = 300	0%	20.60%	20.50%
L = 200	0%	16.60%	22.70%
L = 100	0%	35.60%	21.00%

The deterministic PSO without any random noise never finds the global minimum in these cases, whereas with sufficiently large noise (of the order of the radius of the search domain), the method finds it quite consistently. In fact the method appears to improve as the initialization area gets smaller, although that could be caused by function specific conditions which has not been investigated. A possible explanation for this could be that when uniformly starting in the 100×100 square then the function has a bias to have its best located minimum in the top right direction, leading to the particles more often searching this area. A single run of PSO with $|\sigma| = 0$ starting in $-300 \leq x, y \leq 300$ plotted at iterations $0, 100, 300, 500, 700, 999$ can be seen in Figure 3.7.

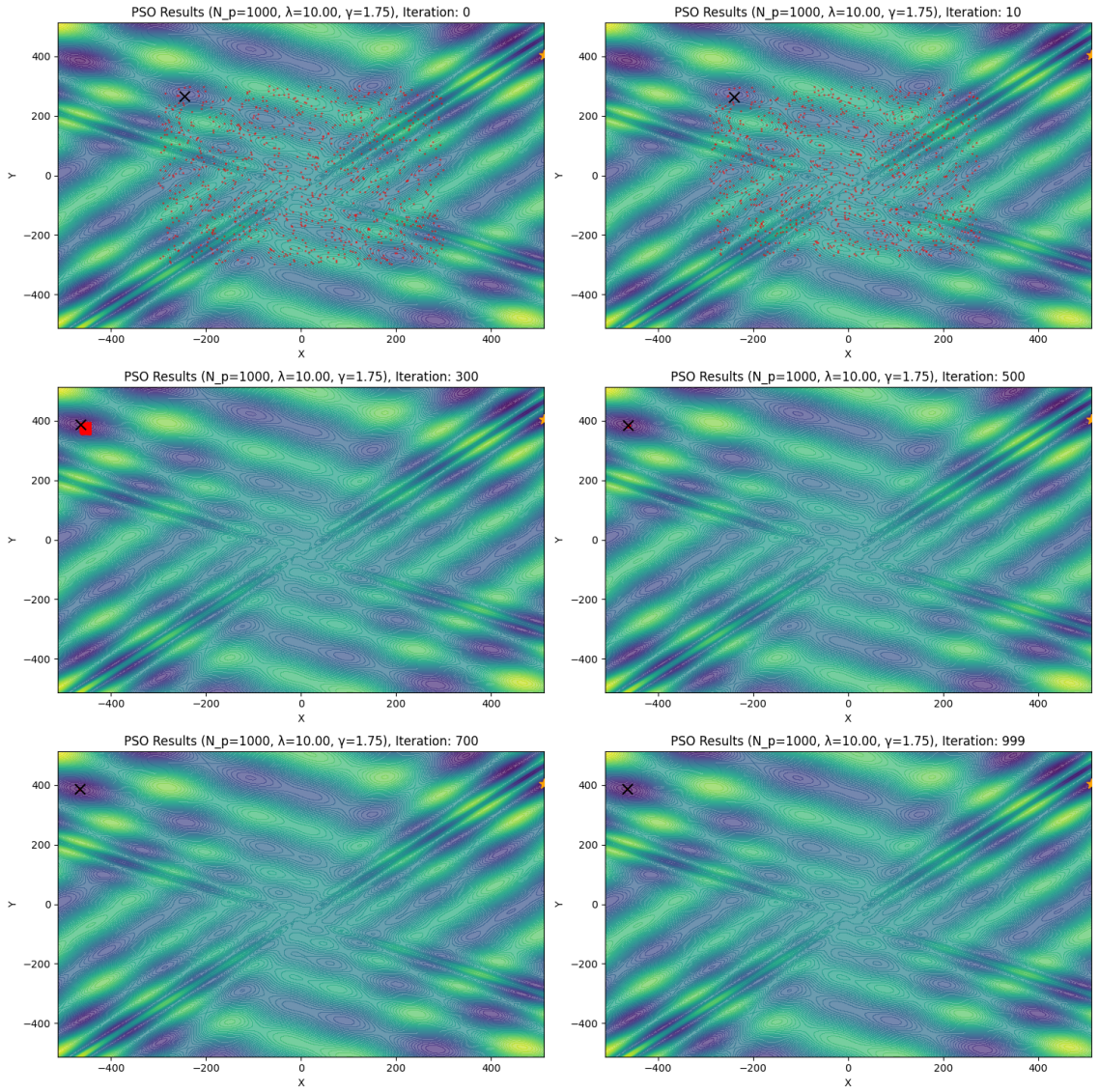


Figure 3.7: Plot visualizing how the particles behave when searching for the global minimum of the Eggholder function. In each plot the global minima are marked with yellow stars, the best particle with a black cross, and each red dot is a particle. There are 1000 particles in each plot, particles are initialized in the square $-300 \leq x, y \leq 300$ and the plot occurs for iterations $0, 100, 300, 500, 700, 799$ Each simulation ran for a total runtime of $T = 10$ with a stepsize $h = 0.01, t_0 = 7$. Here $\gamma = 1.75, \lambda = 10$, $|\sigma| = 0$.

The particles are carried quite far via their momentum, however since the initial best minima, which decides where the particles will be launched, is not in the direction of the global minima the particles are unable to find it. When noise is introduced, with $|\sigma| = 500, t_0 = 7$ and running for a total time of $T = 10$ then the particles can be instantiated in an even smaller regime of $-100 \leq x, y \leq 100$ and they behave as shown in Figure 3.8.

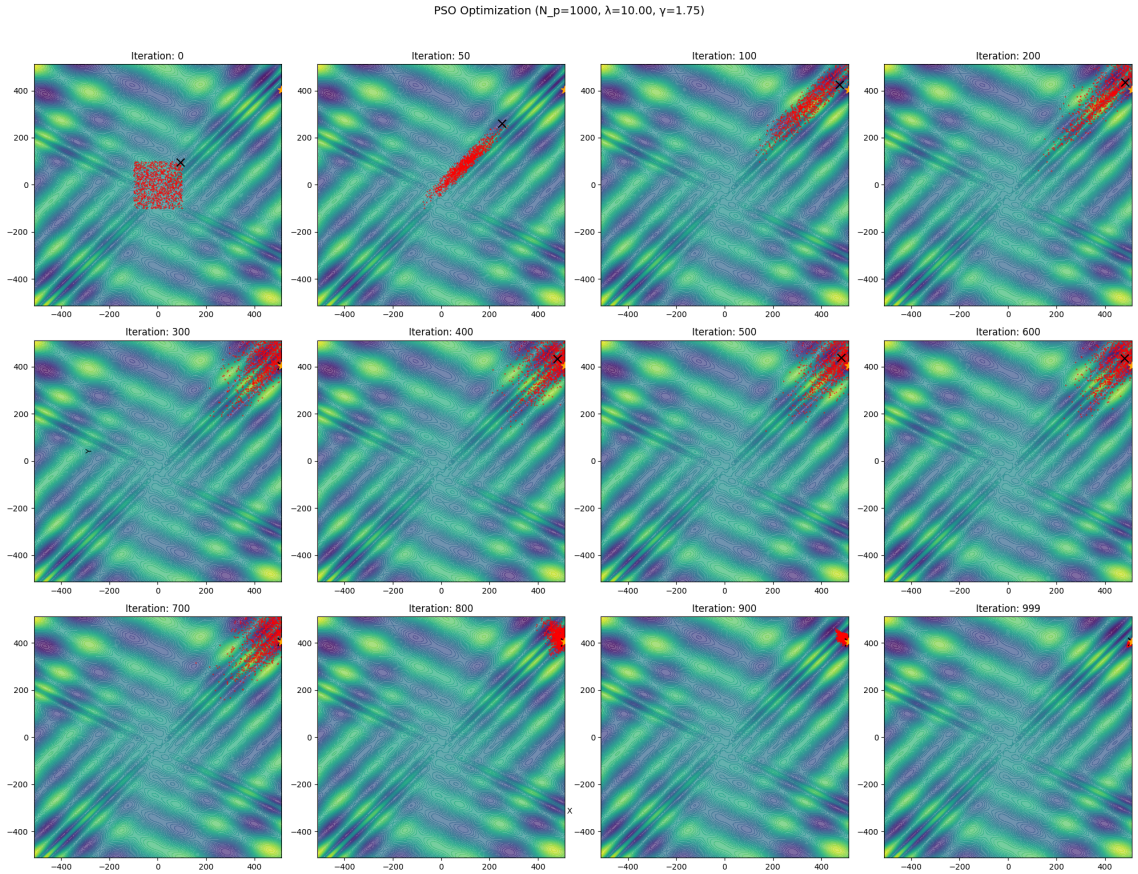


Figure 3.8: Plot visualizing how the particles behave when searching for the global minimum of the Eggholder function. In each plot the global minima are marked with yellow stars, the best particle with a black cross, and each red dot is a particle. There are 1000 particles in each plot, particles are initialized in the square $-100 \leq x, y \leq 100$ and the plot occurs for iterations $0, 50, 100, 200, 300, 400, 500, 700, 800, 800, 900, 999$ Each simulation ran for a total runtime of $T = 10$ with a stepsize $h = 0.01, t_0 = 7$. Here $\gamma = 1.75, \lambda = 10$, $|\sigma| = 500$.

4

Second order (Kinetic) Kalman-Langevin SDE with constraint

4.1 Introduction

The second method investigated in this project is the application of the second-order (Kinetic) Kalman-Langevin stochastic differential equation with constraint on different benchmark test functions. The incorporation of constraints again ensures that the results obtained from this method is in accordance with eventual restrictions, such as positivity constraints for determining ground state energy, by preventing the system from leaving the valid domain $G \subset \mathbb{R}^d$. The constraint is implemented using a reflection term, similar to that used in PSO, which enforces boundary conditions by adjusting the velocity vector when particles reach the boundary of G.

The dynamics of the second-order (kinetic) Kalman-Langevin SDE are described by two coupled equations for position and velocity:

$$\begin{aligned} X^i(t) &= X^i(0) + \int_0^t V^i(s)ds \\ V^i(t) &= V^i(0) - \int_0^t C^N(s)\nabla U(X^i(s))ds - \gamma \int_0^t V^i(s)ds + \int_0^t \sigma(s)Q^N(s)dB^i(s) \\ &\quad - 2\sum_{0<s\leq t} \langle n(X^i(s)), V^i(s)\rangle n(X^i(s))I_{\partial G}(X^i(s)). \end{aligned} \tag{4.1}$$

Here the parameters γ and σ have the same interpretation as in the PSO method, i.e., γ is the hyper-parameter corresponding to friction and $\sigma(t)$ is decreasing function of t and $|\sigma|$ is a hyperparameter pertaining to the magnitude of the noise from $B^i(t)$, the N-dimensional Brownian motion. Brownian motion . There is now an ensemble covariance $C^N(t)$, defined as

$$C^N(t) = \frac{1}{N} Q^N(t) Q^N(t)^\top, \tag{4.2}$$

where $Q^N(t) = \left[X^1(t) - \tilde{X}(t), \ldots, X^N(t) - \tilde{X}(t)\right]_{d\times N}$, for $i = 1, ..., N$, where $\tilde{X}(t)$

is the ensemble mean $\frac{1}{N}\sum_{i=1}^{N} X^i(t)$.

The equations of motion can be expressed in both continuous and discrete time formulations. For numerical simulations, we employ the Euler-Maruyama scheme to discretize these equations:

$$\begin{aligned} X_{k+1}^i &= X_k^i + V_k^i h, \\ V_{k+1}^i &= V_k^i - C_k^N \nabla U(X_k^i)h - h\gamma V_k^i + \sqrt{h}\sigma_k \eta_{k+1}^i (Q_k^N)^\top + \Delta R_k, \end{aligned} \tag{4.3}$$

where h is the time step size, η_{k+1}^i is an $N \times N$ dimensional matrix where each element of the matrix is drawn from a normal distribution $\mathcal{N}(0,1)$, and ΔR_k represents the discrete-time implementation of the reflection term for a rectangular constraint explained in Section 2.4.

4.2 Implementation

As second order Kalman-Langevin SDE contains a gradient of the objective function in its equations of motion, having an analytic expression for the gradient of the objective function makes analysis easier. The first function where the Kalman-Langevin method is applied is on the Rastrigin function defined as

$$U(\mathbf{x}) = An + \sum_{i=1}^{n} \Big[x_i^2 - A\cos(2\pi x_i) \Big], \tag{4.4}$$

where $A = 10$ and, in our 2 dimensional case, $d = 2$, with $x_1 \in [-5.12, 5.12], x_2 \in [-5.12, 5.12]$. The gradient of the Rastrigin function in two dimensions is given by

$$\nabla U(\mathbf{x}) = \left[\frac{\partial U}{\partial x_1}, \frac{\partial U}{\partial x_2} \right]^\top = (2x_1 + 2\pi A \sin(2\pi x_1),\ 2x_2 + 2\pi A \sin(2\pi x_2)) \tag{4.5}$$

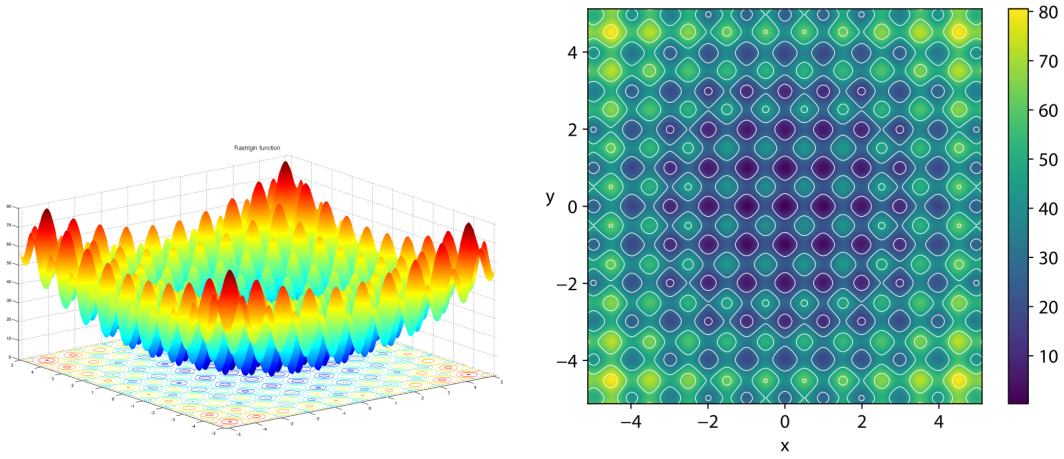
where $A = 10$. The global minimum of the Rastrigin function that the particles will attempt to find is

$$U(0,0) = 0, \tag{4.6}$$

and the Rastrigin function is shown in Figure 4.1.

Comparing the equations of motion for kinetic Kalman-Langevin dynamics with PSO one can see that there is no scaling parameter λ in the interaction term which has to be balanced with the friction parameter γ. Instead, it is now the noise term that is in contention with the friction and the two need to be balanced according to each other. The magnitude of the N-dimensional Brownian motion is therefore proportional to the friction for Kalman-Langevin dynamics and is given by

$$\sigma(t) = \begin{cases} \sqrt{2\gamma}, & t_0 < 0.7T, \\ 10^{-5}, & t_0 \geq 0.7T, \end{cases} \tag{4.7}$$



(a) Surface plot of the Rastrigin function. Image from [20].

(b) Contour plot of the Rastrigin function, with global minima at $(0, 0)$. Image by Nschloe, used under CC BY-SA 4.0 license [21].

Figure 4.1: Surface plot and contour plot of the Rastrigin function. The function has its global minimum $U(0, 0) = 0$ which is in the center of the contour plot.

and since the total runtime will be $T = 10$ for all runs in this chapter, the exploration time $t_0 = 7$ in all experiments presented in this chapter. With a step-size of $h = 0.01$ this corresponds to exploration stopping at iteration 700.

4.3 Numerical illustration

4.3.1 Kalman-Langevin performance on Rastrigin

Although the friction parameter γ was previously tuned to a value of 1.75 for the particle swarm optimization algorithm in Chapter 3, it is not evident that this value will also yield optimal performance for the Kalman-Langevin method. For this reason, we conduct a separate tuning of γ for the Kalman-Langevin method, as described below. Firstly we conduct a sparse search over a broad range of γ values to identify promising regions for further investigation. The values tested are $\gamma = 0.1, 1, 5, 10, 15$ and the results of this search can be seen in Table 4.1.

Table 4.1: Success rates of different values of γ for 1000 runs of second order Kalman-Langevin SDE on the Rastrigin function. The total number of particles was $N = 50$, the runtime was $T = 10$ with a step-size of $h = 0.01$. Also included is the amount of times a particle was reflected at the boundary of the search domain. The reflections are only for the last run, not all 1000 runs.

γ	0.1	1	5	10	15
Success Rate	6.4%	28.1%	62.2%	50.0%	54.4%
Reflections	1492	1060	897	904	899

The results from this sparse search indicate that values of γ between 1 and 5 are promising, so this region is investigated in more detail by taking a closer look with another search using more finely spaced values. The results of this search can be seen in Table 4.2.

Table 4.2: Success rates for different values of γ for 1000 runs of second order Kalman-Langevin SDE on the Rastrigin function. The total number of particles was $N = 50$, the runtime was $T = 10$ with a step-size of $h = 0.01$. Also included is the amount of times a particle was reflected at the boundary of the search domain. The reflections are only for the last run, not all 1000 runs.

γ	1.5	1.75	2	2.5	3	3.5	4	4.5
Success Rate	51.0%	62.3%	71.8%	73.9%	72.4%	71.4%	65.9%	65.0%
Reflections	1055	1011	995	1087	1000	990	965	913

Among the values tested, the highest success rate was achieved at $\gamma = 2.5$.

Now, with the hyperparameter γ having been tuned we can now create a plot illustrating how the particles behave while searching for the global minimum of the Rastrigin function. The plot can be seen in Figure 4.2.

To see how the method performs on average depending on the number of particles present, a thousand runs were done with $\gamma = 2.5$, total runtime of $T = 10$, step-size of $h = 0.01$ with exploration time $t_0 = 7$, same as for when the illustrative plot was generated with the particle count varying as $N = 10, 20, 50, 100, 200$. The resulting success rates can be seen in Table 4.3.

Table 4.3: Success rates for different number of particles N for second order Kalman-Langevin SDE. The total runtime was $T = 10$ on for different numbers of particles ($N = 10, 20, 50, 100, 200$). Also included is the amount of times a particle was reflected at the boundary of the search domain during the last run of each experiment.

N	10	20	50	100	200
Success Rate	38.3%	48.5%	72.2%	91.7%	99.8%
Reflections	33	218	1009	3133	9908

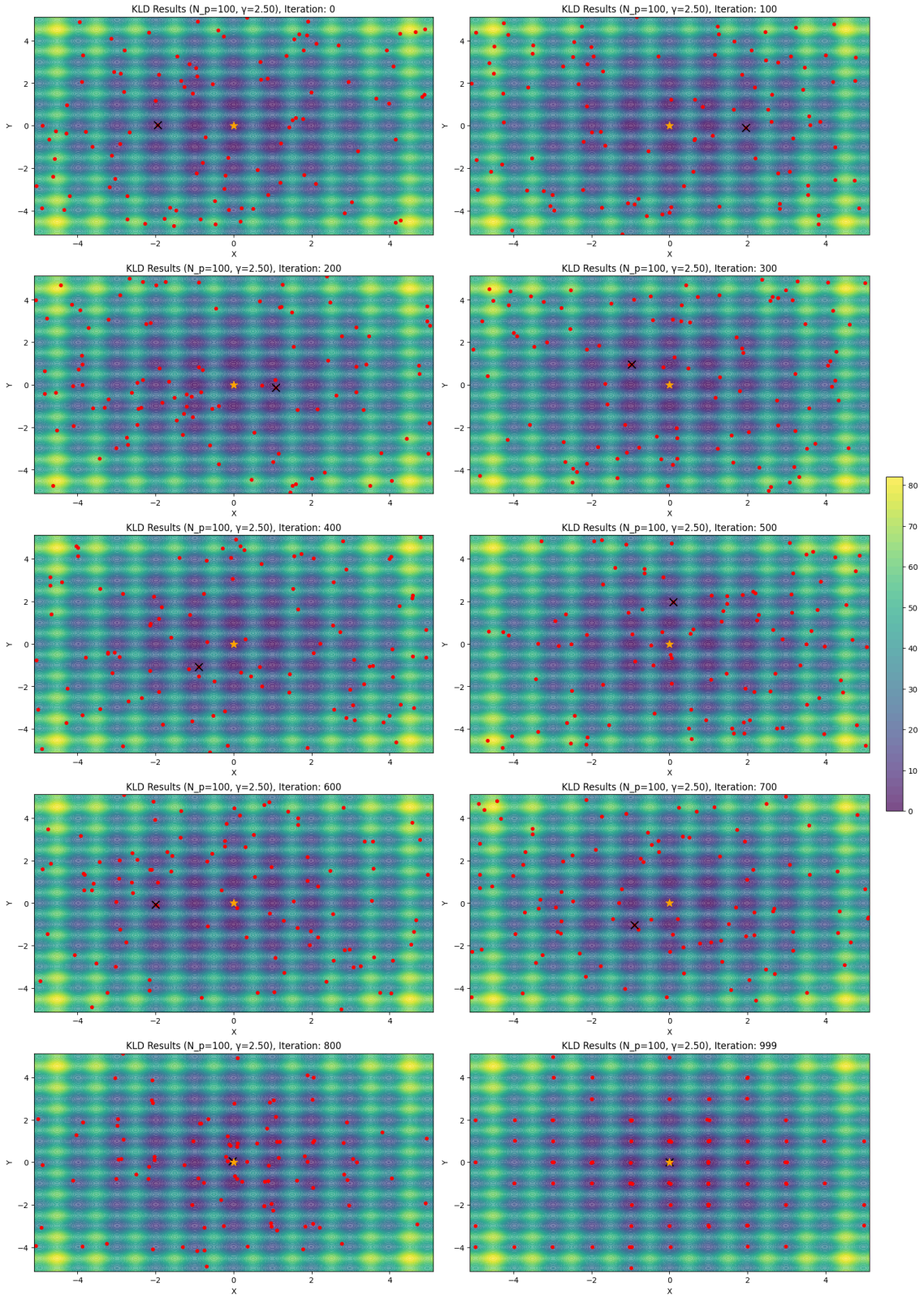


Figure 4.2: Plot showing iterations $0, 100, 200, 300, 400, 500, 600, 700, 800, 999$ of a single run of the second order Kalman-Langevin SDE algorithm defined in (4.1) with 100 particles. The red dots are particles, the black X is the particle with the lowest corresponding function value and the yellow star is the global minimum. Total runtime was $T = 10$ with exploration time $t_0 = 7$, friction parameter was $\gamma = 2.5$ and noise magnitude was $\sqrt{2\gamma}$.

It is interesting to see what impact allowing interaction between particles has on the performance. The interaction for Kalman-Langevin dynamics comes from the covariance matrix, C^N and the matrix Q^N in the gradient term and Brownian term respectively. By removing these two matrices the particles no longer interact with each other and we obtain the second order Langevin SDE defined as

$$\begin{aligned} dX^i(t) &= V^i(t)dt, \\ dV^i(t) &= -\nabla U(X^i(t))dt - \gamma V^i(t)dt + \sigma(t)dB^i(t) + \Delta R(t). \end{aligned} \tag{4.8}$$

The behavior of the particles driven by second order Langevin SDE can be seen in Figure 4.3, and the results from testing how the success rate depends on the number of particles can be seen in Table 4.4.

Table 4.4: Success rates for different number of particles N for non interacting second order Kalman-Langevin SDE. The total runtime was $T = 10$ for different numbers of particles ($N = 10, 20, 50, 100, 200$). Also included is the amount of times a particle was reflected at the boundary of the search domain for the last run.

N	10	20	50	100	200
Success Rate	9.0%	17.8%	38.5%	61.0%	83.3%
Reflections	5	4	16	29	54

For number of particles $N = 10, 20, 50, 100$ the interacting Kalman-Langevin method outperforms the Langevin method when it comes to finding the global minimum by approximately 30 percentage points. As the number of particles increases to 200 then the difference shrinks, likely due to the fact that it becomes increasingly probable that a particle is finding the global minimum just by local gradient.

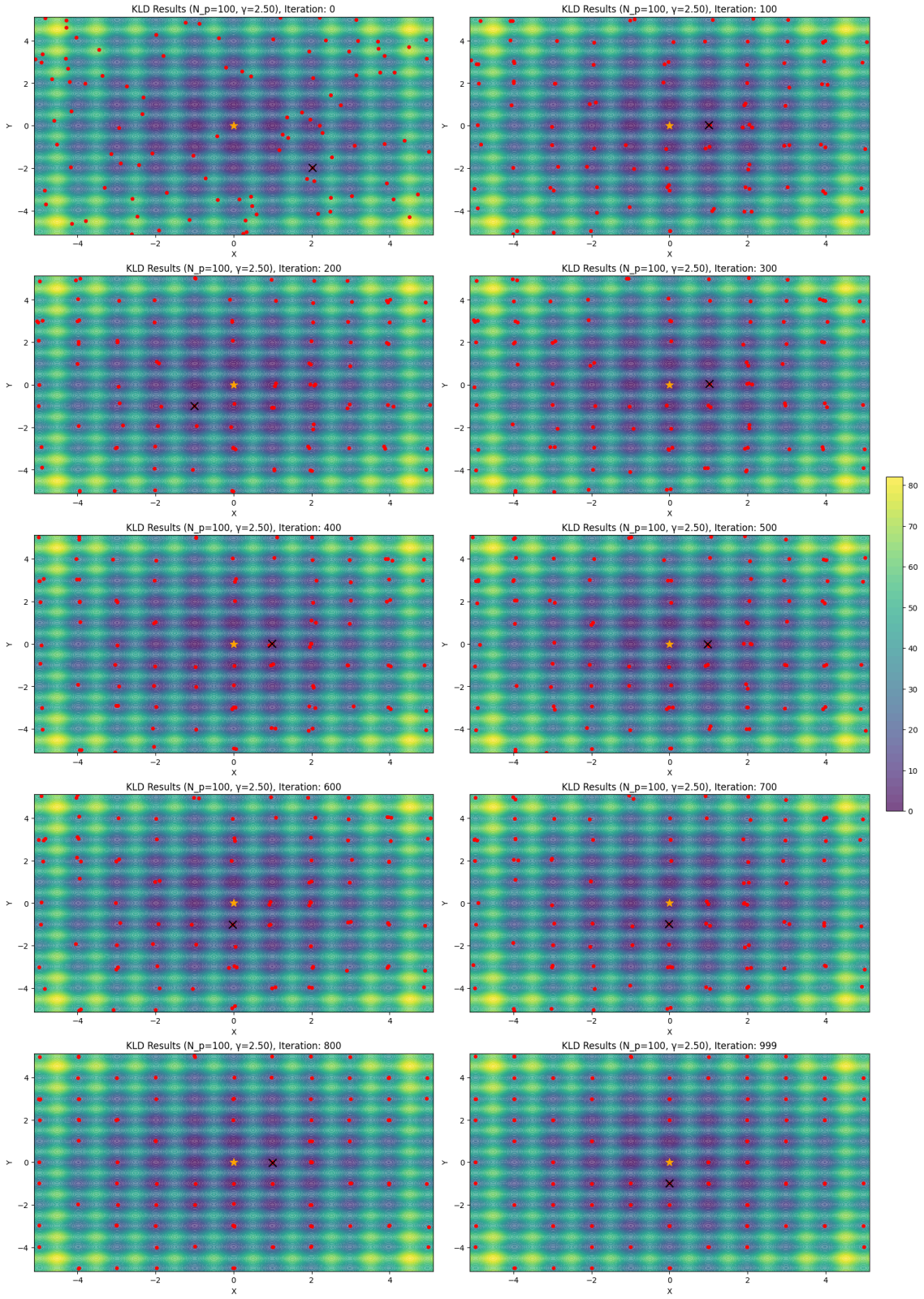


Figure 4.3: Plot showing iterations $0, 100, 200, 300, 400, 500, 600, 700, 800, 999$ of a single run of the Langevin algorithm defined in (4.8) with 100 particles. The red dots are particles, the black X is the particle which has the lowest corresponding function value and the yellow star is the global minimum. Total runtime was $T = 10$ with exploration time $t_0 = 7$, friction parameter was $\gamma = 1.75$ and noise magnitude was $\sqrt{2\gamma}$.

4.3.2 Kalman-Langevin performance on Eggholder

In order to apply the gradient dependent Kalman-Langevin method on the Eggholder function, we need information on the Eggholder's gradient. Although it is possible to derive the Eggholder function's analytical gradient, the gradient is instead locally estimated numerically by using the finite difference method and the central difference formula is used to approximate the partial derivatives. By keeping the optimal hyperparameter configuration found for the Rastrigin function, meaning $\gamma = 2.5, |\sigma| = \sqrt{2\gamma}, t < t_0, T = 10$, and $h = 0.01$, with the exception of the number of particles N being increased, since the search domain is much larger for the Eggholder, now spanning between $[-512, 512]$ in both x_1 and x_2 direction. With this configuration a plot is generated showing how the particles behave following the second order Kalman-Langevin SDE as their equations of motion. The plot can be seen in Figure 4.4.

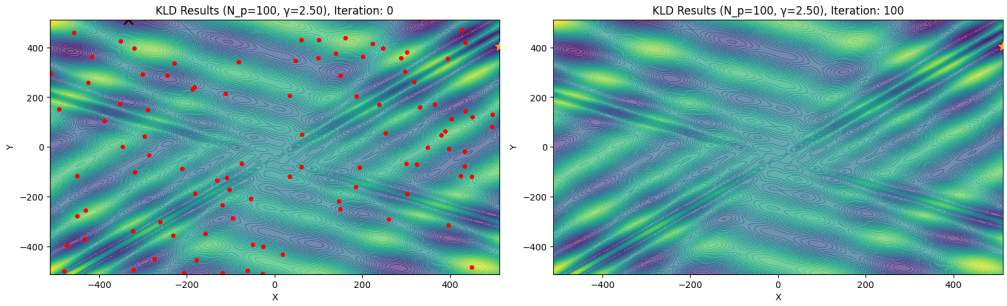


Figure 4.4: Plot showing the $N = 50$ Kalman-Langevin particles' behavior during a single run of $T = 10, h = 0.01, \gamma = 1.75, |\sigma| = \sqrt{2\gamma}$. Each red dot is a particle, the yellow star is the global minimum and the black cross is the particle that currently has the lowest corresponding function value.

With this setup, it was observed that all particles rapidly disappeared from the feasible domain within the first 100 iterations. This issue arises because, in the much larger Eggholder domain, the interaction terms in the covariance and Q matrices can accelerate particles to excessively high velocities. When the velocity is high enough, a particle is able to cross the entire search domain in a single time step. As a result, the reflection mechanism, fails since after reflection, the particle may cross the entire search domain to the opposite boundary, and thus escapes the feasible region entirely. This is shown to be an accurate analysis since behavior was not present when particles were initialized in a smaller region of $x_i \in [-10, 10]$.

A different solution that still allows the method to be used to explore the entire search domain whilst keeping the velocities under control is to introduce a velocity cap via clipping. By setting the maximum velocity to $v_i \in [-100, 100]$ it guarantees that no particle can traverse 1024 units in a single timestep and therfore can no longer escape. This allows us to again initialize the particles in the entire search domain, which is seen in Figure 4.5.

The method is again evaluated by taking a series of runs and seeing how many of

these resulted in the global minimum being found by both the interacting Kalman-Langevin method and the Langevin method. Due to computational costs the number of runs is reduced to 100 from 1000 and a N dependence was tested where the number of particles varies as $N = 10, 20, 50, 100, 200$. the method gets success rate for different number of particles as seen in Table 4.5.

Table 4.5: Success rates for 100 runs investigating different number of particles N for second order Kalman-Langevin SDE on Eggholder. The total runtime was $T = 10$ on for different numbers of particles ($N = 10, 20, 50, 100, 200$). Also included is the amount of times a particle was reflected at the boundary of the search domain for the last run.

N	10	20	50	100	200
Success Rate	0%	0%	0%	0%	0%
Reflections	1959	10	3059	10110	14479

When not allowing for interaction and performing the same experiment again we obtain the resulting success rates for the Langevin process seen in Table 4.6.

Table 4.6: Success rates for 100 runs investigating different number of particles N for non interacting second order Kalman-Langevin SDE on Eggholder. The total runtime was $T = 10$ on for different numbers of particles ($N = 10, 20, 50, 100, 200$). Also included is the amount of times a particle was reflected at the boundary of the search domain for the last run.

N	10	20	50	100	200
Success Rate	0%	0%	0%	0%	0%
Reflections	2	52	102	35	88

The runs show that neither the particles for the Kalman-Langevin nor the Langevin methods are able to find the global minimum regardless of the number of particles searching the space. This was investigated by plotting several runs and inspecting. What seems to be happening can be seen in Figure 4.5. By looking closely one can see a red dot and a black cross where the global minimum is in the bottom rightmost plot, indicating a successful run. However, this counts as a failure since the particle coordinates are: (511.71, 409.85) and this is more than 0.2 Euclidean distance from the actual global minimum at $(512, 404.2319)$. While not every run results in a particle approaching the global minimum, when this does occur, the particle typically becomes stuck very close to the optimum without ever reaching it. This is not the expected behavior and the cause of this is unknown and would have to be investigated more thoroughly to draw any definitive conclusion on what is happening.

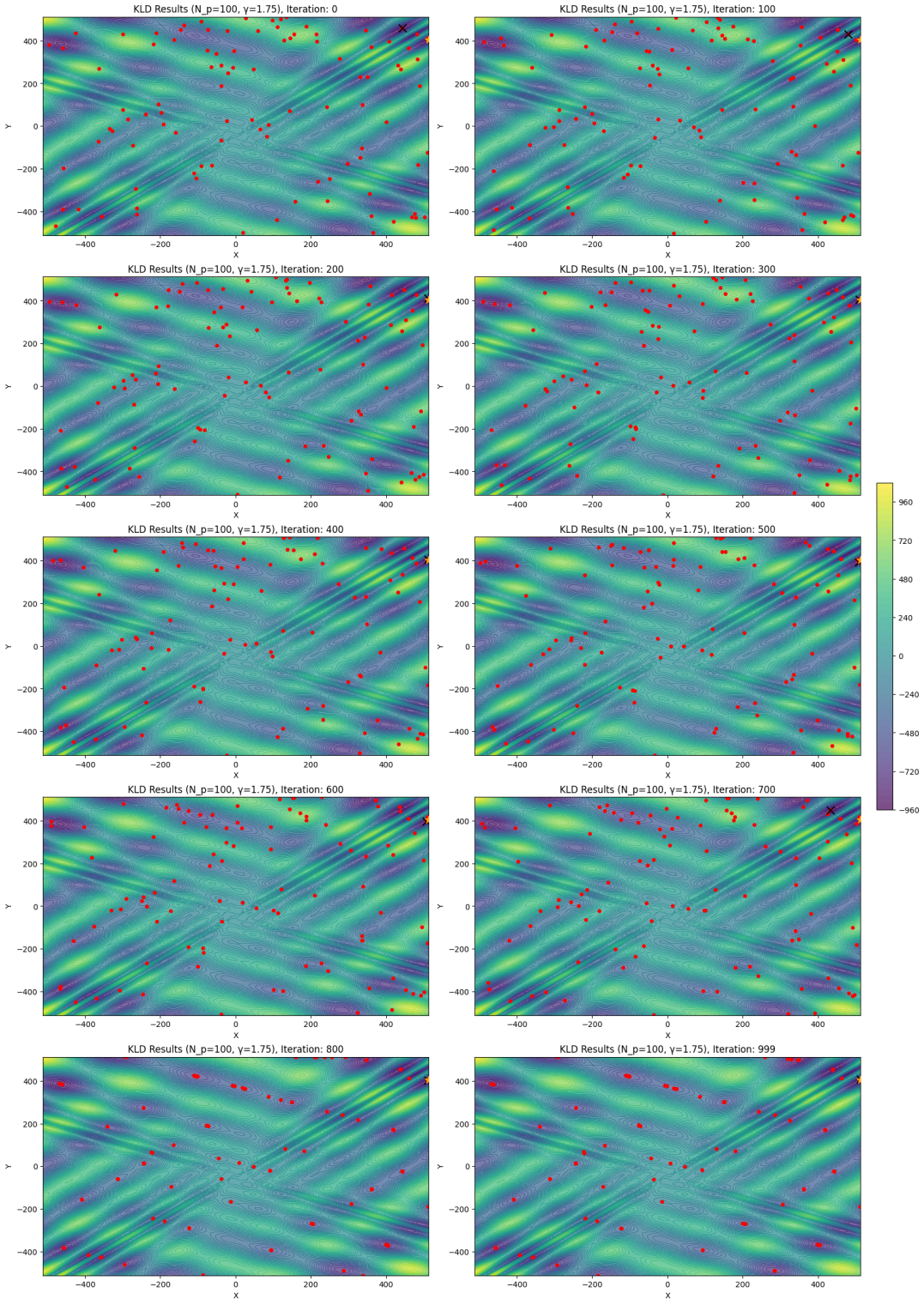


Figure 4.5: Plot showing the $N = 100$ Kalman-Langevin particles' behavior during a single run of $T = 10, h = 0.01, \gamma = 1.75, |\sigma| = \sqrt{2\gamma}$. Each red dot is a particle, the yellow star is the global minimum and the black cross is the particle that currently has the lowest corresponding function value. Here the velocity has been capped at 100 in each direction and all particles are initialized uniformly in a square $[-512, 512]$.

5

Application of PSO and Kalman-Langevin for finding optimal placement of single Au-atom on Au-surface

While previous chapters have focused on evaluating optimization algorithms using test functions, real-world problems often present additional challenges and complexities. This chapter addresses such a challenge by applying particle swarm optimization and a constrained second-order Kalman-Langevin SDE to the problem of locating the optimal (meaning the one which minimizes the potential energy) position for a single gold atom atop a gold surface.

5.1 Problem description

The Au atom placed on top of the surface is referred to as an ad-atom. A visualization of the ad-atoms location relative to the gold surface can be seen in Figure 5.1.

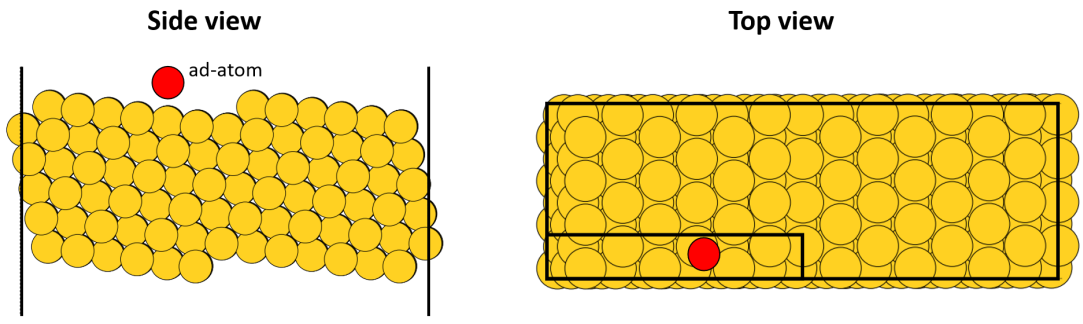


Figure 5.1: Image visualizing the placement of the extra Au-atom on top of the Au-surface. Designation and affiliation of Paul Erhart. Used here with permission.

The left image shows the surface from a side view with the ad-atom colored in red.

The right image shows surface from a top view (z-direction). Here the primitive cell is marked as the smaller rectangle with black edges.

Finding the position which minimizes the energy of the ad-atom is a difficult problem due to the several local minima in the PES. Another complication comes from the fact that density functional theory (DFT) is used to evaluate the potential energy, resulting in expensive computations.

We only need to consider positions inside the primitive cell when searching for the optimal location of the ad-atom since the PES repeats itself outside the primitive cell. We therefore only consider positions in the domain

$$0 < x < 16.65653 \quad \text{and} \quad 0 < y < 2.884996. \tag{5.1}$$

5.1.1 Evaluating the potential energy surface

The ad-atom energy of interest, E, is defined as

$$E = E_{\text{ad}} - E_{\text{surface}}, \tag{5.2}$$

where E_{ad} is the potential energy of the system with the ad-atom present and E_{surface} the energy of just the bare surface system. When evaluating the PES, we consider the surface to be completely rigid and only allow the ad-atom to move. The energy can thus be written as

$$E = E(x, y, z), \tag{5.3}$$

where (x, y, z) corresponds to the position of the ad-atom. Here, we will allow the z-coordinate of the ad-atom to relax to its local minimum (given x and y) and thus only consider the $x - y$ dependency of the energy, meaning we write the energy E as

$$E(x, y) = \min_z E(x, y, z). \tag{5.4}$$

In order to optimize the application of the PSO and Kalman-Langevin methods on a DFT-derived potential energy surface, we precomputed the energy values over the entire search domain with a spatial resolution of $\Delta x = 0.015$ and stored them as a matrix. During optimization, the PSO algorithm accessed this matrix to obtain energy values for particle positions, eliminating the need for repeated, computationally expensive DFT calculations.

While this approach circumvents the original challenge of optimizing with expensive function calls, since the entire function is now available and could be minimized by direct search, it allows for more efficient testing and analysis of the methods performance on a landscape that has ties to reality and that is not constructed with the purpose of being a benchmark function.

After computing the PES it can be visualized with a heatmap seen in Figure 5.2.

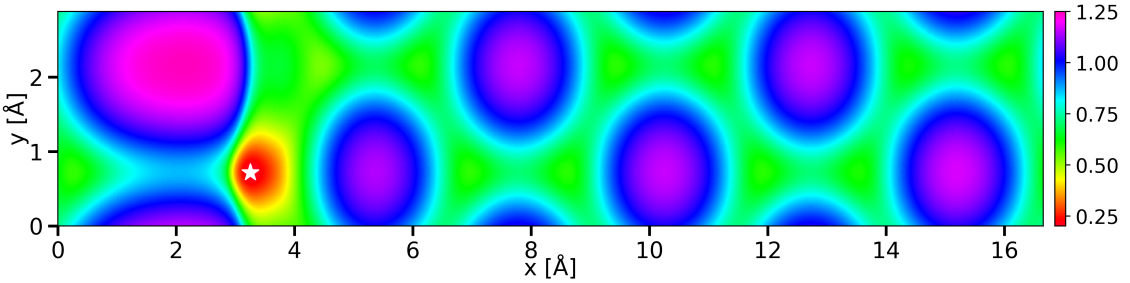


Figure 5.2: Plot visualizing the potential energy surface with the global minimum marked with a white star.

In Figure 5.2, the white star located at $x \approx 3.25, y \approx 0.72$ represents the global minimum. The pink areas are identified as local maxima and the green are flat areas with local minima in between the hills. By taking the logarithm of the PES and plotting it in Figure 5.3 one can more easily see the local minima. One can

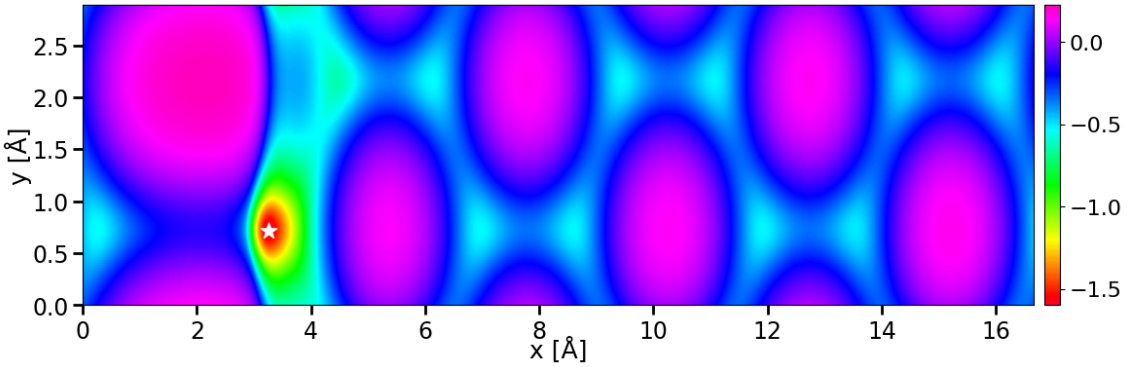


Figure 5.3: Plot visualizing the logarithm of the potential energy surface with the global minimum marked with a white star.

see here how, for example, a gradient based optimization method might struggle at finding the global minimum in this domain due to the many stationary points it can get stuck in.

5.2 Application of PSO and simulated annealing

The PSO method defined by equation (3.9) will be used to explore and search the potential energy surface (PES) for this system. The performance of the PSO method will be compared to simulated annealing (SA), which is a gradient based optimization method combined with an exploration term to allow for the escape from local minima. The equations of motion for simulated annealing are [22] [23] [24]

$$X_{k+1} = X_k - \nabla U(X_k)h + \sqrt{\frac{2}{\ln(1+hk)}}\xi_{k+1}\sqrt{h}, \tag{5.5}$$

where $U(x_k)$ is the potential energy of the surface at position x at iteration $k = 0, ..., n-1$, $n = \frac{T}{h}$, where h is the step size and $\xi_{k+1} \sim \mathcal{N}(0, I_d)$.

The simulated annealing particles are constrained via clipping the positions at the boundaries and setting the particles' position there instead of the reflection that is used for PSO.

All experiments with simulated annealing and PSO in this chapter are run for a total runtime $T = 10$, with a step size of $h = 0.01$ and the number of particles vary between 10 and 200. The methods are evaluated via success rate where the success criterion is having a particle be within a Euclidean distance of 0.2 from the global minimum at the final iteration and the rate is the share of successes from 1000 runs.

To ensure a fair and meaningful comparison between simulated annealing and PSO, particles for both algorithms are deliberately initialized away from the region where the global minimum is known to be located. This setup forces both methods to explore the search space rather than having the success chance be influenced by starting near the optimum and being guided instantly. By enforcing this condition, the comparison of the the algorithms' inherent ability to navigate the energy landscape is made easier. The allowed domain for initialization is $6 < x < 16.65653$, and $0 < y < 2.884996$. The behavior of the simulated annealing with 50 particles and 100 particles can be seen in top and bottom plots respectively in Figure 5.4.

In Figure 5.4 the black dots are the starting positions of the simulated annealing particles, we can see that there are no particles for x values between 0 and 6. The white dots are the final positions and the dashed lines show which final positions correspond to which starting positions. In the second plot in Figure 5.4 the number of particles has been increased to 100 and only the final positions are shown.

To see the behavior of the PSO algorithm on the PES a plot was generated for a successful run where the global minimum was found can be seen in Figure 5.5.

The PSO method appears to have no problem traversing the hills nor the green flat patches. The particles move somewhat together and explores the search domain. After iteration 700, corresponding to $t = 7$, the noise amplitude $|\sigma|$ decreases to 10^{-5} and the particles collapse to the best particle, which in this case is the global minimum.

In Figure 5.6 a run where the particles failed to find the global minimum is shown.

In Figure 5.6 the particles' exploration term never put any of the particles close enough to the global minimum for the best particle to drag the other particles in that direction. These two illustrations of a successful run and failed run indicate the model is quite sensitive to the initial direction.

To properly evaluate and compare the performance of PSO with simulated annealing, more data was gathered. By running both methods for 1000 iterations and calculating the success rate as the proportion of runs where the best particle was within a

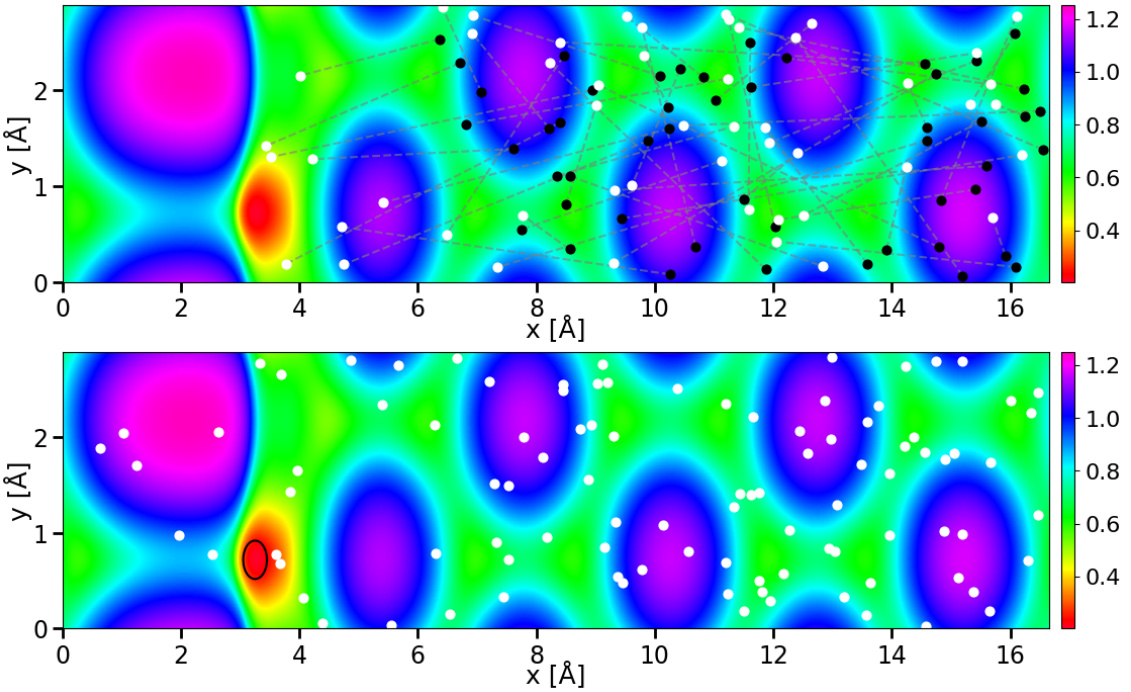


Figure 5.4: The first plot shows with black dots where each of the 50 simulated annealing particle is initialized and final positions with black dots and the second plot shows the final positions for a different run of 100 simulated annealing particles. Both plots are for a total runtime of $T = 10$ with a step-size of $h = 0.01$. The area of Euclidean distance less than 0.2 from the global minimum is shown in the bottom plot with a black circle.

Euclidean distance of 0.2 from the global minimum (located at $x \approx 3.25, y \approx 0.72$), the success rates were 6.1% for simulated annealing and 42.4% for PSO. Here the total runtime was $T = 10$, step-size was $h = 0.01$ and the number of particles was $N_p = 50$. From a search like this the strength of the PSO method is really showcased as it is a whole 7 times likelier to find the global minimum than simulated annealing.

It is also of interest to see how both methods rely on the number of particles to find the global minimum. Therefore the same search of 1000 runs each was conducted where N varies as $N = 10, 20, 50, 100, 200$. The results of this search can be seen in Table 5.1.

Table 5.1: Success rates of PSO and SA methods on PES for different numbers of particles ($N = 10, 20, 50, 100, 200$). The total runtime was $T = 10$ with step-size $h = 0.01$ and the hyperparameter configuration used was $\lambda = 10, \gamma = 1.75$. The magnitude of the noise term was $|\sigma| = 5, t < t_0$ and $|\sigma| = 10^{-5}, t > t_0$.

N	10	20	50	100	200
PSO	29.4%	34.5%	45.1%	63.8%	81.3%
SA	1.1%	1.3%	5.4%	10.9%	20.2%

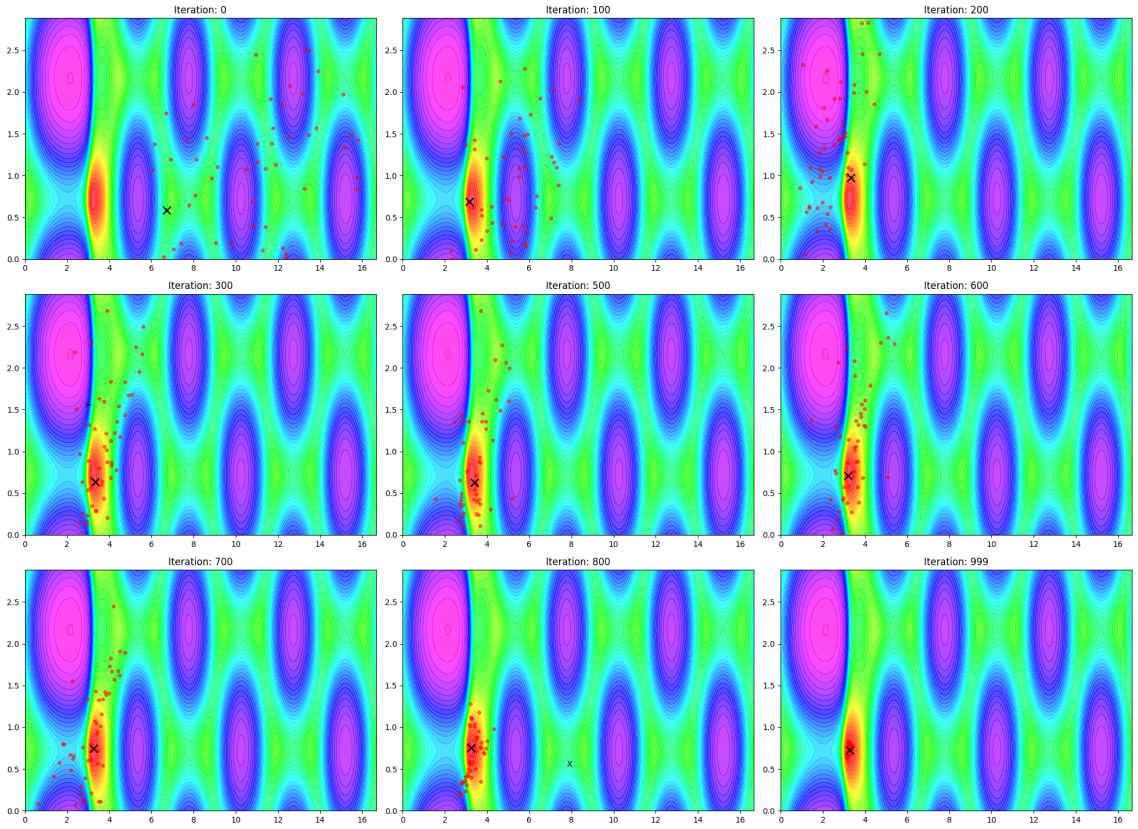


Figure 5.5: Plot showing the movement of the particles during a single run from initialization to final positions. The iterations plotted are $0, 100, 200, 300, 500, 600, 700, 800, 999$.

The superiority of the PSO algorithm is very clear in these results. Simulated annealing requires 200 particles to perform at the same level as 10 PSO particles do.

These runs took 5 minutes and 7 seconds for simulated annealing to complete and PSO took 8 minutes and 44 seconds. From doing these two runs and comparing the runtime it is noted that simulated annealing is around 70% faster than PSO. This raises question of whether the performance disparity between the two methods could be remedied at all by instead comparing wall clock time. Instead of running for a set total time $T = 10$ with a time step $h = 0.01$ and comparing the two methods, simulated annealing will take advantage of its faster calculations and can run for more iterations when governed by the wall clock time. The results of such an investigation can be seen in Table 5.2.

Table 5.2: Success rates of simulated annealing adjusted for wall clock time so running for $T = 17$ on PES for different numbers of particles ($N = 10, 20, 50, 100, 200$).

N	10	20	50	100	200
Success Rate	1.2%	2.5%	6.5%	12.9%	23.7%

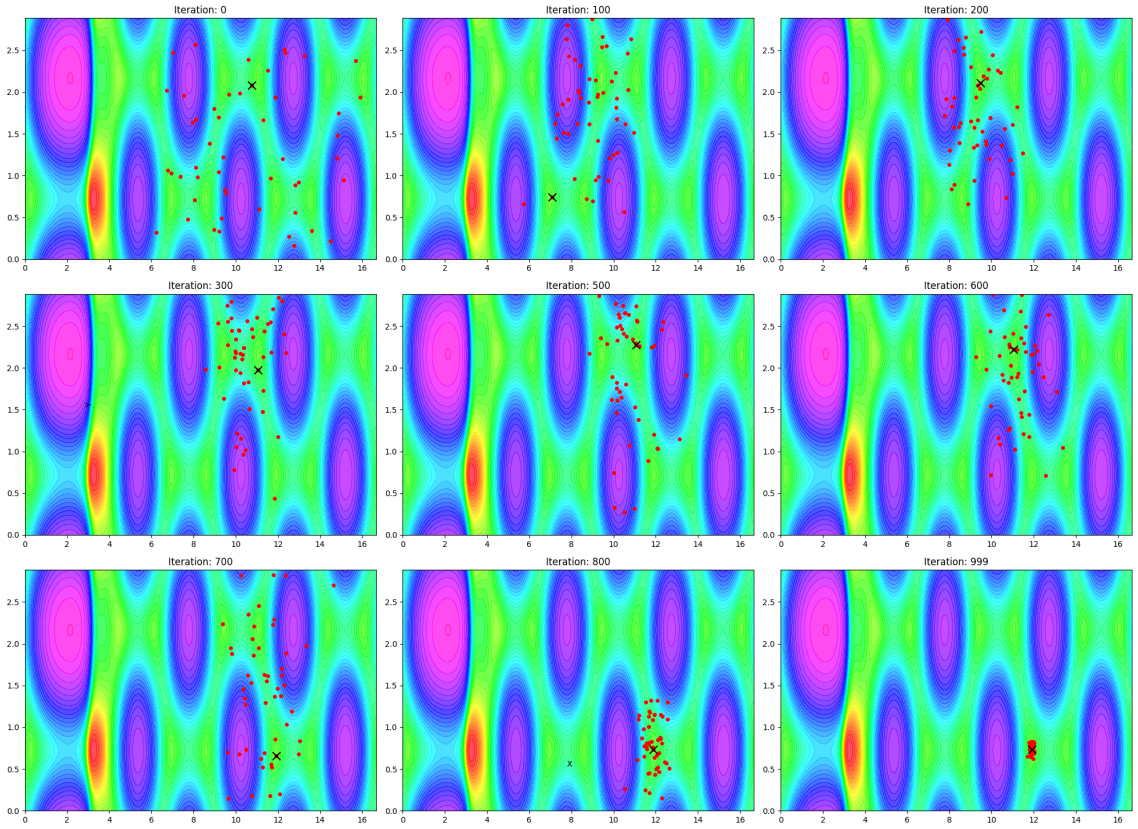


Figure 5.6: Plot showing the movement of the particles during a single run from initialization to final positions. The iterations plotted are $0, 100, 200, 300, 500, 600, 700, 800, 999$.

A slight increase in performance is noted. However, accounting for wall clock time is not enough for simulated annealing to reach a performance similar to PSO.

5.3 Application of Kalman-Langevin

The gradient dependence of Kalman-Langevin is problematic for application on this problem where the analytical expression for the gradient of the objective function is unknown. Therefore, the gradient is instead computed numerically using the `numpy.gradient` function. This method uses second-order accurate central differences in the interior and one-sided differences at the boundaries. The use of this function requires precomputing the objective function over the entire search domain, which was done in this thesis to improve efficiency.

Using the same methodology as we did when testing second order Kalman-Langevin SDE for constrained optimization of the Rastrigin function, meaning governing the particles' movement with equation (4.1) with $\gamma = 2.5$ and $|\sigma| = \sqrt{2\gamma}, \quad t < t_0$, we will now investigate how Kalman-Langevin handles finding the optimal position of

the Au-atom on the Au-surface. The gradient is calculated with `numpy.gradient`, utilizing that the entire surface has been precomputed. This is a disadvantage of this method on this real world problem as the two other methods only need to evaluate the function once in the current point unlike if one would use e.g. finite differences to obtain the gradient.

Just as for PSO the particles will be initialized away from the global minimum, more specifically the lowest possible x-value at initialization is 6.

We begin by looking at an image where the behavior of the particles can be seen via plotting at each hundredth iteration of a single run with the exception of iteration 900. This image can be seen in Figure 5.7.

The particles can be seen exploring the entire space well and then after iteration 700, which corresponds to $t = t_0$ and thus lowering of the noise magnitude to $|\sigma| = 10^{-5}$, the particles collapse to the closest stable point. A better evaluation of the performance can be obtained by increasing the sample size and doing a more extensive search to see how Kalman-Langevin depends on the number of particles present. However, because of the increased computation cost that comes with evaluating the gradient at each step as well as calculating the covariance matrix, only 100 runs were used to calculate the success rate for Kalman-Langevin. The results can be seen in Table 5.3.

Table 5.3: Success rates of second order Kalman-Langevin method on PES for different numbers of particles ($N = 10, 20, 50, 100, 200$). The total number of runs was 100, the total runtime was $T = 10$ and the exploration time was $t_0 = 7$.

N	10	20	50	100	200
Success Rate	55%	76%	96%	100%	100%

This is noticeably better than not only simulated annealing but also PSO, whose results can be seen in Tables 5.1 and 5.2. Just as for the Rastrigin function in Section 4.3 this can be compared with the Langevin method by removing the covariance matrix and the Q matrix from the gradient term and the exploration term respectively. For second order Langevin SDE the results of the success rate search is as follows:

Table 5.4: Success rates of second order Langevin method on PES for different numbers of particles ($N = 10, 20, 50, 100, 200$). The total number of runs was 100, the total runtime was $T = 10$ and the exploration time was $t_0 = 7$.

N	10	20	50	100	200
Success Rate	5%	12%	28%	63%	84%

Which is noticeably worse than both PSO and the interacting Kalman-Langevin method.

In conclusion, both of the methods tested in this thesis perform well on this real world problem of optimizing the location of an Au-atom on an Au-surface. Comparing the two to simulated annealing one sees that there needs to be 200 simulated annealing particles searching the domain to reach the same level of performance as 10 PSO particles. When comparing second order Kalman-Langevin with simulated annealing one sees that 200 simulated annealing particles has less than half as good a chance at finding the global minimum as the Kalman-Langevin method does with 10 particles.

It should be noted that the computational cost depends not only on the number of particles but also on how many points need to be evaluated. For PSO, each iteration evaluates the potential energy only at the current positions of the particles. In contrast, both Kalman-Langevin and simulated annealing require gradient information. If the gradient is not available analytically and must be computed numerically, this means many more function evaluations are needed, since each gradient estimate involves several energy calculations per particle per step. Therefore, while PSO and Kalman-Langevin require fewer particles to achieve higher success rates, the total computational effort may still be higher for these methods.

This was bypassed in this thesis by precomputing all values for the search domain with a set resolution, which made the PSO and Kalman-Langevin methods more viable. The Kalman-Langevin method is the biggest benefactor of the two methods due to it having a gradient dependence while PSO is gradient free. Simulated annealing, also having a gradient dependence, would also suffer from not only having to obtain the energies via DFT calculations, but also the gradient of the potential energy surface.

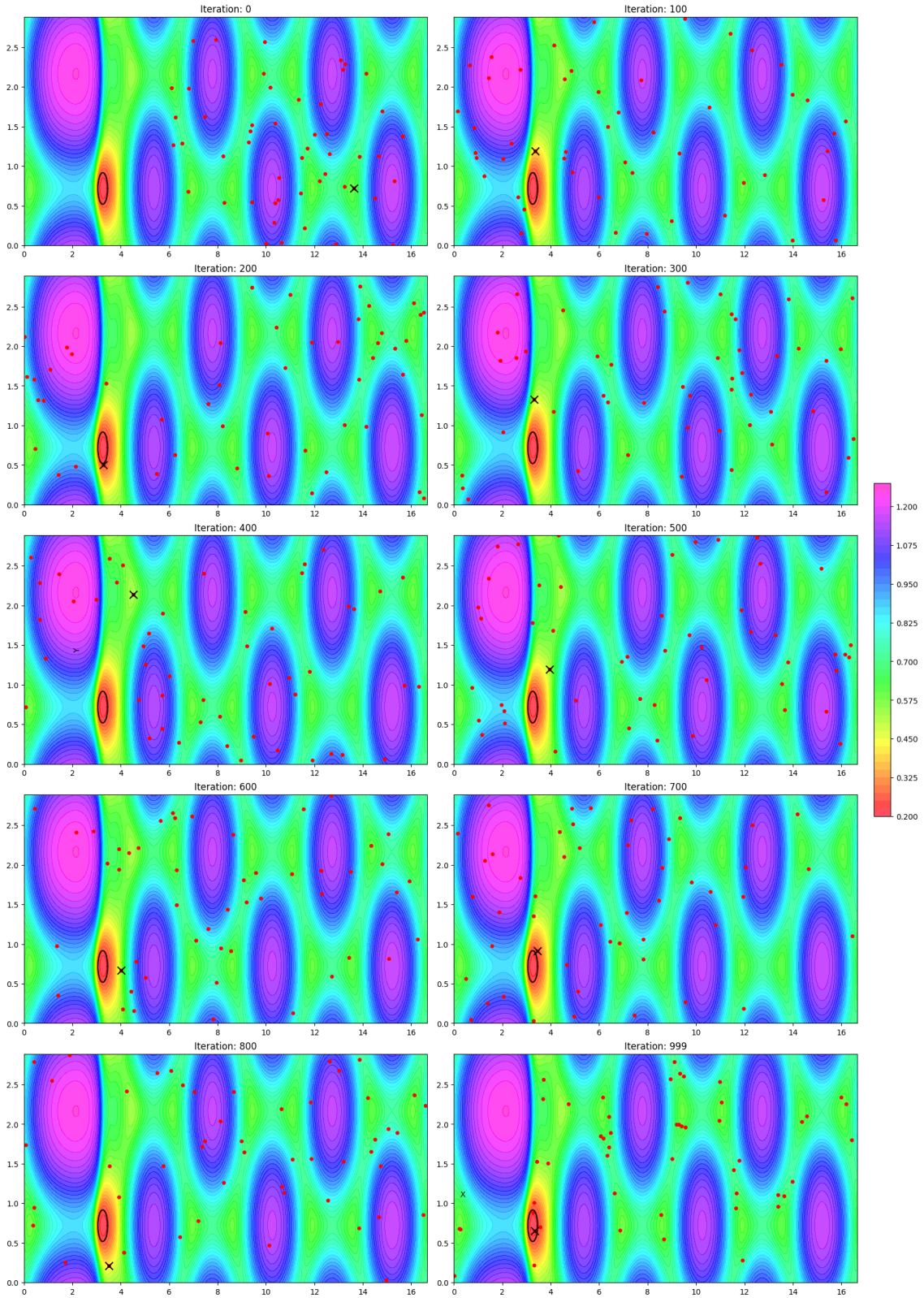


Figure 5.7: Plot showing the $N = 50$ Kalman-Langevin particles' behavior during a single run of $T = 10, h = 0.01, \gamma = 1.75, |\sigma| = \sqrt{2\gamma}$. Each white dot is a particle, the yellow star is the global minimum and the black cross is the particle that currently has the lowest corresponding function value.

6

Conclusion

In this thesis, particle swarm optimization and second order Kalman-Langevin methods driven by SDEs have been tested for constrained optimization of different test functions and also been applied on a real world problem. Both methods have shown themselves to be viable for all of these application areas, with the exception of some unexpected behavior from Kalman-Langevin on the Eggholder function. The results show that both methods are easy to apply and do not require too much objective function specific tuning for the methods to be able to perform at a good level.

For the PSO method it is important to balance the λ variable which governs interaction strength between particles and the γ variable which governs the friction of the particles, which was shown in Table 3.6. The optimal configuration that was found from testing on the Eggholder function also proved to be a good choice when applied on the real world problem. The PSO method has no distance scaling on its noise parameter unlike the Kalman-Langevin method, which means that the σ parameter needs to be manually set to be of the order of the radius of the search domain. A strength of the PSO method is how well it traverses large distances and searches the entire domain when $|\sigma|$ parameter is a reasonable size. PSO performs well on the objective functions it was tested on, this is especially seen in the real world problem, where simulated annealing requires 200 particles to match the performance of 10 PSO particles. PSO being gradient free should also be an advantage for it when it comes to computational cost, however, the extent of this has not been thoroughly tested in this thesis.

The Kalman-Langevin method requires even less tuning, with only two free parameters γ,and σ, of which one is chosen to be proportional to the other for this method, with $\sigma = \sqrt{2\gamma}, t < t_0$. Kalman-Langevin performed well at finding the global minima, and when compared to second order Langevin there is a clear increase in performance for allowing interaction between particles. The Kalman-Langevin method also performed exceptionally well for the real world problem of optimizing the placement of the gold atom. Comparing Kalman-Langevin to simulated annealing one sees that 200 simulated annealing particles fail to reach a performance that is even half as good as just 10 Kalman-Langevin particles. A drawback of the Kalman-Langevin method however, is the need for gradient information, which, for a problem relying on DFT calculations for energy evaluations, can become very expensive.

Because of this the conclusion of this thesis is that both methods work well for the purposes of constrained optimization. The inclusion of constraints when doing the optimization is well motivated by looking at the number of reflections which was included in Tables 3.7, and 4.3, and 4.5. It would be interesting to see how the performance would be impacted by using a different way of enforcing the constraint such as projection or penalty methods. Another interesting thing to investigate is applying the methods in a higher dimensional setting, considering only 2 dimensional problems have been addressed in this thesis.

Bibliography

[1] N. Kovachki and A. Stuart, "Ensemble kalman inversion: a derivative-free technique for machine learning tasks," *Inverse Problems*, vol. 35, p. 095005, 8 2019.

[2] R. Pinnau, C. Totzeck, O. Tse, and S. Martin, "A consensus-based model for global optimization and its mean-field limit," *Mathematical Models and Methods in Applied Sciences*, vol. 27, no. 01, pp. 183–204, 2017.

[3] Z. Liu, A. M. Stuart, and Y. Wang, "Second order ensemble langevin method for sampling and inverse problems," *arXiv preprint arXiv:2208.04506*, 2022.

[4] J. A. Carrillo, C. Totzeck, and U. Vaes, "Consensus-based optimization and ensemble kalman inversion for global optimization problems with constraints," in *Modeling and simulation for collective dynamics*, pp. 195–230, World Scientific, 2023.

[5] D. Kalise, A. Sharma, and M. V. Tretyakov, "Consensus-based optimization via jump-diffusion stochastic differential equations," *Mathematical Models and Methods in Applied Sciences*, vol. 33, no. 02, pp. 289–339, 2023.

[6] V. Molin, A. Ringh, M. Schauer, and A. Sharma, "Controlled stochastic processes for simulated annealing," *arXiv preprint arXiv:2504.08506*, 2025.

[7] A. Ringh and A. Sharma, "Kalman-langevin dynamics: exponential convergence, particle approximation and numerical approximation," *arXiv preprint arXiv:2504.18139*, 2025.

[8] P. D. Hinds, A. Sharma, and M. V. Tretyakov, "Well-posedness and approximation of reflected mckean–vlasov sdes with applications," *Mathematical Models and Methods in Applied Sciences*, vol. 35, no. 08, pp. 1845–1887, 2025.

[9] J. Kennedy and R. Eberhart, "Particle swarm optimization," in *Proceedings of ICNN'95-international conference on neural networks*, vol. 4, pp. 1942–1948, ieee, 1995.

[10] S. Grassi, H. Huang, L. Pareschi, and J. Qiu, "Mean-field particle swarm optimization," in *Modeling and Simulation for Collective Dynamics*, pp. 127–193, World Scientific, 2023.

[11] M. Bossy and J.-F. Jabir, "On confined mckean langevin processes satisfying the mean no-permeability boundary condition," *Stochastic Processes and their Applications*, vol. 121, no. 12, pp. 2751–2775, 2011.

[12] B. Leimkuhler, A. Sharma, and M. V. Tretyakov, "Numerical integrators for confined langevin dynamics," *arXiv preprint arXiv:2404.16584*, 2024.

[13] M. Wilkins, "A brief introduction to numerical methods for constrained optimization." https://people.duke.edu/~ccc14/sta-663-2017/notebooks/S06_Numerical_Optimization.html, 2017. Accessed: 2024-06-10.

[14] H. Fawzi, "Topics in convex optimisation (lent 2022)." https://www.damtp.cam.ac.uk/user/hf323/L22-III-OPT/, 2022. Lecture notes, University of Cambridge.

[15] J. Nocedal and S. J. Wright, *Numerical Optimization*. Springer Series in Operations Research and Financial Engineering, New York: Springer, 2nd ed., 2006.

[16] D. Bingham, "Eggholder function." https://www.sfu.ca/~ssurjano/egg.html, 2013. Retrieved April 1, 2025.

[17] Nschloe, "Contour plot of the eggholder function." https://commons.wikimedia.org/w/index.php?curid=114939892, 2022. Wikimedia Commons. Licensed under CC BY-SA 4.0: https://creativecommons.org/licenses/by-sa/4.0.

[18] Gaortizg, "Cross-in-tray function surface plot." https://upload.wikimedia.org/wikipedia/commons/f/f9/Cross-in-tray_function.pdf, 2012. Wikimedia Commons. Licensed under CC BY-SA 3.0: https://creativecommons.org/licenses/by-sa/3.0.

[19] Nschloe, "Cross-in-tray contour plot." https://upload.wikimedia.org/wikipedia/commons/f/f5/Cross-in-tray_contour.svg, 2022. Wikimedia Commons. Licensed under CC BY-SA 4.0: https://creativecommons.org/licenses/by-sa/4.0.

[20] Diegotorquemada, "Rastrigin function." https://commons.wikimedia.org/wiki/File:Rastrigin_function.png, 2010. Image created by Diegotorquemada. Released into the public domain.

[21] Nschloe, "Rastrigin smooth contour plot." https://commons.wikimedia.org/wiki/File:Rastrigin-smooth-contour.svg, 2022. Image licensed under Creative Commons Attribution-Share Alike 4.0 International. https://creativecommons.org/licenses/by-sa/4.0/.

[22] P. J. Van Laarhoven and E. H. Aarts, *Simulated Annealing: Theory and Applications*. Dordrecht, The Netherlands: Springer, 1987.

[23] S. Kirkpatrick, C. D. Gelatt Jr., and M. P. Vecchi, "Optimization by simulated annealing," *Science*, vol. 220, no. 4598, pp. 671–680, 1983.

[24] T.-S. Chiang, C.-R. Hwang, and S. J. Sheu, "Diffusion for global optimization in $\mathbb{R}^n$," *SIAM Journal on Control and Optimization*, vol. 25, no. 3, pp. 737–753, 1987.

DEPARTMENT OF MATHEMATICAL SCIENCES
CHALMERS UNIVERSITY OF TECHNOLOGY
Gothenburg, Sweden
www.chalmers.se

CHALMERS
UNIVERSITY OF TECHNOLOGY