Parameter Estimation of a Parametric Curve using a Hybrid CMA-ES and L-BFGS Approach

Rahul Prasad(2201MC25)

10-11-2025

Contents

1 Introduction

The objective of this assignment is to find the values of three unknown variables, θ, M, and X, that define a parametric curve, such that the curve best fits a given dataset of (x, y) points.
The curve is defined by the following parametric equations:

$$x(t) = \left(t\cos(\theta) - e^{M|t|}\sin(0.3t)\sin(\theta) + X\right) \tag{1}$$

$$y(t) = \left(42 + t\sin(\theta) + e^{M|t|}\sin(0.3t)\cos(\theta)\right) \tag{2}$$

1.1 Given Constraints

- **Unknown Parameters** ($\mathbf{p} = [\theta, M, X]^T$)**:**
  - $0.0 \leq \theta < 50.0$ (degrees)
  - $-0.05 \leq M < 0.05$
  - $0.0 < X < 100.0$
- **Parameter** t **Range:**

$$6.0 < t < 60.0$$

1.2 Descriptive Statistics of the Dataset

The dataset provided contains $N_{obs} = 1500$ observed points representing the coordinates (x_i, y_i) of the curve to be fitted. To understand the range, spread, and central tendency of the observed data, descriptive statistics were computed for both x and y coordinates.

Table 1: Descriptive statistics of the observed dataset

Statistic	**x**	**y**
Count (N)	1500	1500
Mean (μ)	83.7139	58.2635
Standard Deviation (σ)	13.6972	7.6965
Minimum	59.6572	46.0323
25th Percentile (Q_1)	72.2829	51.1242
Median (Q_2)	84.7110	57.6813
75th Percentile (Q_3)	93.3683	66.1384
Maximum	109.2315	69.6855

The dataset indicates that both x and y values exhibit moderate variation, with x spanning a wider range than y.

2 Mathematical Formulation

2.1 Objective Function

The problem is formulated as an optimization problem to determine the parameters $\mathbf{p} = [\theta, M, X]^T$ that define a parametric curve $\mathbf{C}(\mathbf{p}, t)$ best fitting a given set of observed data points $\mathbf{d}_i = [x_i, y_i]^T$.
The goal is to minimize a scalar objective function $\mathcal{L}(\mathbf{p})$ that measures the discrepancy between the observed data and the predicted curve:

$$\min_{\mathbf{p}} \mathcal{L}(\mathbf{p})$$

The parametric curve is defined as:

$$\mathbf{C}(\mathbf{p}, t) = \begin{bmatrix} x(\mathbf{p}, t) \\ y(\mathbf{p}, t) \end{bmatrix}$$

where $x(\mathbf{p}, t)$ and $y(\mathbf{p}, t)$ are functions governed by the parameters θ, M, and X.

2.2 Loss Function

The implemented loss function, denoted as $\mathcal{L}(\mathbf{p})$, is based on the mean minimum ℓ_1 distance between each observed point $\mathbf{d}_i$ and its closest predicted point along the curve $\mathbf{C}(\mathbf{p}, t_j)$. This ensures that the predicted curve aligns as closely as possible to the observed data.
The loss function is expressed as:

$$\mathcal{L}(\mathbf{p}) = \frac{1}{N_{\text{obs}}} \sum_{i=1}^{N_{\text{obs}}} \left[\min_j \left(|x_i - x(\mathbf{p}, t_j)| + |y_i - y(\mathbf{p}, t_j)| \right) \right]$$

where:

- N_{obs} is the number of observed data points = 1500
- t_j are sampled time points used to generate the predicted curve =8000
- The inner minimum $\min_j$ ensures that, for each observed point, only the closest predicted point contributes to the loss.

This formulation corresponds directly to the implemented function `batched_l1`, which computes the above expression efficiently using GPU-based batching.

2.3 Normalization

To ensure numerical stability and a well-conditioned loss landscape, both the observed and predicted points are normalized before loss computation using Z-score normalization derived from the observed data statistics:

$$\mathbf{d}_{norm} = \frac{\mathbf{d} - \mu_{\text{obs}}}{\sigma_{\text{obs}}} \qquad \text{and} \qquad \mathbf{c}_{norm} = \frac{\mathbf{c} - \mu_{\text{obs}}}{\sigma_{\text{obs}}}$$

where μ_{obs} and σ_{obs} denote the mean and standard deviation of the observed data, respectively.

3 Optimization Methodology

The given parametric functions $\mathbf{C}(\mathbf{p}, t)$ are highly non-linear and non-convex, often characterized by multiple local minima and steep valleys in the loss landscape. Traditional gradient-based methods such as Conjugate Direction or Quasi-Newton (BFGS) can easily become trapped in these local minima, leading to suboptimal convergence. To address this challenge, a two-phase hybrid optimization framework was implemented.
In the first phase, the **Covariance Matrix Adaptation Evolution Strategy (CMA-ES)**, a robust derivative-free global optimizer, is employed to perform an extensive exploration of the parameter space and identify a near-global minimum. The resulting parameters from CMA-ES serve as an excellent initialization for the second phase, where a **Quasi-Newton BFGS-based optimizer (specifically, the L-BFGS variant)** is applied for fine-tuning. This local refinement step leverages gradient information to achieve a more precise convergence and minimize the ℓ_1 loss, yielding an accurate and stable fit to the observed data.

3.1 CMA-ES Global Optimization

Covariance Matrix Adaptation Evolution Strategy (CMA-ES) is a stochastic, **derivative-free** optimization algorithm particularly effective for non-convex and multimodal functions. CMA-ES is used for the **global search** of the three unknown parameters θ, M, and X that best fit the observed (x, y) data. Its robustness comes from three key mechanisms: **normal distribution sampling**, **rank-based recombination**, and **adaptive control of step size and covariance matrix**.

3.1.1 State Variables

- Mean vector: $m_k \in \mathbb{R}^n$
- Step-size: $\sigma_k > 0$

- Covariance matrix: $C_k \in \mathbb{R}^{n \times n}$, symmetric and positive definite
- Evolution paths: $p_\sigma, p_c \in \mathbb{R}^n$

3.1.2 Sampling from Multivariate Normal Distribution

At each generation, CMA-ES samples λ candidate solutions from a multivariate normal distribution:

$$x_k \sim \mathcal{N}(m, \sigma^2 C), \quad k = 1, \ldots, \lambda$$

where m is the mean vector (current estimate of the best parameters), σ is the global step size, and C is the covariance matrix that defines the search shape. Each candidate is evaluated using the loss function $\mathcal{L}(x_k)$, and the top μ candidates with the lowest loss values are selected. The new mean m' is computed using weighted recombination of these μ best candidates:

$$m' = \sum_{i=1}^{\mu} w_i \, x_{i:\lambda}$$

where $x_{i:\lambda}$ is the i-th best candidate and w_i are positive, normalized weights.

3.1.3 Update Evolution Paths

Step-size path:

$$p_\sigma \leftarrow (1 - c_\sigma) p_\sigma + \sqrt{c_\sigma (2 - c_\sigma) \mu_{\text{eff}}} \frac{C_k^{-1/2} (m_{k+1} - m_k)}{\sigma_k}$$

It updates the covariance path by accumulating the normalized mean steps across generations.This helps CMA-ES remember consistent progress directions to adapt the covariance matrix accordingly.

Step-size:

$$\sigma_{k+1} = \sigma_k \times \exp\left(\frac{c_\sigma}{d_\sigma} \left(\frac{\|p_\sigma\|}{E\|\mathcal{N}(0, I)\|} - 1 \right) \right)$$

This allows the algorithm to take larger steps when progress is steady and smaller steps when it approaches an optimum, ensuring a balance between exploration and exploitation.

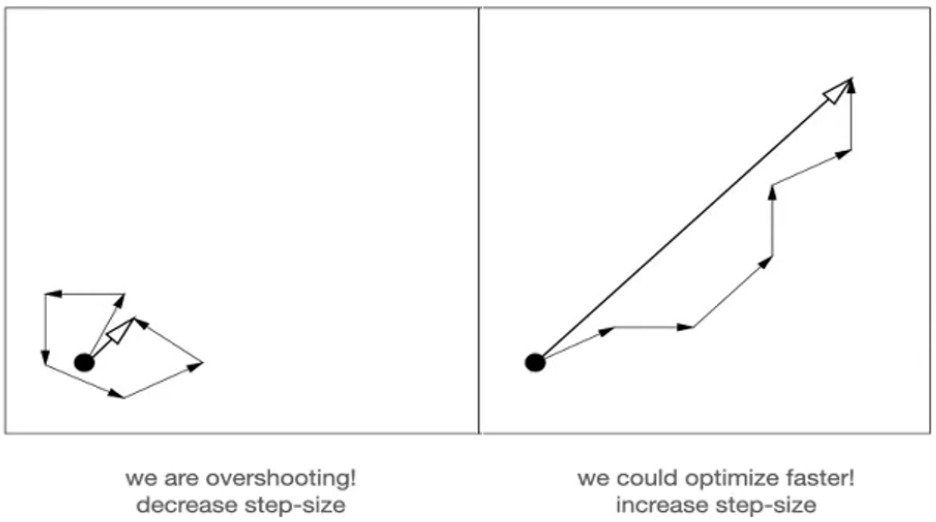


Figure 1: Length of the evolution paths (sum of steps) is remarkably different and is exploited for step-size control

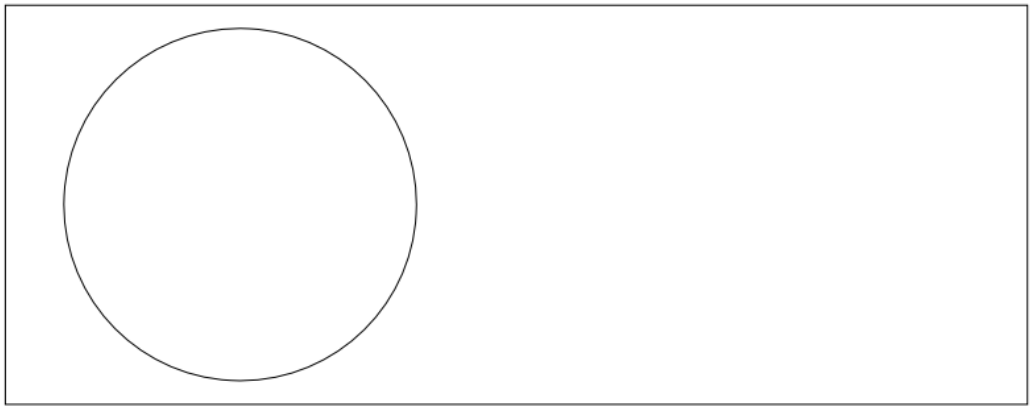
3.1.4 Update Covariance Matrix

Covariance path:

$$p_c \leftarrow (1 - c_c)p_c + h_\sigma \sqrt{c_c(2 - c_c)\mu_{\text{eff}}} \frac{m_{k+1} - m_k}{\sigma_k}$$

$$C_{k+1} = (1 - c_1 - c_\mu)C_k + c_1\, p_c p_c^T + c_\mu \sum_{i=1}^{\mu} w_i \frac{(x_{i:\lambda} - m_k)(x_{i:\lambda} - m_k)^T}{\sigma_k^2}$$

This adaptation allows CMA-ES to "learn" the contour of the objective function, efficiently navigating curved or correlated search spaces without requiring gradient information.



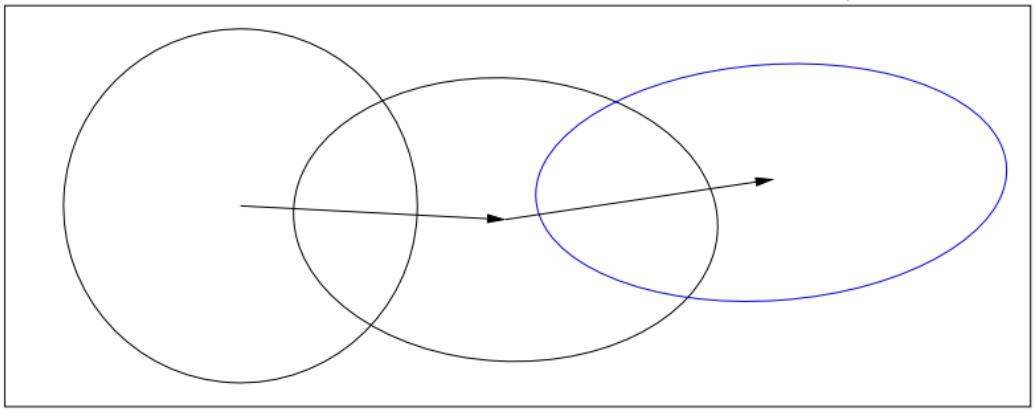


Figure 2: Schematic diagram for the adaptation of contours of Covariance Matrix

3.1.5 Population Parameters

The default settings of CMA-ES automatically determine λ and μ based on the dimensionality of the search space (n parameters):

$$\lambda = 4 + \lfloor 3 \ln(n) \rfloor, \quad \mu = \frac{\lambda}{2}$$

For this problem ($n = 3$):

$$\lambda = 7, \quad \mu = 3$$

Hence, at each iteration, the algorithm samples 7 candidate parameter sets and uses the 3 best ones to update the mean and covariance.

3.1.6 Initialization Parameters

The CMA-ES is initialized with:

$$x_0 = [25.0, 0.0, 50.0]^T, \quad \sigma_0 = [5.0, 0.01, 10.0]^T$$

where σ_0 represents the initial standard deviations of each parameter, roughly 10% of their range. This allows a balanced initial exploration of the parameter space. The initial covariance matrix is set to the identity matrix:

$$C_0 = I$$

which assumes that the parameters are uncorrelated at the start and have unit variance in their normalized space.

3.1.7 Pseudo Code

Algorithm 1 CMA-ES Optimizer

Input: fitness function f, initial step size $\sigma^{(0)}$, initial mean value $\mathbf{m}^{(0)}$, population size λ, parent number μ
Output: the optimal solution: $\mathbf{x}^{opt}$ and its response f^{opt}

initialize the evolution path $\mathbf{p}_\sigma^{(0)} = \mathbf{0}$, $\mathbf{p}_c^{(0)} = \mathbf{0}$, covariance matrix $\mathbf{C}^{(0)} = \mathbf{I}$
for $g = 0, 1, 2, \ldots$ until the termination criterion is satisfied **do**
  sample a new population $\mathbf{x}_k$ $(k = 1, 2, \ldots, \lambda)$ by Eq.(1)
  evaluate and sort $\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_\lambda$ by fitness function f
  update $\mathbf{m}^{(g+1)}$
  update $\mathbf{p}_\sigma^{(g+1)}$
  update $\sigma^{(g+1)}$
  update $\mathbf{p}_c^{(g+1)}$
  update $\mathbf{C}^{(g+1)}$
end for
sort all the evaluated sample points by their responses to find $\mathbf{x}^{opt}$ and f^{opt}
return $\mathbf{x}^{opt}$ and f^{opt}

After a fixed number of iterations, the best parameters are selected as:

$$[\theta^*, M^*, X^*] = \arg\min_{\theta, M, X} \mathcal{L}(\theta, M, X)$$

These optimized parameters serve as the starting point for the **LBFGS local refinement stage**.

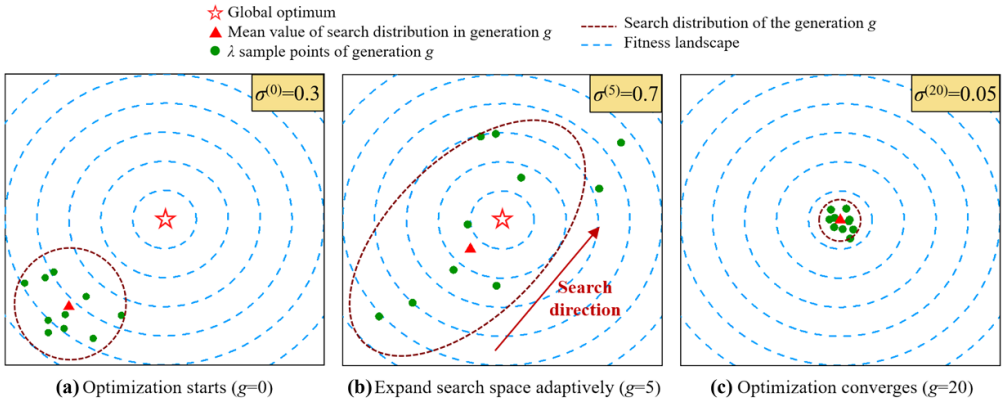


Figure 3: Schematic diagram of the optimization process of CMA-ES

3.1.8 Optimized Parameters

The global parameters found by the CMA-ES are.

Table 2: Optimized Parameter Values

Parameter	Optimized Value
θ	30.0002
M	0.030000
X	55.0021

- **Number of Iterations: = 50**
- **Final Mean ℓ_1 Distance: $\mathcal{L}(\mathbf{p}_{opt}) = 0.003820$**

3.2 Quasi-Newton BFGS Local Optimization

The Broyden–Fletcher–Goldfarb–Shanno (BFGS) algorithm is a **quasi-Newton method** used for local optimization of differentiable functions. Unlike CMA-ES, which performs global search, BFGS refines the parameters by exploiting gradient information near the optimum. It provides rapid convergence once the solution is close to the true minimum.

3.2.1 Optimization Principle

BFGS approximates the inverse of the Hessian matrix using successive gradient evaluations to guide the search toward a local minimum. At each iteration k, the search direction is computed as:

$$d^{(k)} = -H_k\, g^{(k)}$$

and the parameters are updated using a line search along this direction:

$$x^{(k+1)} = x^{(k)} + \alpha_k d^{(k)}$$

where the step size α_k is obtained by solving:

$$\alpha_k = \arg\min_{\alpha \geq 0} f\big(x^{(k)} + \alpha d^{(k)}\big)$$

Here, H_k denotes the current approximation of the inverse Hessian matrix, $g^{(k)} = \nabla f(x^{(k)})$ is the gradient, and α_k ensures sufficient decrease in the objective function via a line search (typically satisfying the Wolfe conditions).

3.2.2 Hessian Update Rule

The update for H_k (inverse Hessian approximation) is given by:

$$H_{k+1} = \left(I - \frac{s_k y_k^T}{y_k^T s_k}\right) H_k \left(I - \frac{y_k s_k^T}{y_k^T s_k}\right) + \frac{s_k s_k^T}{y_k^T s_k}$$

where

$$s_k = x_{k+1} - x_k, \quad y_k = \nabla f(x_{k+1}) - \nabla f(x_k)$$

This update maintains H_k as a symmetric and positive definite matrix, ensuring that the search direction remains a descent direction.

3.2.3 Armijo-Wolfe Line Search Conditions

To ensure a proper step size α_k that maintains the descent direction, the **Armijo–Wolfe conditions** are applied during line search.

1. Armijo (Sufficient Decrease) Condition:

$$f(x_k + \alpha_k d_k) \leq f(x_k) + c_1 \alpha_k \, g_k^T d_k$$

This ensures that the function value decreases sufficiently along the search direction.

2. Wolfe (Curvature) Condition:

$$\nabla f(x_k + \alpha_k d_k)^T d_k \geq c_2 \, g_k^T d_k$$

This guarantees that the new slope is not too steep, maintaining the descent property.
Typical values are $c_1 = 10^{-4}$ and $c_2 = 0.9$. Together, these conditions ensure both adequate function decrease and stable convergence of the BFGS update.

3.2.4 Initialization

BFGS is initialized using the best parameters obtained from the CMA-ES stage:

$$x_0 = [\theta^*, M^*, X^*]^T$$

The initial inverse Hessian approximation is set to the identity matrix:

$$H_0 = I$$

and iterations continue until the gradient norm falls below a given tolerance ϵ.

3.2.5 Pseudo Code

Algorithm 2 Quassi-Newton BFGS

Input: differentiable loss function f, initial parameters $x^{(0)}$, tolerance ϵ
Output: locally optimized parameters x^{opt}
initialize $H^{(0)} = I$
for $k = 0, 1, 2, \ldots$ until $\|\nabla f(x^{(k)})\| < \epsilon$ **do**
  compute gradient $g_k = \nabla f(x^{(k)})$
  compute search direction $p_k = -H_k g_k$
  perform line search to find α_k satisfying **Wolfe conditions**
  update parameters: $x^{(k+1)} = x^{(k)} + \alpha_k p_k$
  compute $s_k = x^{(k+1)} - x^{(k)}$
  compute $y_k = \nabla f(x^{(k+1)}) - \nabla f(x^{(k)})$
  update H_{k+1} using:

$$H_{k+1} = \left(I - \frac{s_k y_k^T}{y_k^T s_k}\right) H_k \left(I - \frac{y_k s_k^T}{y_k^T s_k}\right) + \frac{s_k s_k^T}{y_k^T s_k}$$

end for
return $x^{opt} = x^{(k)}$

3.2.6 L-BFGS PyTorch Package

The Limited-memory BFGS (L-BFGS) algorithm is an efficient variant of BFGS designed for large-scale optimization problems. Unlike BFGS, which explicitly stores and updates the full approximate Hessian matrix $H_k \in \mathbb{R}^{n \times n}$, L-BFGS maintains only a limited number of correction pairs (s_i, y_i), where:

$$s_i = x_{i+1} - x_i, \quad y_i = g_{i+1} - g_i$$

These pairs represent recent changes in position and gradient, respectively, allowing L-BFGS to implicitly approximate H_k without the heavy memory and computational cost of storing the entire matrix.

Key Advantages:

- **Memory Efficiency:** Stores only a few past updates (typically 10–20), making it suitable for high-dimensional problems.
- **Scalability:** Reduces storage from $\mathcal{O}(n^2)$ in BFGS to $\mathcal{O}(mn)$, where m is the memory size.
- **Loss Computation:** Utilizes PyTorch's `autograd` mechanism for automatic differentiation, enabling efficient backpropagation even for complex or non-smooth loss functions.

4 Results

The hybrid optimization framework converged successfully, yielding optimal parameter values with up to 32-bit precision.

4.1 Optimized Parameters

The final parameters found by the L-BFGS are

Table 3: Optimized Parameter Values

Parameter	Optimized Value
θ	29.99962997
M	0.029999520630
X	55.00122833

4.2 Final Loss and Performance

- **Total LBFGS Iteration 200**
- **Final Mean ℓ_1 Distance: $\mathcal{L}(\mathbf{p}_{opt}) = 0.00025639$**
- **Total Optimization Time: 0.67 seconds**

4.3 Curve Fitting Visualization

The quality of the fit is illustrated in Figure 4, which plots the original data points against the curve generated by the optimized parameters.

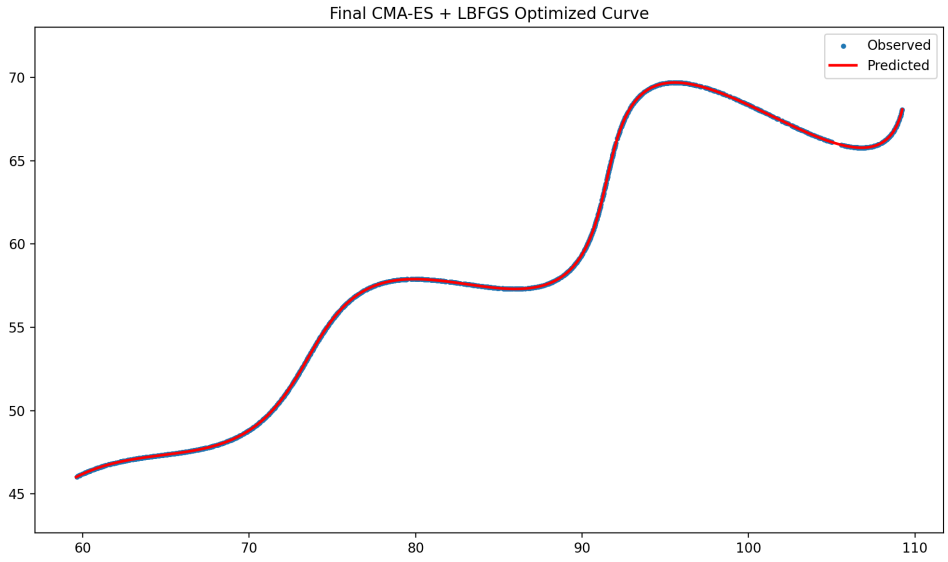


Figure 4: Observed data points (blue) versus the optimized parametric curve (red)

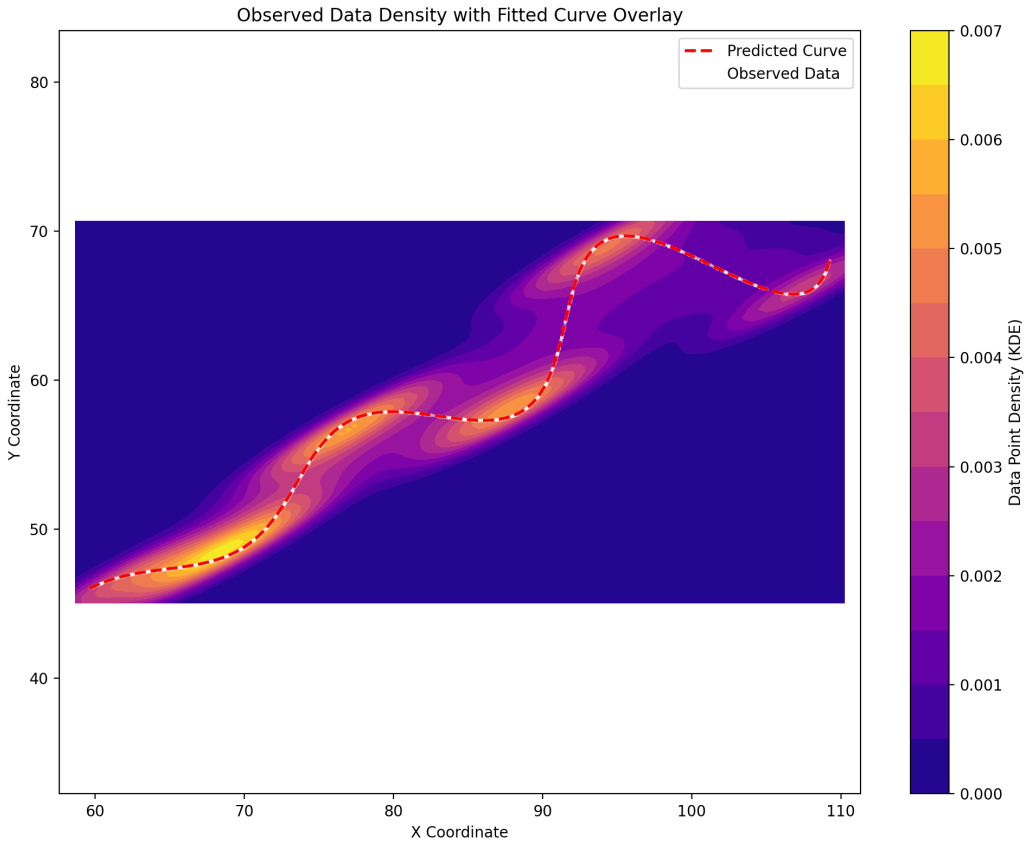


Figure 5: Observed Data Density Contour Plot

5 Optimization Time Comparison

The optimization time primarily depends on the number of iterations performed by the CMA-ES and L-BFGS optimizers in finding the global minimum. CMA-ES is computationally more expensive than L-BFGS due to its population-based search and covariance adaptation mechanism. Therefore, the total optimization time is heavily influenced by the number of CMA-ES iterations.

Running more iterations of CMA-ES generally increases the likelihood of converging to the exact global minimum, although at the cost of longer computation time. In contrast, L-BFGS is a much faster gradient-based method, and hence its number of iterations is fixed to 200 for fair comparison.

The following table presents the comparison of optimization time and final loss obtained for different iteration counts of CMA-ES, with L-BFGS iterations held constant.

Table 4: Comparison of Optimization Time for CMA-ES Optimizer

Iterations	**CMA-ES Time (s)**	**Total Time (s)**	**CMA-ES Loss**	**Final Loss**
10	0.10	0.27	0.024161	0.019783
30	0.34	0.55	0.003974	0.000701
50	0.57	0.71	0.003821	0.000272
70	0.92	1.02	0.003820	0.000256
100	1.23	1.34	0.003820	0.000261

From the results, it can be observed that the CMA-ES optimizer rapidly reduces the loss within the first 50 iterations, achieving a near-global minimum with a significant drop in loss value from 0.024 to 0.00027. Beyond 60 iterations, the reduction in loss becomes marginal, indicating convergence towards the global optimum. However, the optimization time continues to increase with additional iterations, demonstrating the trade-off between computational cost and precision. Therefore, around 50–60 iterations provide an optimal balance between accuracy and efficiency for the CMA-ES optimizer.

6 Conclusion

The hybrid CMA-ES and L-BFGS optimization approach proved highly effective for solving the non-linear parameter estimation problem. The CMA-ES phase successfully identified a promising region within the parameter space, while the GPU-accelerated L-BFGS phase efficiently refined this estimate to achieve a high-precision solution. The final optimized parameters satisfy all boundary constraints, and the extremely low mean ℓ_1 loss confirms an excellent fit to the provided data. Overall, the hybrid framework achieves a strong balance between global exploration (via CMA-ES) and rapid local convergence (via L-BFGS), ensuring both robustness and computational efficiency in parameter estimation.

References

[1] Nikolaus Hansen. *The CMA Evolution Strategy: A Tutorial*. arXiv preprint arXiv:1604.00772v2, 2017. Available at: https://arxiv.org/pdf/1604.00772

[2] Nikolaus Hansen. *The CMA Evolution Strategy: A Tutorial – PPSN 2024*. Inria Research Report, 2024. Available at: https://inria.hal.science/hal-04709819v1/file/CMATutorialPPSN2024.pdf

[3] K.P. Chong and S.H. Zak. *An Introduction to Optimization*. John Wiley & Sons, 4th Edition, 2013.

[4] PyTorch Documentation. *torch.optim.LBFGS — PyTorch 2.0 Documentation*. Available at: https://pytorch.org/docs/stable/optim.html#torch.optim.LBFGS

[5] Hansen, N. *pycma: Evolution Strategy for Python*. Available at: https://github.com/CMA-ES/pycma