

EXPLORING BLACK-BOX PARAMETER CALIBRATION TECHNIQUES ON
GENERIC SYSTEM DYNAMICS MODELS, USING SYNTHETIC DATA

by

Tuğba Uyar

B.S., Industrial Engineering, Istanbul Technical University, 2021

Submitted to the Institute for Graduate Studies in
Science and Engineering in partial fulfillment of
the requirements for the degree of
Master of Science

Graduate Program in Industrial Engineering
Boğaziçi University

2025

ACKNOWLEDGEMENTS

First and foremost, I would like to express my sincere gratitude to Prof. Yaman Barlas. Having the opportunity to work with him has been a true privilege; his deep knowledge of system dynamics and his thoughtful guidance have helped me develop a much broader perspective throughout this journey. I am genuinely thankful for his support and encouragement. I am also grateful to Assoc. Prof. Mustafa Gökçe Baydoğan, my thesis supervisor, for his guidance and supervision of this study.

My heartfelt appreciation goes to Assoc. Prof. Gönenç Yücel, whose insightful lectures, valuable feedback, and genuine encouragement have significantly shaped my academic experience. I feel lucky to have crossed paths with him both as a teacher and as a jury member. I would also like to thank Assoc. Prof. Mert Edalı for his participation in my thesis committee and for his valuable contributions.

I would also like to extend my sincere thanks to Melis, with whom I shared the ups and downs of graduate school over the past three years. Her friendship, support, and companionship made this journey much more meaningful and memorable.

I am deeply grateful to my family for their constant support. Above all, I am especially thankful to my brother, my lifelong source of unconditional encouragement. His steady presence, quiet confidence, and unwavering belief in me have been invaluable.

To Kerim Alp, whose presence made me believe that with you by my side, anything was possible. Thank you for being my greatest source of strength, hope, and happiness throughout this journey.

Finally, I would like to sincerely thank TÜBİTAK for supporting my graduate studies through the BİDEB 2210-A scholarship program.

ABSTRACT

EXPLORING BLACK-BOX PARAMETER CALIBRATION TECHNIQUES ON GENERIC SYSTEM DYNAMICS MODELS, USING SYNTHETIC DATA

This thesis investigates the application of black-box optimization algorithms for parameter calibration in system dynamics models under various noise conditions and limitations. Two representative models, the SEIR epidemiological model and the stock management model, were selected to evaluate the performance of five optimization methods: Grid Search, Random Search, Tree-structured Parzen Estimator (TPE), Covariance Matrix Adaptation Evolution Strategy (CMA-ES), and Gaussian Process (GP) Sampler. Synthetic datasets incorporating pure and autocorrelated noise were generated to simulate real-world observational uncertainties. The findings indicate that Covariance Matrix Adaptation Evolution Strategy (CMA-ES) outperforms other algorithms in terms of accuracy and robustness in most scenarios. Tree-structured Parzen Estimator (TPE) achieved competitive results, whereas Grid Search and Random Search showed limited accuracy and poor adaptability under changing noise conditions. While Gaussian Process (GP) Sampler produced consistent results when calibrating interdependent parameters, its high computational costs made it less efficient compared to other algorithms. Additionally, the extreme condition tests conducted with the stock management model underscored the importance of search space exploration. Algorithms with strong exploration capabilities were able to identify oscillatory behaviors that would otherwise have been overlooked. This study emphasizes the potential benefits of integrating traditional error-based metrics with dynamic behavior analysis to improve calibration processes. Future research should focus on examining hybrid optimization approaches, evaluating more complex models, and incorporating real-world data.

ÖZET

KARA-KUTU KALİBRASYON YÖNTEMLERİNİN SENTETİK VERİLERLE SİSTEM DİNAMİĞİ MODEL KALIPLARI ÜZERİNDE İNCELENMESİ

Bu tez, farklı gürültü koşulları ve sınırlamalar altında sistem dinamiği modellerinde parametre kalibrasyonu için kara-kutu optimizasyon algoritmalarının uygulanmasını incelemektedir. SEIR epidemiyolojik modeli ve Stok Yönetimi modeli örnek olarak seçilmiş ve Grid Search, Random Search, Tree-structured Parzen Estimator (TPE), Covariance Matrix Adaptation Evolution Strategy (CMA-ES) ve Gaussian Process (GP) Sampler yöntemlerinin performansları değerlendirilmiştir. Gerçek dünya belirsizliklerini simüle etmek amacıyla saf ve otokorelasyonlu gürültü içeren sentetik veri setleri üretilmiştir. Sonuçlar, Covariance Matrix Adaptation Evolution Strategy (CMA-ES)'in çoğu senaryoda doğruluk ve sağlamlık açısından diğer algoritmalarından üstün olduğunu göstermektedir. Tree-structured Parzen Estimator (TPE) rekabetçi sonuçlar elde ederken, Grid Search ve Random Search algoritmaları kalibrasyon doğruluğu konusunda zayıf performans sergilemiş ve değişen gürültü koşullarına uyum sağlamakta yetersiz kalmıştır. Gaussian Process (GP) Sampler, bağımlı parametreleri kalibre ederken istikrarlı sonuçlar sağlamış, ancak yüksek hesaplama maliyeti nedeniyle diğer yöntemlere göre daha az verimli olmuştur. Ayrıca, Stok Yönetimi modelindeki zorlu koşul testleri, kara-kutu algoritmalarında arama alanı keşif yeteneğinin önemini vurgulamıştır. Özellikle, keşif yeteneği güçlü algoritmaların, göz ardı edilebilecek salımlı davranışları dahi tespit edebilme potansiyeline sahip olduğu gözlemlenmiştir. Bu çalışma, geleneksel hata-tabanlı ölçütlerin dinamik davranış analiziyle birleştirilmesinin kalibrasyon süreçlerini iyileştirme potansiyeline dikkat çekmektedir. Gelecek çalışmalarda, hibrit optimizasyon yaklaşımlarının incelenmesi, daha karmaşık modellerin değerlendirilmesi ve gerçek dünya verilerinin dahil edilmesi önerilmektedir.

TABLE OF CONTENTS

ACKNOWLEDGEMENTS	iii
ABSTRACT	iv
ÖZET	v
LIST OF FIGURES	viii
LIST OF TABLES	xv
LIST OF SYMBOLS	xviii
LIST OF ACRONYMS/ABBREVIATIONS	xix
1. INTRODUCTION	1
2. LITERATURE REVIEW	4
2.1. Generic System Dynamics Models	4
2.2. Parameter Calibration in System Dynamics	7
2.3. Black-box Optimization Methods	10
2.4. Noise in Simulation Models	13
3. RESEARCH OBJECTIVES	15
4. METHODOLOGY	17
4.1. Model Selection and Definition	17
4.1.1. SEIR Model	18
4.1.2. Stock Management Model	19
4.2. Noise Generation and Synthetic Dataset Creation	19
4.3. Implementation of Optimization Methods	22
4.3.1. Grid Search	22
4.3.2. Random Search	23
4.3.3. Tree-structured Parzen Estimator (TPE)	24
4.3.4. Gaussian Process Sampler (GP Sampler)	26
4.3.5. Covariance Matrix Adaptation Evolution Strategy (CMA-ES)	27
4.4. Evaluation Metrics	30
5. EXPERIMENTS AND RESULTS	33
5.1. SEIR Model	33
5.1.1. Model Description	33

5.1.2. Experimental Design and Results	39
5.2. Stock Management Model	67
5.2.1. Model Description	67
5.2.2. Experimental Design and Results	73
5.2.3. Extreme Condition Test	101
5.3. Comparative Analysis Across Models	105
5.4. Analysis and Discussion of Results	109
6. CONCLUSION	111
REFERENCES	114
APPENDIX A: SEIR MODEL EQUATIONS	125
APPENDIX B: STOCK MANAGEMENT MODEL EQUATIONS	128

LIST OF FIGURES

Figure 2.1.	Examples of dynamic behaviors in system dynamics.	7
Figure 5.1.	SEIR model structure.	35
Figure 5.2.	Exposed and observed exposed population in pure noise case. . . .	42
Figure 5.3.	Best parameter sets of (a) infection fatality rate, (b) time to recover, and (c) incubation time in pure noise case.	43
Figure 5.4.	(a) Mean squared error (MSE), (b) vector distance results of the best parameter sets in pure noise case – SEIR model.	44
Figure 5.5.	Dynamic behaviors from noise seeds that yield the highest MSE values in pure noise case – SEIR model.	45
Figure 5.6.	Heatmap of vector distance by method and seed in pure noise case - SEIR model.	45
Figure 5.7.	Distribution of best trial numbers by optimization method in pure noise case - SEIR model.	46
Figure 5.8.	Exposed and observed exposed population in strong pure noise case.	47
Figure 5.9.	Best parameter sets of (a) infection fatality rate, (b) time to recover, and (c) incubation time in strong pure noise case.	48
Figure 5.10.	(a) Mean squared error (MSE), (b) vector distance results of the best parameter sets in strong pure noise case - SEIR model.	49

Figure 5.11. Dynamic behaviors from noise seeds that yield the highest MSE values in strong pure noise case - SEIR model.	50
Figure 5.12. Heatmap of vector distance by method and seed in strong pure noise case - SEIR model.	50
Figure 5.13. Distribution of best trial numbers by optimization method in strong pure noise case - SEIR model.	51
Figure 5.14. Exposed and observed exposed population in autocorrelated noise - mild case.	52
Figure 5.15. Best parameter sets of (a) infection fatality rate, (b) time to recover, and (c) incubation time in autocorrelated noise - mild case.	53
Figure 5.16. (a) Mean squared error (MSE), (b) vector distance results of the best parameter sets in autocorrelated noise - mild case - SEIR model.	54
Figure 5.17. Dynamic behaviors from noise seeds that yield the highest MSE values in autocorrelated noise - mild case - SEIR model.	55
Figure 5.18. Heatmap of vector distance by method and seed in autocorrelated noise - mild case - SEIR model.	55
Figure 5.19. Distribution of best trial numbers by optimization method in autocorrelated noise - mild case - SEIR model.	56
Figure 5.20. Exposed and observed exposed population in autocorrelated noise - strong case.	57

Figure 5.21. Best parameter sets of (a) infection fatality rate, (b) time to recover, and (c) incubation time in autocorrelated noise - strong case. . . .	58
Figure 5.22. (a) Mean squared error (MSE), (b) vector distance results of the best parameter sets in autocorrelated noise - strong case - SEIR model.	59
Figure 5.23. Dynamic behaviors from noise seeds that yield the lowest MSE values in autocorrelated noise - strong case - SEIR model.	60
Figure 5.24. Heatmap of vector distance by method and seed in autocorrelated noise - strong case - SEIR model.	60
Figure 5.25. Distribution of best trial numbers by optimization method in autocorrelated noise - strong case - SEIR model.	61
Figure 5.26. Exposed and observed exposed population in strong pure noise - sparse data case.	62
Figure 5.27. Best parameter sets of (a) infection fatality rate, (b) time to recover, and (c) incubation time in strong pure noise - sparse data case. . .	63
Figure 5.28. (a) Mean squared error (MSE), (b) vector distance results of the best parameter sets in strong pure noise - sparse data case - SEIR model.	64
Figure 5.29. Dynamic behaviors from black-box seeds that yield the highest MSE values in strong pure noise - sparse data case - SEIR model.	65
Figure 5.30. Heatmap of vector distance by method and seed in strong pure noise - sparse data case - SEIR model.	66

Figure 5.31. Distribution of best trial numbers by optimization method in strong pure noise - sparse data case - SEIR model.	66
Figure 5.32. Stock management model structure.	69
Figure 5.33. Inventory and observed inventory in pure noise case.	76
Figure 5.34. Best parameter sets of (a) PDel1, (b) PDel2, and (c) PDel3 in pure noise case.	77
Figure 5.35. (a) Mean squared error (MSE), (b) vector distance results of the best parameter sets in pure noise case - stock management model.	78
Figure 5.36. Dynamic behaviors from noise seeds that yield the highest MSE values in pure noise case - stock management model.	79
Figure 5.37. Heatmap of vector distance by method and seed in pure noise case - stock management model.	79
Figure 5.38. Distribution of best trial numbers by optimization method in pure noise case - stock management model.	80
Figure 5.39. Inventory and observed inventory in strong pure noise case.	81
Figure 5.40. Best parameter sets of (a) PDel1, (b) PDel2, and (c) PDel3 in strong pure noise case.	82
Figure 5.41. (a) Mean squared error (MSE), (b) vector distance results of the best parameter sets in strong pure noise case - stock management model.	83

Figure 5.42. Dynamic behaviors from noise seeds that yield the highest MSE values in strong pure noise case - stock management model.	84
Figure 5.43. Heatmap of vector distance by method and seed in strong pure noise case - stock management model.	84
Figure 5.44. Distribution of best trial numbers by optimization method in strong pure noise case - stock management model.	85
Figure 5.45. Inventory and observed inventory in autocorrelated noise - mild case.	86
Figure 5.46. Best parameter sets of (a) PDel1, (b) PDel2, and (c) PDel3 in autocorrelated noise - mild case.	87
Figure 5.47. (a) Mean squared error (MSE), (b) vector distance results of the best parameter sets in autocorrelated noise - mild case - stock management model.	88
Figure 5.48. Dynamic behaviors from noise seeds that yield the highest MSE values in autocorrelated noise - mild case - stock management model.	89
Figure 5.49. Heatmap of vector distance by method and seed in autocorrelated noise - mild case - stock management model.	89
Figure 5.50. Distribution of best trial numbers by optimization method in autocorrelated noise - mild case - stock management model.	90
Figure 5.51. Inventory and observed inventory in autocorrelated noise - strong case.	91

Figure 5.52. Best parameter sets of (a) PDel1, (b) PDel2, and (c) PDel3 in autocorrelated noise - strong case.	92
Figure 5.53. (a) Mean squared error (MSE), (b) vector distance results of the best parameter sets in autocorrelated noise - strong case - stock management model.	93
Figure 5.54. Dynamic behaviors from noise seeds that yield the highest MSE values in autocorrelated noise - strong case - stock management model.	94
Figure 5.55. Heatmap of vector distance by method and seed in autocorrelated noise - strong case - stock management model.	94
Figure 5.56. Distribution of best trial numbers by optimization method in autocorrelated noise - strong case - stock management model.	95
Figure 5.57. Inventory and observed inventory in strong pure noise - sparse data case.	96
Figure 5.58. Best parameter sets of (a) PDel1, (b) PDel2, and (c) PDel3 in strong pure noise - sparse data case.	97
Figure 5.59. (a) Mean squared error (MSE), (b) vector distance results of the best parameter sets in strong pure noise - sparse data case - stock management model.	98
Figure 5.60. Dynamic behaviors from black-box seeds that yield the highest MSE values in strong pure noise - sparse data case - stock management model.	99

- Figure 5.61. Heatmap of vector distance by method and seed in strong pure noise - sparse data case - stock management model. 100
- Figure 5.62. Distribution of best trial numbers by optimization method in strong pure noise - sparse data case - stock management model. 100
- Figure 5.63. (a) Best MSE, (b) worst MSE results of the best parameter sets in extreme condition test - stock management model. 103
- Figure 5.64. Best mean squared error results of an extreme condition in strong pure noise - sparse data case - SEIR model. 106

LIST OF TABLES

Table 5.1.	Parameter ranges for SEIR model.	40
Table 5.2.	Actual parameter values for SEIR model.	41
Table 5.3.	Smoothing parameters for pure noise case - SEIR model.	42
Table 5.4.	Execution times for each algorithm in pure noise case - SEIR model.	46
Table 5.5.	Smoothing parameters for strong pure noise case - SEIR model. . .	47
Table 5.6.	Execution times for each algorithm in strong pure noise case - SEIR model.	51
Table 5.7.	Smoothing parameters for autocorrelated noise - mild case - SEIR model.	52
Table 5.8.	Execution times for each algorithm in autocorrelated noise - mild case - SEIR model.	56
Table 5.9.	Smoothing parameters for autocorrelated noise - strong case - SEIR model.	57
Table 5.10.	Execution times for each algorithm in autocorrelated noise - strong case - SEIR model.	61
Table 5.11.	Parameters for strong pure noise - sparse data case - SEIR model.	62

Table 5.12.	Execution times for each algorithm in strong pure noise - sparse data case - SEIR model.	67
Table 5.13.	Parameter ranges for stock management model.	74
Table 5.14.	Actual parameter values for stock management model.	74
Table 5.15.	Smoothing parameters for pure noise case - stock management model.	76
Table 5.16.	Execution times for each algorithm in pure noise case - stock management model.	80
Table 5.17.	Smoothing parameters for strong pure noise case - stock management model.	81
Table 5.18.	Execution times for each algorithm in strong pure noise case - stock management model.	85
Table 5.19.	Smoothing parameters for autocorrelated noise - mild case - stock management model.	86
Table 5.20.	Execution times for each algorithm in autocorrelated noise - mild case - stock management model.	90
Table 5.21.	Smoothing parameters for autocorrelated noise - strong case - stock management model.	91
Table 5.22.	Execution times for each algorithm in autocorrelated noise - strong case - stock management model.	95

Table 5.23.	Parameters for strong pure noise - sparse data case - stock management model.	96
Table 5.24.	Execution times for each algorithm in strong pure noise - sparse data case - stock management model.	101



LIST OF SYMBOLS

$C^{(t)}$	Covariance matrix in CMA-ES
c_c	Learning rate in CMA-ES
c_μ	Learning rate in CMA-ES
c_σ	Learning rate in CMA-ES
d_σ	Damping parameter in CMA-ES
$g(x)$	Probability density function of unpromising solutions in TPE
k	Scaling factor in noise generation
$l(x)$	Probability density function of promising solutions in TPE
$m^{(0)}$	Initial mean in CMA-ES
p_c	Evolution paths for the covariance matrix in CMA-ES
p_σ	Evolution paths for the step size in CMA-ES
R_0	Basic reproduction number in SEIR
$Stime$	Smoothing time in noise generation
y_i	Objective function value in GP Sampler
z_k	Standard normal distribution in CMA-ES
$x_k^{(t)}$	Candidate solutions in CMA-ES
w_k	Weights assigned to each solution in CMA-ES
$\alpha(\theta)$	Acquisition function in GP Sampler
μ	Leading solutions in CMA-ES
μ_{eff}	Effective number of parents in CMA-ES
$\mu(\theta)$	Mean function in GP Sampler
$\sigma^{(0)}$	Initial step size in CMA-ES
θ_i	Parameter set in BBO algorithms
γ	Percentage of successful performance trials in TPE

LIST OF ACRONYMS/ABBREVIATIONS

ABI	Amortized Bayesian Inference
BATS	Behavior Analysis and Testing Software
BBO	Black-Box Optimization
CFD	Computational Fluid Dynamics
CMA-ES	Covariance Matrix Adaptation Evolution Strategy
EGO	Efficient Global Optimization
EI	Expected Improvement
GP	Gaussian Process
GPO	Gaussian Process Optimization
HMC	Hamiltonian Monte Carlo
ISTS	Indirect Structure Testing Software
MCMC	Markov Chain Monte Carlo
MSE	Mean Squared Error
ODE	Ordinary Differential Equations
POPS	Pattern-Oriented Parameter Determination
PSO	Particle Swarm Optimization
SD	System Dynamics
SEIR	Susceptible-Exposed-Infectious-Recovered
SIR	Susceptible-Infectious-Recovered
TMC	Traditional Monte Carlo
TPE	Tree-structured Parzen Estimator
TVR	Targeted Variance Reduction
WIP	Work-in-Process

1. INTRODUCTION

System dynamics (SD) modeling serves as an essential instrument for understanding and analyzing complex systems by simulating their dynamic behavior. It has been applied to a wide range of fields, including supply chain management, environmental management, industrial dynamics, and public health [1]. SD models allow researchers to investigate feedback loops, delays, and nonlinear behaviors frequently found in real-world systems [2,3].

The accuracy and reliability of SD models depend on sensitive parameter calibration that corresponds to the model's output with target empirical data. Traditional optimization techniques, such as gradient-based techniques and local search algorithms, have been utilized for parameter calibration [4,5]. Nonetheless, these methods often encounter significant challenges when it comes to tackling the nonlinearity and high-dimensional parameter spaces that are inherent to SD models [6]. Additionally, they may encounter convergence challenges and vulnerability to local optimum, which compromises model validity by preventing accurate replication of real-world behavior and reducing predictive accuracy [7].

A further problem is caused by the lack of reliable and comprehensive historical time series data, which are frequently essential to parameter estimation. In these situations, novel strategies such as the Pattern-Oriented Parameter Determination (POPS) method have been suggested. POPS prioritizes qualitative pattern matching instead of depending on quantitative data, providing an adaptable solution in scenarios where quantitative data are insufficient [6]. Although effective for parameter calibration using qualitative patterns, these strategies may face challenges in fully capturing the complexities of highly nonlinear and high-dimensional dynamic systems.

Moreover, noise can be present in data used for simulation-based studies. White noise has a constant power spectral density. In contrast, pink noise exhibits a power spectral density that is inversely related to frequency. Noise can mask critical system

behaviors, making the calibration process more complex and introducing uncertainty into model predictions [8].

To address these challenges, black-box optimization (BBO) techniques have been evaluated as an alternative approach. BBO approaches are particularly useful in cases where the model's algebraic equations are unknown and data include noise, as they allow the system to be treated as a 'black box' that relies only on inputs and outputs [9]. This property makes BBO approaches well-suited for complex nonlinear systems where obtaining analytical gradients or full model equations is unattainable. BBO methods use heuristic or probabilistic strategies to explore the parameter space and aim to identify high-quality solutions, instead of relying on conventional gradient-based approaches. Moreover, BBO approaches, such as the Targeted Variance Reduction (TVR) method in Bayesian optimization, have shown effectiveness in addressing noisy optimization problems, enhancing robustness under stochastic disturbances [10].

BBO approaches encompass diverse strategies designed to address optimization challenges in complex and uncertain environments. Several search strategies, such as Grid Search and Random Search, are used in some BBO frameworks to explore the parameter space either systematically or randomly [11]. However, unlike adaptive BBO methods, they do not use previous evaluations to refine their search. Other methods, such as Tree-structured Parzen Estimators (TPE) and Gaussian Processes (GP), utilize probabilistic models to pinpoint promising search areas [12]. Evolutionary techniques, such as the Covariance Matrix Adaptation Evolution Strategy (CMA-ES), dynamically adjust their search strategies to effectively tackle complex and high-dimensional optimization challenges. These methods emphasize the iterative refinement of solutions based on input-output evaluations, balancing exploration and exploitation of the solution space without requiring explicit gradient data [13]. Exploration involves investigating unexplored parts of the search space to discover diverse potential solutions, while exploitation focuses on refining the most promising areas to improve outcomes. Due to their adaptability, BBO methods are widely applied in diverse fields, including machine learning for hyperparameter tuning [12], engineering design optimization [9], and operations research for complex decision-making problems [14].

This research examines the use of BBO approaches for calibrating SD models, evaluating their ability to handle challenges such as noise in data. To achieve this, synthetic data will be generated using the SEIR and stock management models, both well-established in SD. The efficacy of five black-box optimization approaches (Grid-Search, Random Search, TPE, GP Sampler, CMA-ES) will be assessed across various noise types. Their performance will also be evaluated based on the oscillatory and S-shaped behaviors, including overshoot and decline, exhibited by the selected models. Ultimately, the findings aim to contribute to the development of more robust calibration procedures, which may enhance the reliability of parameter estimation in SD models.

The rest of this thesis is organized as follows: Chapter 2 provides a review of related literature, including SD modeling, parameter calibration, BBO methods, and noise in simulation models. Chapter 3 presents the research objectives. Chapter 4 describes the methodology, covering model selection, synthetic data creation, implementation of optimization methods, and evaluation metrics. Chapter 5 presents the experimental results and analyzes, including findings for two models, comparative evaluations, and the extreme condition test. Finally, Chapter 6 summarizes the study and provides suggestions for future work.

2. LITERATURE REVIEW

This chapter provides an overview of SD modeling with a focus on established models such as the SEIR model and stock management models. Parameter calibration plays a crucial role in ensuring that model outputs align with empirical data. However, traditional calibration techniques often struggle to address the complexity and nonlinearity of dynamic systems. In response to these challenges, various optimization approaches, including black-box optimization (BBO) methods, have been explored for parameter estimation. Although the application of BBO in SD remains an emerging area, studies have investigated its potential in addressing calibration difficulties. This section reviews relevant research on calibration techniques, the role of BBO methods, and the effects of noise on model performance.

2.1. Generic System Dynamics Models

System dynamics, a methodology established by Jay Forrester in the late 1950s, provides a robust framework for analyzing and understanding the behavior of complex systems over time. Originally introduced in Industrial Dynamics [2], SD was developed to address industrial challenges, including inventory control and production delays. Over the decades, it has evolved to encompass applications in various domains, including healthcare [15], environmental management [16], policy-making [17], and education [18].

The essence of SD lies in its ability to model feedback-driven behaviors. It enables analysis of how interdependent components interact to produce emergent system behaviors. This characteristic is particularly crucial for addressing complex issues that require a holistic perspective, as conventional linear cause-and-effect assumptions often fail to capture the complexities of real-world systems [19]. In these respects, SD provides a powerful approach for simulating system behavior and supporting strategic decision-making in various fields, particularly in environmental sustainability, resource management, and long-term economic planning [20].

SD has been effectively applied to epidemiology, complementing existing models and enhancing our understanding of disease transmission and control mechanisms. The SIR (Susceptible-Infectious-Recovered) model, introduced by Kermack and McKendrick in 1927 [21], is a foundational mathematical model in epidemiology. The SIR model classifies the population into three compartments: Susceptible (S) individuals who can contract the disease, Infectious (I) individuals who actively transmit the disease, and Recovered (R) individuals who have gained immunity. This foundational yet powerful model captures the dynamics of infectious disease transmission by simulating transitions between compartments over time. However, while the SIR model can represent infections with rapid transmission after exposure, it does not explicitly account for diseases with incubation periods.

The SEIR (Susceptible-Exposed-Infectious-Recovered) model was developed to address this limitation by extending the SIR model by incorporating an incubation phase. The SEIR model introduces an additional Exposed (E) compartment, representing individuals who have contracted the infection but are not yet fully infectious. Although, in the standard SEIR model, exposed individuals do not contribute to transmission, some COVID-19 model variations, such as Struben’s model [22] and the SEAHIR model [23], allow limited transmission from the Exposed (E) compartment. By incorporating this adjustment, the SEIR model accounts for the latency period observed in many infectious diseases, enhancing its applicability to diseases such as measles and influenza [24,25]. By extending the original SIR model, the SEIR model has become a widely used approach in SD modeling, particularly for public health interventions and policy evaluation [26–28].

SEIR models are widely used to study disease transmission dynamics in epidemiology, whereas models used in stock management illustrate the application of SD in operations and supply chain management. Stock management models represent the flow of goods through production stages, inventory management, and demand fulfillment by incorporating feedback loops that regulate ordering, production, and replenishment decisions. These models illustrate how delays in feedback loops, particularly inventory replenishment, can cause oscillatory behaviors known as the bullwhip effect [29].

This phenomenon occurs when small fluctuations in demand amplify as they propagate through the supply chain, leading to inefficiencies such as overordering or stock shortages. Perceived shortages can trigger overstocking, which in turn leads to underordering, creating cycles of instability [2]. These fluctuations are not only costly but can also impact service quality and reduce customer satisfaction [30]. Stock management models are commonly used to examine and manage feedback-driven instabilities in supply chains.

SEIR and stock management models demonstrate their versatility by illustrating how fundamental SD principles, such as feedback loops and stock-and-flow architectures, can be applied to address challenges in epidemiology and supply chain management. In the literature, there are also studies that consider these two domains together. For example, Li et al. [31] developed a mathematical model that explores the bidirectional feedback between global supply chain disruptions and COVID-19 dynamics, highlighting the interconnected nature of epidemic spread and supply chain instabilities.

Building on this perspective, this study examines these two models separately to assess how calibration techniques perform in different dynamic behaviors. SEIR models capture disease spread dynamics, typically exhibiting exponential growth in the early stages before stabilizing over time, sometimes displaying S-shaped or overshoot behavior depending on model parameters, while stock management models exhibit oscillatory patterns due to feedback delays. This difference allows for evaluating how calibration techniques perform under varying system behaviors, making them suitable for this research. To clarify the dynamic behaviors mentioned, Figure 2.1 presents conceptual examples that illustrate the characteristic patterns discussed in this study.

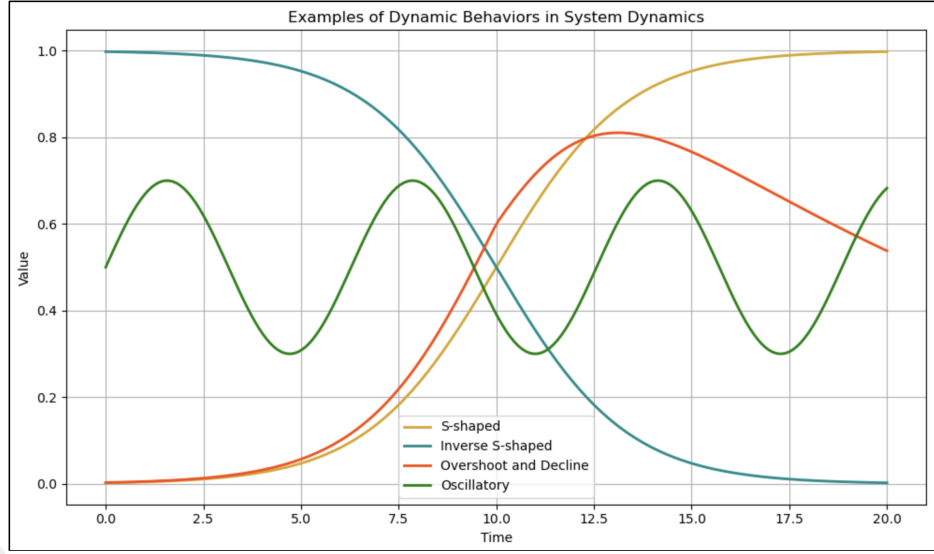


Figure 2.1. Examples of dynamic behaviors in system dynamics.

2.2. Parameter Calibration in System Dynamics

Parameter calibration is a critical process in SD, ensuring that models accurately represent real-world system behavior. It involves adjusting model parameters, such as transmission rates, recovery times, production delays, or inventory targets, so that simulated outputs align with empirical data. These parameters typically reflect key assumptions or rates that influence system behavior. Calibrating them appropriately can improve the model's validity and reliability. Parameter calibration is widely applied across multiple disciplines to enhance model accuracy and predictive reliability, including SD [4], engineering [32], epidemiology [33], and environmental science [34]. In epidemiology, SEIR models have been used for parameter calibration to estimate transmission rates and incubation periods during the COVID-19 pandemic, providing essential insights for public health decision-making [28]. In supply chain management, accurate parameter calibration enhances stock management precision, as discrepancies in calibration can contribute to inventory variations that impact operational efficiency [35].

Achieving cost-effective, efficient, and precise model calibration remains a challenge. Inadequate calibration may prevent even well-established models from aligning

with empirical data in practical applications. Historically, manual calibration techniques were employed to align model outputs with empirical data, often relying on iterative adjustments and expert judgment. Manual calibration can be highly time-consuming and relies on subjective decisions by the modeler, requiring significant expertise and iterative adjustments, which can limit its effectiveness for complex models with nonlinear dynamics [36].

To address these limitations, various statistical and optimization-based approaches have been developed to improve the efficiency and accuracy of parameter calibration. Statistical techniques, such as least squares estimation and maximum likelihood estimation, are commonly applied to minimize the differences between observed and simulated data [37]. Bayesian inference is used in parameter calibration due to its ability to incorporate prior knowledge and quantify uncertainty, providing a probabilistic framework for model estimation [38]. These techniques offer robust parameter estimation methods, especially when data quality and availability are adequate.

Metaheuristic optimization approaches are applied for parameter calibration, particularly in complex and nonlinear models, due to their ability to navigate high-dimensional search spaces [39, 40]. More broadly, methods such as genetic algorithms, simulated annealing, particle swarm optimization, and differential evolution have been developed to explore large parameter spaces and optimize complex functions [41, 42]. Metaheuristic approaches aim to explore multimodal search spaces and reduce the likelihood of getting trapped in local minima, challenges frequently encountered in parameter calibration processes [41, 43]. Genetic algorithms, inspired by evolutionary principles, iteratively refine parameter sets using selection, mutation, and crossover to enhance solution quality [44]. Simulated annealing employs a probabilistic acceptance criterion, sometimes permitting suboptimal solutions to avoid being trapped in local minima, with this probability gradually decreasing as the algorithm progresses [45]. Particle swarm optimization (PSO) is inspired by the social behavior of swarms, where particles adjust their positions based on both their previous best and their neighbors' best positions [46]. While PSO can effectively explore complex search spaces, its convergence behavior is sensitive to parameter tuning and problem-specific dynamics [47].

Despite advancements, parameter calibration in SD remains a complex task, as the nonlinearity and feedback-driven nature of these models can make calibration computationally intensive. Different parameter sets may replicate observed behavior even if they are based on unrealistic assumptions, potentially leading to accurate results for incorrect reasons [4]. The sufficiency and quality of experimental data are critical for ensuring parameter identifiability, as limited data can lead to nonidentifiability issues, thereby impacting the accuracy of model predictions [48]. In addition, overfitting is a challenge as it leads to poor generalization; a model may perform well on training data but fail when applied to unseen data [49].

While these challenges persist, black-box optimization (BBO) techniques provide a viable alternative for parameter calibration, particularly in navigating complex search spaces without directly relying on model equations. This makes them a valuable option for SD models, where objective functions often arise from complex feedback loops and nonlinear dependencies. Bayesian BBO approaches, such as Hamiltonian Monte Carlo (HMC), have demonstrated effectiveness in parameter calibration by addressing model uncertainty and exploring high-dimensional search spaces [38].

Similar to metaheuristic approaches, black-box optimization (BBO) methods can help mitigate the risk of overfitting by balancing exploration and exploitation throughout the calibration process. Unlike gradient-based or deterministic methods, many black-box optimization approaches employ stochastic mechanisms that promote exploration and can help mitigate the risk of getting trapped in local minima. Using approaches such as Bayesian optimization, black-box methods incorporate prior knowledge about parameter distributions through probabilistic surrogate models [12]. According to the same study, in data-scarce situations, Bayesian models may risk overfitting, necessitating careful hyperparameter tuning to maintain generalizability.

These characteristics position black-box optimization as a viable approach for parameter calibration in SD. However, its application in this domain remains underexplored, particularly in comparative studies evaluating its performance against traditional and metaheuristic methods. Conducting systematic evaluations of BBO's

performance in SD could enhance the robustness of parameter calibration and expand its usability in large-scale and high-dimensional modeling scenarios.

2.3. Black-box Optimization Methods

Black-box optimization (BBO) methods are commonly applied to optimization problems where the objective function is unknown, computationally expensive, or lacks a well-defined mathematical formulation. These methods are characterized by their ability to treat the objective function as a 'black box', relying solely on input-output evaluations without requiring derivative information or explicit mathematical assumptions. The adaptability of BBO approaches has made them widely applied in various domains, including engineering design and machine learning [12, 50].

A key characteristic of BBO is adaptive exploration, where the optimization algorithm iteratively selects evaluation points in the parameter space based on prior observations. Methods such as Bayesian optimization and evolutionary techniques are designed to balance exploration of untested regions and exploitation of promising areas, improving the likelihood of finding high-quality solutions. Bayesian optimization typically employs surrogate models, such as Gaussian Processes (GP), to construct a probabilistic model of the objective function and guide the search process through an acquisition function [12]. Evolutionary algorithms, such as the Covariance Matrix Adaptation Evolution Strategy (CMA-ES), dynamically adjust their search distribution, making them effective for high-dimensional optimization problems [13, 51].

Various BBO methods have been developed, each characterized by unique properties and performance trade-offs. Grid Search is a systematic approach that exhaustively evaluates predefined parameter values within a structured grid. In contrast, Random Search selects parameter values randomly, allowing for a broader exploration of high-dimensional spaces while reducing computational costs. When the dimensionality of the hyperparameter space is high, Random Search can be more effective than Grid Search, as it avoids exhaustive evaluations and focuses on a diverse set of parameter values [52]. This stochastic approach makes Random Search a viable alternative for

hyperparameter tuning, particularly when exhaustive search is computationally impractical [11].

Compared to Random Search and Grid Search, advanced techniques such as the Tree-structured Parzen Estimator (TPE) and Gaussian Process (GP) Sampler leverage probabilistic modeling to guide the search process more efficiently. TPE applies Bayesian probabilistic modeling to categorize parameter combinations into promising and less promising groups, focusing the search on areas more likely to yield optimal configurations [53]. Alternatively, Gaussian Process Optimization (GPO) develops a probabilistic model of the objective function and employs acquisition functions such as Expected Improvement (EI) to direct the search, making it particularly effective for continuous optimization problems [54]. CMA-ES dynamically adjusts the covariance matrix of its search distribution for nonconvex and high-dimensional problems, demonstrating strong adaptability in complex search spaces [13].

BBO methods have been applied in various domains to solve computationally expensive and analytically challenging optimization problems. One notable application is Google Vizier [55], a large-scale BBO service widely used for hyperparameter tuning in machine learning models. It serves as a core component of Google’s Cloud Machine Learning HyperTune subsystem. In simulation-based optimization, multi-fidelity Gaussian process bandit optimization has been proposed to explore parameter spaces more efficiently by balancing computational accuracy and cost [56]. In chemical and engineering applications, BBO has been employed for optimizing chemical reactor designs using Bayesian multi-objective optimization, integrating computational fluid dynamics (CFD) to improve reactor efficiency while minimizing simulation costs [57]. Furthermore, derivative-free optimization algorithms have been utilized for tuning global optimization solvers, such as BARON, demonstrating improvements in solver efficiency by optimizing parameter settings [58]. These applications highlight the adaptability of BBO methods in addressing complex and computationally expensive optimization problems, demonstrating their effectiveness in fields such as machine learning, engineering design, and simulation-based optimization [50].

In SD, BBO approaches are gaining attention as researchers investigate their potential to calibrate complex models. Recent applications in SD include Bayesian inference-based methods and Monte Carlo techniques for parameter estimation and uncertainty quantification. Demonstrating the efficacy of Bayesian techniques in managing parameter uncertainty, Andrade and Duggan [38] calibrated a system dynamics-based SEIR model using Hamiltonian Monte Carlo (HMC) and Bayesian inference. Building on this, Duggan [59] examined the impact of data availability on parameter calibration in an SEIR model using HMC, highlighting the importance of multiple indicators in improving estimation accuracy. Similarly, Wakeland and Homer [60] contrasted Traditional Monte Carlo (TMC) and Markov Chain Monte Carlo (MCMC) approaches in a health policy simulation model, underlining the relevance of uncertainty quantification in increasing model robustness. Offering computational benefits over conventional Bayesian techniques, Rahmandad et al. [61] proposed Amortized Bayesian Inference (ABI) to speed parameter estimation in SD models employing deep learning. Although not directly applied to SD models, Angun and Kleijnen [62] applied a variant of Efficient Global Optimization (EGO), a Bayesian Optimization-related method, to a (s, S) inventory model, demonstrating its effectiveness in handling stochastic constraints for risk-aware decision-making. While their focus was on inventory management, similar optimization techniques could be relevant for certain SD applications. Similarly, Bansal et al. [63] investigated Bayesian optimization for learning task-specific dynamics models, suggesting potential applicability in SD-based control frameworks.

The integration of BBO techniques into SD modeling remains an emerging area of research. Some studies have explored different types of BBO approaches for SD model calibration, including Bayesian optimization and stochastic optimization methods. BBO techniques have been applied in areas such as parameter estimation in epidemiological models and uncertainty quantification in policy simulations, suggesting their potential to support SD model calibration. While some studies focus directly on SD, others provide insights from related optimization contexts that may inform future research. As the application of BBO in SD continues to be explored, future studies could systematically evaluate their applicability and performance across different modeling scenarios.

2.4. Noise in Simulation Models

Noise in simulation models originates from three primary sources. First, intrinsic stochasticity arises when randomness is inherently part of SD. Second, extrinsic noise affects model outputs due to external factors such as measurement inaccuracies. Third, computational or structural noise emerges from numerical approximations, discretization errors, or limitations in model representation. Comprehending and managing noise can contribute to the development of more robust simulation models and improve their ability to capture real-world dynamics.

In this study, the focus is on extrinsic noise, which affects model outputs due to external factors. To generate synthetic data that reflects real-world uncertainties, noise was deliberately introduced into the model outputs, simulating the impact of measurement inaccuracies. This noise-affected synthetic dataset served as the reference data source in the BBO process. Instead of directly calibrating the model to its original outputs, the optimization aimed to adjust model parameters such that its unperturbed outputs closely matched the noisy dataset. This approach reflects real-world scenarios where observed data are often subject to inaccuracies, allowing the optimization process to account for uncertainty and improve the reliability of parameter estimation.

Beyond categorizing noise sources and identifying the relevant source in this study, it is also important to consider the specific characteristics of different noise profiles. White noise and pink noise are two commonly studied types of noise, each exhibiting distinct spectral characteristics and influencing the way uncertainty is introduced into simulation outputs. White noise is a stochastic process with a constant power spectral density across all frequencies and no temporal correlation, mathematically described as the formal derivative of Brownian motion [64]. Pink noise, also referred to as $1/f$ noise, is a type of colored noise characterized by greater power at lower frequencies and has been utilized in modeling natural and stochastic processes [65]. This is an example of autocorrelated noise, in which current values depend on past values. Pink noise appears in various contexts, including music, speech, and biological systems, and is associated with structured variability in long-term processes [66, 67].

Extrinsic noise, including measurement errors and environmental fluctuations, can introduce variability in simulation outputs, leading to inconsistencies in repeated runs and complicating parameter calibration and model validation by increasing uncertainty in predictions. Due to the challenges posed by different types of noise, including pink and white noise, specialized optimization approaches have been employed to mitigate their effects. Bayesian optimization techniques have been applied to manage noise in black-box optimization problems, leveraging probabilistic models to improve optimization performance under uncertainty [12,68]. In the context of SD modeling, alternative parameter estimation techniques have been investigated in certain cases, particularly where noise and data limitations can introduce additional challenges. In this regard, Hosseinichimeh et al. [37] introduced indirect inference as a simulation-based method for estimating parameters in SD models with constrained data, demonstrating its applicability in scenarios where likelihood functions are intractable and datasets are limited.

Building on these discussions, this study primarily examines the performance of BBO algorithms while also exploring how different noise characteristics may influence their effectiveness. To achieve this, pink and white noise were incorporated as extrinsic noise sources directly into the model outputs, representing aspects of real-world uncertainties, such as measurement errors.

3. RESEARCH OBJECTIVES

System dynamics (SD) models are widely used to analyze and simulate complex systems across various disciplines, including epidemiology, supply chain management, and ecological systems, among others [3]. The reliability and predictive accuracy of these models depend in part on effective parameter calibration, which aims to align model output with observed data. The calibration process in SD models can be challenging due to high-dimensional parameter spaces and nonlinear dependencies, often limiting the applicability of conventional optimization methods [7].

Noise in real-world data introduces additional complexity to the calibration process. Noise, such as measurement inaccuracies, environmental fluctuations, or random disturbances, can introduce uncertainty into system behavior patterns and may affect the accuracy of parameter estimations [8]. In SD modeling, white noise (pure noise) and pink noise (autocorrelated noise) influence system responses by introducing different levels of uncertainty. White noise, in particular, increases variability and can challenge parameter estimation reliability, whereas pink noise introduces correlation effects that can influence system behavior [64, 65]. These challenges highlight the relevance of various modeling and optimization approaches, which may contribute to improved parameter estimation and system reliability.

Black-box optimization (BBO) methods have emerged as viable alternatives to conventional calibration techniques, offering flexibility and potential robustness in handling high-dimensional parameter spaces and uncertainty without requiring explicit derivative information or complete model equations [12]. BBO approaches treat the model as a 'black box', primarily relying on input-output data to iteratively refine parameter estimations. This strategy may be particularly useful for dynamic systems where model complexity or the absence of explicit equations could limit the practicality of gradient-based techniques. Despite its increasing utilization across several domains, research on the efficacy of BBO approaches in SD modeling remains relatively limited. There is still a lack of systematic comparisons evaluating the performance of different

black-box methods in SD models. This gap underscores the importance of a thorough assessment of black-box approaches to evaluate their suitability and effectiveness in SD calibration.

The main objective of this research is to contribute to addressing this gap by evaluating the performance of five notable BBO methods, Grid Search, Random Search, Tree-structured Parzen Estimators (TPE), Gaussian Process (GP) Sampler, and Covariance Matrix Adaptation Evolution Strategy (CMA-ES), on two generic SD models. In this study, Grid Search and Random Search are included as baseline methods rather than as strong candidates for optimization, serving as reference points for more sophisticated BBO approaches. The selected models, chosen for their recognition in the literature, will be calibrated using synthetic datasets that include both pure and autocorrelated noise to approximate real-world complexities. The study aims to evaluate the relative accuracy and robustness of BBO methods in parameter estimation across different types and levels of noise.

By systematically analyzing the strengths and limitations of each BBO method, this research seeks to provide insights into the applicability of these techniques for parameter calibration in noisy environments. The results are expected to serve as a potential reference for researchers and practitioners engaged in SD modeling, while also contributing to a better understanding of the efficacy of optimization methods under challenging data conditions. Furthermore, this study seeks to contribute to future research by identifying potential gaps and exploring opportunities to advance the development of optimization methods in dynamic systems.

4. METHODOLOGY

This chapter details the systematic approach employed in this study to evaluate the performance of black-box optimization (BBO) methods in calibrating parameters of system dynamics (SD) models. This research examines the performance of optimization techniques applied to two classical models: the SEIR epidemiological model and the stock management model, under different conditions of noise and data availability. The methodological framework consists of four key components: model selection and definition, noise generation and synthetic dataset creation, implementation of optimization methods, and evaluation metrics.

The selected models are introduced, emphasizing their relevance, dynamic behavior, and alignment with the study objectives. The subsequent section outlines the generation of synthetic datasets incorporating distinct noise characteristics to approximate real-world uncertainties via controlled stochastic perturbations. The next section details the implementation of various BBO algorithms, including relevant adaptations to apply these methods to the selected models. The evaluation metrics used to assess algorithmic performance, namely accuracy, computational efficiency, and robustness, are outlined. This systematic approach ensures that the experiments are conducted rigorously and are designed to be replicable, facilitating comparisons and conclusions.

4.1. Model Selection and Definition

This study evaluates the performance of BBO algorithms using two classical models, the SEIR model and the stock management model, which exhibit distinct dynamic behaviors. The selected models illustrate two fundamental dynamic patterns frequently observed in SD: S-shaped growth and oscillatory behavior. The SEIR model, commonly utilized in epidemiology, demonstrates S-shaped dynamics in cumulative infection patterns, typically exhibiting an initial exponential growth phase of infections. Depending on parameter values and intervention scenarios, the model may stabilize, exhibit an overshoot followed by a decline, or represent a typical epidemic wave. The stock man-

agement model, integral to supply chain dynamics, often exhibits oscillatory behaviors due to feedback loops, delays, and decision rules, leading to variations in inventory levels and production rates.

The selection of these models is justified by their capacity to illustrate key dynamic behaviors. These models are widely examined in the literature and represent real-world processes where external uncertainties can affect data quality. In epidemiology, reporting errors and inconsistencies in case detection introduce noise into observed infection data. Similarly, in supply chains, measurement inaccuracies, delays, and data aggregation issues create distortions in stock and demand records. The noise added to the output data in this study is used to approximate uncertainty arising from external factors, specifically reporting and measurement errors in real-life data. These considerations guided the selection of the SEIR and stock management models for this study.

4.1.1. SEIR Model

As an extension of the SIR model, the SEIR model incorporates an incubation phase to better represent diseases with latent periods. In this study, a variation of the SEIR model is employed, where exposed individuals contribute to transmission, unlike in the standard SEIR framework. This adjustment aligns with certain COVID-19 model variations that incorporate partial infectiousness in the Exposed (E) compartment [22, 23]. The model is governed by a system of ordinary differential equations (ODEs) that describe the transitions between compartments over time, characterized by key parameters such as the basic reproduction number (R_0), the incubation period, and the recovery rate [69]. Mathematical models, including SEIR, inform public health interventions by modeling disease spread and contributing to the assessment of vaccination strategies and quarantine measures [70]. This study employs an SEIR model that captures a single epidemic wave, characterized by an S-shaped overshoot followed by a decline. Details of the model are presented in Section 5.

4.1.2. Stock Management Model

The stock management model is a well-established model in SD, capturing the dynamics of inventory levels, production rates, and customer demand in supply chains. It represents inventory as stocks and production/shipment as flows, interconnected by feedback loops that regulate supply-demand dynamics [3]. Decision rules, such as goal inventory and production adjustment rates, influence the system's behavior. These attributes make the model suitable for analyzing dynamic behaviors in supply chain management, including inventory control and production adjustments in response to demand variability [2]. The structured yet flexible nature of the model helps researchers and practitioners analyze and address inefficiencies in supply chain operations, including overstocking, understocking, and demand amplification. This study employs a standard stock management model, which exhibits oscillatory behavior. Further details are presented in Section 5.

4.2. Noise Generation and Synthetic Dataset Creation

To better represent real-world uncertainties, synthetic noise was introduced into the model outputs to simulate the effects of measurement inaccuracies and external variability. Instead of fitting the model to its original noiseless outputs, the optimization process adjusts the parameters to align the model's outputs with the noise-added synthetic dataset. This approach acknowledges the presence of uncertainties in observed data and allows for a more realistic assessment of the optimization procedure.

This multi-step approach, detailed below, aims to generate noise that approximates the uncertainties and fluctuations commonly observed in real-world data.

- **Random Noise Generation:** A normal distribution was used to generate noise, as it provides a statistically well-defined way to introduce stochastic variations spread around a mean.
- **Temporal Smoothing:** A first-order exponential smoothing was applied to the noise signal to reduce short-term fluctuations while preserving longer-term trends.

This was implemented using the SMOOTHI function in Vensim, which performs exponential smoothing over a specified delay time [71]. This process introduces temporal correlation, making the noise signal more consistent with the autocorrelated characteristics of real-world dynamic systems.

- **Integration with Observed Variables:** The smoothed noise was adjusted in magnitude and integrated into specific model variables through an additive transformation. Specifically, the noise signal was scaled by a fraction of the original variable and then added to its deterministic value, producing a noise-affected version of the observed output. This approach preserves the underlying system dynamics while introducing controlled stochastic variability.

This study introduced synthetic noise into the models to simulate real-world data imperfections, such as reporting inaccuracies and measurement variability. To better reflect real-world uncertainties, noise was incorporated into specific observed variables, including the ‘Exposed Population’ in the SEIR model and the ‘Inventory’ in the stock management model. This noise generation process does not change the underlying dynamics of the model; instead, it introduces variability in the observed output, thereby simulating the stochastic nature of real-world data.

In the SEIR model, synthetic noise was introduced into the ‘Exposed Population’ variable. The noise values were generated using a normal distribution with a mean of 0 and a standard deviation of 1, bounded between -10 and 10. A fixed seed value was used to ensure reproducibility across simulations. For this purpose, Vensim’s RANDOM NORMAL function was used

$$X = \text{RANDOM NORMAL}(-10, 10, 0, 1, \text{Seed}). \quad (4.1)$$

To smooth the generated noise and introduce temporal correlations, an exponential smoothing function was applied. Vensim’s SMOOTHI function was used for this purpose, where the input noise undergoes a first-order smoothing process over a specified smoothing time. This transformation reduces short-term fluctuations while preserving longer-term trends, ensuring that the noise remains autocorrelated. Addi-

tionally, the smoothed noise is scaled by a factor to control its magnitude before being incorporated into the model. The third parameter in the SMOOTHI function specifies the initial value of the smoothed variable, ensuring a stable transition at the start of the simulation. The smoothing process is defined as follows

$$SMOOTHX = SMOOTHI(X, Stime, 0) \times k. \quad (4.2)$$

In this context, k is a scaling factor, while $Stime$ represents the smoothing time constant that controls the extent of temporal smoothing. Adjusting $Stime$ and k controls both the frequency content and magnitude of the generated noise. A smaller $Stime$ preserves high-frequency fluctuations, resembling white noise, while a larger $Stime$ smooths the noise over time, introducing temporal correlations characteristic of pink noise. Similarly, increasing k amplifies noise intensity, whereas reducing it lowers the impact of noise on the system. By tuning these parameters, various stochastic noise profiles can be simulated. The smoothed noise is added to the exposed population variable to generate a noise-affected observed output

$$Observed\ Exposed = SMOOTHX \times (0.2 \times Exposed\ Population) + Exposed\ Population. \quad (4.3)$$

The above formula includes a scaling factor of 0.2, determined by testing different coefficient values to achieve a noise level that closely aligns with real-world variability. Different scaling factors were tested to ensure that the synthetic noise captures the stochastic variability and observational errors characteristic of real-world epidemiological data. After testing multiple values, 0.2 was chosen as it provided a balance between introducing realistic uncertainty in data reporting and ensuring that the underlying system behavior remained unaffected.

In the stock management model, noise was added to the ‘Inventory’ variable, following a similar procedure as in the SEIR model. The noise variable X was generated using Vensim’s RANDOM NORMAL function and then smoothed using the $Stime$ parameter. The resulting smoothed noise was scaled by a factor k , following the same

approach as in the SEIR model. The observed inventory is computed by adding the scaled smoothed noise to the inventory value, using the same scaling factor 0.2 as in the SEIR model

$$\text{Observed Inventory} = \text{SMOOTHX} \times (0.2 \times \text{Inventory}) + \text{Inventory}. \quad (4.4)$$

Thus, the inventory values incorporate noise-induced observational errors. This method provides a structured approach to incorporating stochastic variability in both models.

4.3. Implementation of Optimization Methods

This study performs parameter calibration in SD models using black-box optimization methods, which are known for solving complex, nonlinear, and nonconvex optimization problems. These methods are particularly useful when the objective function is unknown or too complex to express analytically. This study employs five black-box optimization techniques: Grid Search, Random Search, Tree-structured Parzen Estimator (TPE), Gaussian Process (GP) Sampler, and Covariance Matrix Adaptation Evolution Strategy (CMA-ES). This section provides a structured explanation of these methods by outlining their algorithmic procedures, mathematical formulations, and key properties.

4.3.1. Grid Search

Grid Search is an exhaustive search method that assesses a predefined set of parameter combinations. Methodically, it investigates the parameter space by establishing a grid of possible values for each parameter and evaluating the model performance at each point. In this respect, Grid Search is a basic in parameter calibration and represents a naive approach. The Grid Search algorithm steps are as follows:

- (i) Define a discrete set of values for each parameter.
- (ii) Compute the Cartesian product of these sets to form a complete parameter grid.

- (iii) For each parameter combination:
 - Use the given parameters to train the model.
 - Evaluate the model's performance on a validation dataset.
- (iv) Select the parameter set that yields the best performance within the grid.

In summary, the mathematical formulation of the Grid Search method involves defining $\theta_1, \theta_2, \dots, \theta_d$ as discrete sets of possible values for each of the d parameters, where the objective function f is evaluated at every point within the Cartesian product $\theta_1 \times \theta_2 \times \dots \times \theta_d$.

Grid Search systematically evaluates all possible parameter combinations, typically in a sequential manner. However, some Python libraries, including the Optuna library used in this study, introduce randomization into the process. This randomness influences the order in which parameter combinations are selected; however, all combinations are eventually evaluated.

Grid Search explores only predefined discrete points within a specified parameter space. As a result, even if the true optimal value lies within the given range, it may be missed if it does not coincide with any of the tested values. This study assigned five discrete values to each of the three parameters to define the search space for Grid Search in both models. As detailed later in the Experimental Design and Results, this configuration resulted in 125 parameter combinations, all of which were evaluated by setting the number of trials to 125.

4.3.2. Random Search

Random Search offers an alternative to Grid Search by randomly sampling parameter combinations from the search space instead of exhaustively evaluating all possible configurations. This method improves search efficiency in high-dimensional spaces by exploring a broader range of parameter values without being restricted to a fixed grid [11]. The steps of the Random Search algorithm are as follows:

- (i) Define the search space by specifying a range or probability distribution for each parameter.
- (ii) Randomly sample a fixed number of parameter sets from the defined search space.
- (iii) For each sampled parameter set:
 - Use the sampled parameters to train the model.
 - Evaluate the model’s performance on a validation dataset.
 - Select the parameter set that yields the best performance based on a predefined evaluation metric.

The mathematical definition of Random Search defines θ as the parameter space, where n points $\{\theta_1, \theta_2, \dots, \theta_n\}$ are sampled independently and identically distributed according to a predefined probability distribution over θ . The objective function f is evaluated at each θ_i , and the best-performing parameters are identified by $\theta^* = \arg \min_{\theta_i} f(\theta_i)$ [11].

In Random Search, each trial is performed independently. This technique relies on randomness to explore different regions of the search space. In large search spaces, Random Search is less constrained than Grid Search, potentially increasing the likelihood of identifying well-performing parameter sets. However, its random nature may lead to redundant trials, reducing sampling efficiency.

This study applied Random Search with 125 trials within the predefined parameter search range, consistent across all methods except Grid Search. To ensure reproducibility, the algorithm was executed with five distinct seed values for each experimental scenario.

4.3.3. Tree-structured Parzen Estimator (TPE)

The Tree-structured Parzen Estimator (TPE) is a Bayesian optimization technique that models the conditional probability of parameter values given their objective function scores. Instead of directly modeling $p(y|x)$, as in traditional Bayesian optimization, TPE estimates two probability density functions: one for high-performing

parameter configurations and another for the remaining ones. New parameter values are selected by maximizing the ratio of these distributions to optimize expected improvement [52]. The procedure of the TPE algorithm is outlined as follows:

- (i) Initialize by evaluating the objective function at an initial set of parameter configurations sampled from a predefined prior distribution.
- (ii) Partition the evaluated parameter configurations into two sets: one containing the best-performing configurations (based on a quantile threshold) and another with the remaining ones.
- (iii) Use Parzen estimators to model the conditional densities $l(x)$ (for high-performing parameters) and $g(x)$ (for the remaining ones).
- (iv) Select new candidate parameters by maximizing the acquisition function based on the ratio $\frac{l(x)}{g(x)}$.
- (v) Evaluate the objective function at the newly selected parameter values.
- (vi) Update the density models $l(x)$ and $g(x)$ to incorporate the new observation.
- (vii) Repeat steps 2–6 until a predefined stopping criterion, such as a maximum number of evaluations or no further improvement, is met [52].

The TPE models the conditional densities of parameter values as $l(x) = p(x|y \leq y^*)$ and $g(x) = p(x|y > y^*)$, where y^* is the γ -quantile threshold, separating the best-performing parameter values from the rest. New parameter values are chosen by maximizing the acquisition function, given by $x^* = \arg \max_x \frac{l(x)}{g(x)}$ [52].

TPE adjusts the balance between exploration (sampling from less-explored regions) and exploitation (focusing on areas with better observed outcomes). This trade-off is controlled by a quantile threshold γ , which defines the fraction of observations classified as favorable. TPE relies on random sampling for its initial evaluations before refining its probabilistic model, making it important to control randomness for reproducibility and consistent comparisons. This study conducted 125 TPE trials, using five distinct black-box seed values, within the same predefined parameter search space as the other optimization methods. The γ value, and the prior weight parameter, which influences the model’s exploration tendency, were set to their default values.

4.3.4. Gaussian Process Sampler (GP Sampler)

Gaussian Process (GP)-based optimization techniques model the objective function as a probabilistic distribution over functions, enabling uncertainty-aware exploration and exploitation in parameter selection. These methods employ Bayesian inference to iteratively refine function estimates, incorporating prior function evaluations to improve search efficiency [54, 72]. The following is an outline of the Gaussian Process Sampler (GP Sampler) algorithm's process:

- (i) Evaluate the objective function at an initial set of randomly sampled parameter configurations.
- (ii) Fit a Gaussian Process model to the observed function evaluations, constructing a surrogate model of the objective function based on prior observations [72].
- (iii) Define an acquisition function, such as Expected Improvement (EI), to guide the selection of the next query point for function evaluation [54].
- (iv) Maximize the acquisition function to determine the next parameter configuration for evaluation.
- (v) Evaluate the objective function at the selected parameter configuration.
- (vi) Update the Gaussian Process model to incorporate the new observation.
- (vii) Repeat steps 3–6 until a predefined stopping criterion, such as a maximum number of evaluations or no further improvement, is met [54, 72].

In Gaussian Process-based optimization, the observed dataset is given as $D = \{(\theta_i, y_i)\}_{i=1}^n$, where θ_i represents the parameters and y_i represents the corresponding objective function values. The objective function $f(\theta)$ is assumed to be drawn from a Gaussian Process prior

$$f(\theta) \sim GP(\mu(\theta), k(\theta, \theta')), \quad (4.5)$$

where $\mu(\theta)$ is the mean function, and $k(\theta, \theta')$ is the covariance function, which defines the smoothness of the function [72].

The acquisition function $\alpha(\theta)$ is used to rank candidate parameter configurations, selecting the next evaluation point by maximizing

$$\theta_{n+1} = \arg \max_{\theta} \alpha(\theta), \quad (4.6)$$

guiding the optimization process by balancing exploration and exploitation, typically through criteria such as EI or Upper Confidence Bound [54].

GP Sampler initializes by randomly selecting a predefined number of trials to construct its model. In each successive trial, a Gaussian Process updates the model to identify promising locations for future evaluations. It balances exploitation, by focusing on regions with favorable outcomes, and exploration, by investigating less-explored areas using the uncertainty estimates of the Gaussian Process. GP Sampler uses the EI function as its default acquisition function. This function estimates the EI over the current best observation and prioritizes parameter configurations with higher potential gains. Consequently, it efficiently guides the search toward regions likely to yield superior outcomes.

In this study, GP Sampler was run for 125 trials with five different black-box seeds in each experimental scenario, using the EI acquisition function within the same parameter search range as other methods.

4.3.5. Covariance Matrix Adaptation Evolution Strategy (CMA-ES)

The Covariance Matrix Adaptation Evolution Strategy (CMA-ES) is an evolutionary optimization algorithm that self-adapts the covariance matrix of a multivariate normal search distribution. This adaptation improves search efficiency and facilitates adaptive exploration of the search space. CMA-ES is particularly effective for nonconvex, multi-modal optimization problems, making it a reliable method for challenging search spaces [13, 73]. The steps of the Covariance Matrix Adaptation Evolution Strategy (CMA-ES) algorithm are as follows:

(i) Initialization [13]:

The algorithm starts by defining an initial mean $m^{(0)}$, an initial covariance matrix $C^{(0)} = \sigma^2 I$, and an initial step size $\sigma^{(0)}$. The evolution paths p_σ and p_c are typically initialized as zero vectors, though their exact initialization may vary depending on implementation.

(ii) Sampling Candidate Solutions [73]:

At each iteration t , λ candidate solutions $\{x_k^{(t)}\}_{k=1}^\lambda$ are sampled as follows

$$x_k^{(t)} = m^{(t)} + \sigma^{(t)} B^{(t)} D^{(t)} z_k, \quad z_k \sim N(0, I), \quad (4.7)$$

where $C^{(t)} = B^{(t)} D^{(t)2} (B^{(t)})^T$ represents the covariance matrix decomposition.

(iii) Evaluation and Selection [13]:

Each candidate solution $x_k^{(t)}$ is evaluated based on the objective function $f(x_k^{(t)})$.

The top μ solutions are selected according to their ranking.

(iv) Update of the Mean [73]:

The mean is updated using the top-ranked μ solutions

$$m^{(t+1)} = \sum_{k=1}^{\mu} w_k x_{k:\lambda}^{(t)}, \quad (4.8)$$

where w_k are weights assigned based on ranking.

(v) Update of Evolution Path for Step Size [13]:

The evolution path p_σ is updated as

$$p_\sigma^{(t+1)} = (1 - c_\sigma) p_\sigma^{(t)} + \sqrt{c_\sigma(2 - c_\sigma) \mu_{eff}} B^{(t)} D^{(t)-1} (m^{(t+1)} - m^{(t)}), \quad (4.9)$$

where c_σ is a learning rate and μ_{eff} is the effective number of parents. The evolution path p_σ accumulates information about the global search direction.

(vi) Adaptation of Covariance Matrix [73]:

The covariance matrix is updated using rank-one and rank- μ updates

$$C^{(t+1)} = (1 - c_c) C^{(t)} + c_c p_c^{(t+1)} (p_c^{(t+1)})^T + c_\mu \sum_{k=1}^{\mu} w_k (x_k^{(t)} - m^{(t)}) (x_k^{(t)} - m^{(t)})^T, \quad (4.10)$$

where c_c and c_μ are learning rates. The rank-one update helps maintain long-term information, while the rank- μ update accelerates adaptation by incorporating variance from multiple selected solutions.

(vii) Adaptation of Step Size [73]:

The step size is updated at each iteration as follows

$$\sigma^{(t+1)} = \sigma^{(t)} \exp \left(\frac{\|p_\sigma^{(t+1)}\| - E[\|z\|]}{d_\sigma} \right), \quad (4.11)$$

where d_σ is a damping parameter adjusted based on problem dimension and population size. Lower values allow faster adaptation but may introduce instability, whereas higher values promote stability at the cost of slower adaptation.

(viii) Termination:

The procedure continues until a termination requirement is satisfied, such as a predefined number of iterations, a sufficiently small step size, or minimal changes in the objective function value.

In CMA-ES, p_c and p_σ denote the evolution paths for the covariance matrix and the step size, respectively. These paths accumulate information about the direction and magnitude of search steps, aiding the adaptation of the covariance matrix and improving convergence [13, 73]. Furthermore, μ_{eff} , the effective number of parents, is computed based on the weights w_k assigned to the selected solutions. It determines how much the selected solutions contribute to the adaptation of the covariance matrix and step size [73].

CMA-ES initiates with a defined starting point (mean) and an initial step size, which controls the scale of the search. These parameters dynamically adapt throughout the iterations to improve search efficiency. The covariance matrix encodes the dependencies between parameters and the shape of the search distribution, allowing the algorithm to adapt to the landscape of the objective function. The adaptation of the step size regulates the exploration-exploitation balance by dynamically expanding or contracting the search space. The step size is based on the success of previous iter-

ations in improving the objective function. When the cumulative step path length is shorter than its expected value, the algorithm reduces the step size, allowing for finer local search.

The initial mean and step size were kept constant throughout the study. The CMA-ES algorithm was run over 125 independent trials, using five different seed values per experimental case, with the same parameter search range as other methods.

In summary, the same parameter search range and identical black-box seed values were applied to the four methods, except for Grid Search, in both models. For Grid Search, parameter sets were created using five predefined values sampled from the standard parameter range. Since the seed value is irrelevant for Grid Search, it was not used. Algorithm input settings were kept at their default values, and each algorithm underwent 125 trials, with parameter calibrations applied accordingly.

4.4. Evaluation Metrics

Evaluating the efficacy of optimization algorithms in SD modeling requires well-defined criteria that reflect key aspects of algorithm performance. This study defines evaluation criteria based on academic research and the challenges of parameter calibration in dynamic systems, particularly accurate time-series fitting and computational efficiency. These criteria ensure a balanced assessment of accuracy, efficiency, and robustness in the context of SD modeling.

- Calibration Accuracy

- (i) Mean Squared Error (MSE):

The Mean Squared Error (MSE) is a standard metric for evaluating calibration accuracy by measuring the average squared deviation between observed and predicted values. It assigns greater weight to larger errors, aiding the optimization process in improving model fit to observed data [74].

- Computational Efficiency

- (i) Optimization Time (Execution Time):

This metric measures the total time taken by an algorithm to complete its optimization process. In SD models, computational efficiency plays an important role, as long optimization durations can affect practical applicability. Execution time is commonly used as a performance metric in optimization algorithms [75].

(ii) Convergence Rate:

This metric measures how quickly an algorithm approaches a sufficiently good solution, typically quantified by the number of iterations required. A faster convergence rate is generally preferred, as it can improve computational efficiency by reducing the number of function evaluations needed [76].

- Robustness to Noise

(i) Noise Sensitivity Analysis:

This measure evaluates an algorithm's performance in the presence of data noise, demonstrating its ability to handle real-world data imperfections. Noisy conditions, such as white or pink noise, can influence parameter estimation accuracy and affect model calibration. Studies on nonlinear dynamical systems suggest that different types of noise impact the robustness of parameter estimation, with colored noise potentially introducing biases in the estimated parameters [77].

- Solution Quality

(i) Normalized Vector Distance to Actual Parameters:

This metric computes the Euclidean distance between the estimated parameters and the actual parameters, providing a direct measure of estimation accuracy. A lower distance indicates better alignment between the estimated and true parameters, reflecting improved calibration quality [77]. Since synthetic data are used in this study, the actual parameter values are known, enabling a direct evaluation of the algorithm's performance in relation to the normalized vector distance to actual parameters.

(ii) Stability of Results:

Algorithm stability is a key aspect of optimization, as different executions with varying initial conditions can lead to fluctuations in performance. A

stable algorithm consistently converges to similar solutions across multiple runs, improving its reliability and reproducibility [78]. In this study, each algorithm was executed with five distinct seed values for each experimental scenario to evaluate its stability.

This research applies the evaluation criteria to assess and compare the efficacy of different optimization methods in calibrating SD models. The study aims to identify optimization methods that balance accuracy, efficiency, robustness, and, solution quality. Calibration accuracy is measured using MSE, computational efficiency is evaluated through execution time and convergence rate, and robustness to noise is examined in terms of its impact on optimization performance. This analysis provides insights into the suitability of various optimization methods for medium-scale SD models, particularly in scenarios where noise in the data may impact the calibration process.

5. EXPERIMENTS AND RESULTS

This chapter presents the experiments conducted to evaluate the performance of BBO methods on two established SD models: the SEIR model and the stock management model. These experiments evaluated the robustness and accuracy of parameter calibration under varying levels of noise and data availability. White and pink noise were applied to observed variables in both models to simulate real-world uncertainties. Additionally, limited data scenarios were tested to assess optimization performance with incomplete or sparse datasets. These experiments assess the adaptability of BBO techniques under complex, noisy, and constrained conditions, providing insights into their applicability in dynamic system modeling.

5.1. SEIR Model

The details of the SEIR model employed in this study are outlined in the next section. After describing the model's structure and key components, the equations for generating synthetic data used in the experimental cases will be presented. This section provides a detailed review of the experimental setup, including the conditions and assumptions governing its application. Finally, the results of the experiments will be presented.

5.1.1. Model Description

The SEIR model is widely used in epidemiological modeling due to its structured approach to disease dynamics, offering a balance between simplicity and analytical capability. Its structure allows extensions that incorporate additional complexities, enabling adaptation to different epidemiological contexts [79]. The SEIR model has been applied to past pandemics, including the 1918 influenza and more recent outbreaks such as COVID-19. Mills et al. [80] used it to estimate the reproductive number of the 1918 influenza pandemic, contributing to understanding its spread. Similarly, Rahman et al. [81] reviewed various COVID-19 modeling approaches, noting that SEIR

models have been commonly used to analyze transmission dynamics and evaluate the effectiveness of intervention strategies. Moreover, the SEIR model has been used to estimate the impact of epidemiological trends on healthcare capacity, as demonstrated in studies projecting COVID-19 healthcare demand [82].

The SEIR model often exhibits S-shaped dynamics in cumulative infection count over time, typically starting with an initial exponential growth phase followed by a slowdown as the susceptible population decreases. However, this pattern is not universal and can vary depending on factors such as intervention measures, population structure, parameter values, and model extensions. When such dynamics occur, they illustrate feedback processes within the model: the depletion of the susceptible population reduces the pool accessible for infection, while the accumulation of recovered individuals contributes to herd immunity by lowering the effective reproduction number below one [69].

The system may exhibit epidemic waves under certain conditions, driven by factors such as seasonal changes, intervention measures, temporary immunity, or population heterogeneity. These waves are characterized by a rapid increase in infectious individuals, a peak, and a subsequent decline as the susceptible population decreases. Interventions such as immunization or social distancing can alter this pattern by flattening the epidemic curve, thereby reducing or delaying the peak.

This study incorporates an additional compartment, the Deceased Population, into the SEIR model to facilitate data collection on disease-related mortality, based on the framework proposed by Struben [22]. This SEIR model represents the progression of a disease in a population by categorizing individuals into five compartments: Susceptible, Exposed, Infected, Recovered, and Deceased. Figure 5.1 illustrates the stock-flow structure of the model, which captures the dynamic interactions between these compartments.

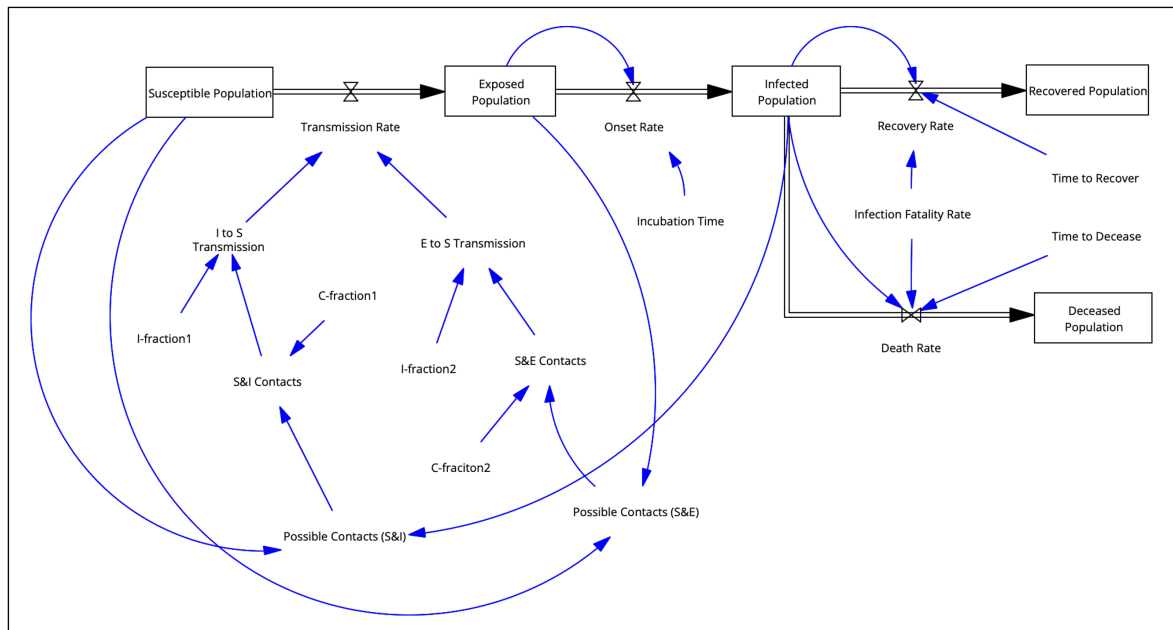


Figure 5.1. SEIR model structure.

The structure of the model is governed by four key parameters: transmission rate, onset rate, recovery rate, and death rate. These parameters collectively describe the transitions between compartments, capturing how the disease spreads, progresses, and ultimately leads to recovery or death. The compartments, or stocks, are interconnected through flows that determine the rate of transitions between them. These flows are influenced by parameters such as incubation time, infection fatality rate, recovery time, and decease time. Decease time represents the average duration from infection to death for individuals who do not recover, which determines the death rate along with the infection fatality rate.

The Susceptible Population decreases as individuals become exposed through interactions with infected and exposed individuals, governed by the Transmission Rate. The Exposed Population increases due to this transmission process and decreases at the Onset Rate, which is determined by an Incubation Time of 5 days. Once exposed individuals progress to the infected stage, they contribute to the Infected Population, which increases at the Onset Rate. The Infected Population decreases through two separate outflows: recovery and death. Recovery occurs at a rate determined by a

Time to Recover of 14 days, while death occurs at a rate determined by a Time to Decease of 7 days. The Death Rate is further influenced by the Infection Fatality Rate of 0.05, which specifies the proportion of infected individuals who eventually die. Recovered individuals are categorized into the Recovered Population, whereas deceased individuals are classified into the Deceased Population.

The model uses a Time to Recover of 14 days and an Incubation Time of 5 days, which are consistent with early studies of COVID-19. The Time to Recover reflects findings that most mild to moderate cases resolve within approximately 14 days [83]. The Incubation Time is set to 5 days, consistent with the reported average duration from exposure to symptom onset for COVID-19 [84, 85]. Additionally, the Infection Fatality Rate is set to 0.05 in this model, which is higher than most real-world estimates. This choice was made to simulate a scenario with a high mortality rate, allowing the model to exhibit more pronounced epidemic dynamics.

The Transmission Rate, which is the primary factor determining disease spread within the model, is calculated based on interactions between Susceptible individuals and both Exposed and Infected individuals. These interactions are weighted by contact rates and infection probabilities to estimate the overall rate of transmission. The main assumptions of the model are described below:

- **Population Dynamics:** The total population remains constant throughout the simulation, with no births, natural deaths, or migrations occurring. A homogeneous mixing assumption is applied, meaning that all individuals have an equal probability of interacting with each other, regardless of their location or status.
- **Disease Transmission:** The Transmission Rate is calculated based on interactions between Susceptible individuals and both Exposed and Infected populations. These interactions are weighted by constant contact and infection fractions, which determine the likelihood of transmission during each encounter. The contact and infection fractions are assumed to remain constant throughout the simulation.
- **Flow Dynamics:** The flow rates between compartments are governed by deterministic processes influenced by system parameters and population stocks. The

Onset Rate, which determines the transition from the Exposed Population to the Infected Population, is exclusively dependent on the Incubation Time and the size of the Exposed Population. In contrast, the Death Rate represents the transition from the Infected Population to the Deceased Population and is determined by the Infection Fatality Rate, adjusted by the Time to Decease.

The SEIR model utilizes a series of ordinary differential equations (ODEs) to characterize the transitions among its compartments. The equations represent the flow dynamics among Susceptible, Exposed, Infected, Recovered, and Deceased populations.

The rate of change in the Susceptible Population (S) is governed by the Transmission Rate, which represents interactions between susceptible individuals and those who are infectious or exposed. This relationship can be mathematically expressed as

$$\frac{dS}{dt} = - \text{Transmission Rate}. \quad (5.1)$$

The Transmission Rate includes interactions of Susceptible Population with both exposed and infected individuals and is calculated as

$$\text{Transmission Rate} = (I \text{ to } S \text{ Transmission}) + (E \text{ to } S \text{ Transmission}), \quad (5.2)$$

and the two components of the Transmission Rate are formulated as follows

$$\begin{aligned} I \text{ to } S \text{ Transmission} &= \text{Possible Contacts } (S \& I) \times C\text{-fraction1} \times I\text{-fraction1}, \\ E \text{ to } S \text{ Transmission} &= \text{Possible Contacts } (S \& E) \times C\text{-fraction2} \times I\text{-fraction2}. \end{aligned} \quad (5.3)$$

The Exposed Population (E) increases when susceptible individuals encounter infectious individuals, resulting in new exposures. It decreases as exposed individuals progress to the Infected compartment at the Onset Rate. This dynamic is represented by the following equation

$$\frac{dE}{dt} = \text{Transmission Rate} - \text{Onset Rate}. \quad (5.4)$$

The Onset Rate, which quantifies the transition speed from Exposed to Infected, is derived from the Incubation Time

$$\text{Onset Rate} = \frac{E}{\text{Incubation Time}}. \quad (5.5)$$

The Infected Population (I) increases as exposed individuals transition to an infectious state through the Onset Rate. It decreases as individuals either recover or die from the disease. The rate of change in this compartment is described by the following equation

$$\frac{dI}{dt} = \text{Onset Rate} - \text{Recovery Rate} - \text{Death Rate}. \quad (5.6)$$

The Recovery Rate quantifies the proportion of individuals who recover from the disease, calculated as follows

$$\text{Recovery Rate} = I \times (1 - \text{Infection Fatality Rate}) / \text{Time to Recover}. \quad (5.7)$$

Likewise, the Death Rate, which indicates the number of individuals who die from the disease, is defined as

$$\text{Death Rate} = I \times \text{Infection Fatality Rate} / \text{Time to Decease}. \quad (5.8)$$

Individuals who recover from the disease move to the Recovered compartment. The population change rate is equivalent to the Recovery Rate

$$\frac{dR}{dt} = \text{Recovery Rate}. \quad (5.9)$$

Individuals who die from the disease are incorporated into the Deceased compartment. The rate of change in this stock is characterized by the Death Rate

$$\frac{dD}{dt} = \text{Death Rate}. \quad (5.10)$$

The set of equations collectively describes the disease progression dynamics in the population by modeling the transmission, recovery, and death processes. Each equation highlights the role of key parameters, including Incubation Time, Infection Fatality Rate, and Time to Recover, in governing the transitions between compartments.

The model represents disease dynamics by mathematically defining interactions and transitions between compartments, allowing for precise calibration of essential parameters. This structure enables the simulation of various epidemic scenarios, including those involving noisy data to reflect real-world imperfections. The SEIR model captures nonlinear behaviors arising from interactions between compartments and provides a suitable structure for evaluating the performance of BBO methods.

5.1.2. Experimental Design and Results

The SEIR model experiments aimed to calibrate key epidemiological parameters, Incubation Time, Time to Recover, and Infection Fatality Rate, to ensure consistency between the model's outputs and synthetic datasets generated under controlled noise conditions. Different synthetic datasets were created to mimic real-world observational uncertainties by varying the noise characteristics through combinations of the smoothing constant $Stime$ and the scaling factor k . The calibration process aimed to produce distinct noise patterns, including white and pink noise, to simulate diverse but realistic scenarios for the optimization process. For each noise configuration, five stochastic runs were performed using different random seeds to account for variability in real-world data collection.

The generated datasets served as a basis for evaluating the performance of BBO methods in identifying parameter sets that best fit the noisy observations. The SEIR model was treated as a black box, with optimization methods aimed at minimizing the discrepancies between synthetic data and model outputs using metrics such as Mean Squared Error (MSE). This approach allowed for a systematic assessment of the optimization methods' effectiveness under various noise conditions.

To ensure a thorough and reproducible evaluation of BBO methods, specific procedures were applied to manage the randomness introduced by initial conditions. Each BBO algorithm was executed using five different seed values, influencing their initial parameter configurations. Additionally, five synthetic datasets were generated for each case, resulting in a total of 25 algorithm runs per case. This setup was designed to enable a comprehensive assessment of algorithm robustness under different initial conditions and across datasets produced with varying random normal function seed values. In the presentation of the results, the synthetic datasets generated using different random seeds are denoted as “Column 1”, “Column 2”, “Column 3”, “Column 4”, and “Column 5”. Similarly, the random seeds used for the initialization of BBO algorithms are referred to as “Seed” values.

Each algorithm was allowed to perform 125 trials per run to explore the parameter space. This setup enabled Grid Search to systematically evaluate all combinations within its predefined search space, consisting of a 5x5x5 grid for the parameters. All experiments in this study were implemented using the Optuna library (v4.0.0) [86]. Notably, GP Sampler was executed with the EI acquisition function, as Optuna v4.0.0 does not support alternative acquisition functions for this sampler.

The parameter ranges used in the Grid Search and other algorithms for the SEIR model are summarized in Table 5.1.

Table 5.1. Parameter ranges for SEIR model.

Parameter	Grid Search	Other Algorithms
<i>Infection Fatality Rate</i>	[0.01, 0.05, 0.1, 0.15, 0.2]	(0.01, 0.2)
<i>Time to Recover</i>	[1, 7, 14, 21, 25]	(1, 25)
<i>Incubation Time</i>	[1, 5, 10, 15, 20]	(1, 20)

The parameter ranges were selected to be sufficiently wide to include the true parameter values considered for this study. The true parameter values used to generate the synthetic datasets are presented in Table 5.2.

Table 5.2. Actual parameter values for SEIR model.

Parameter	Actual Value
<i>Infection Fatality Rate</i>	0.05
<i>Time to Recover</i>	14
<i>Incubation Time</i>	5

These values established a baseline for evaluating the accuracy of parameter estimates obtained through the optimization process. The calibrated results were systematically compared to the true parameter values to assess the effectiveness of each algorithm in accurately capturing the underlying system dynamics.

The experimental results are presented through a series of visualizations, including box plots of parameter estimates, MSE values, and vector distances. The visualizations provide a comprehensive assessment of BBO methods across various scenarios and noise levels. Box plots of parameter estimates reveal the variability and consistency of the optimization results, allowing for a straightforward comparison of each algorithm's robustness. The MSE box plots and vector distance metrics offer insights into the accuracy and efficiency of the methods in approximating the true parameter values. Vector distance heat maps contribute to the analysis by illustrating calibration accuracy patterns under different noise scenarios. These patterns reveal how sensitive each algorithm is to specific types of noise, providing insights into their robustness. Histograms depicting the number of trials required to reach the optimal solution offer a detailed assessment of convergence efficiency, demonstrating how quickly each method identifies the best parameter set within the specified trial limit. The following sections present the case details and experimental results for each case.

- Pure Noise Case:

In the pure noise scenario, synthetic datasets were generated using randomly fluctuating uncorrelated noise to represent real-world measurement errors without tempo-

ral dependencies. This type of noise, commonly referred to as white noise, was applied consistently to all observed values in the SEIR model. The experiments focused on evaluating the performance of black-box optimization methods under purely stochastic noise conditions, where autocorrelation was intentionally excluded. To prevent autocorrelation, $Stime$ was set equal to the model's dt value. Table 5.3 shows the values of noise generation parameters.

Table 5.3. Smoothing parameters for pure noise case - SEIR model.

Parameter	Value
$Stime$	0.125
k	1

Figure 5.2 presents the exposed population along with the observed exposed population data for a fixed SD seed value of 1234 in the context of the pure noise scenario.

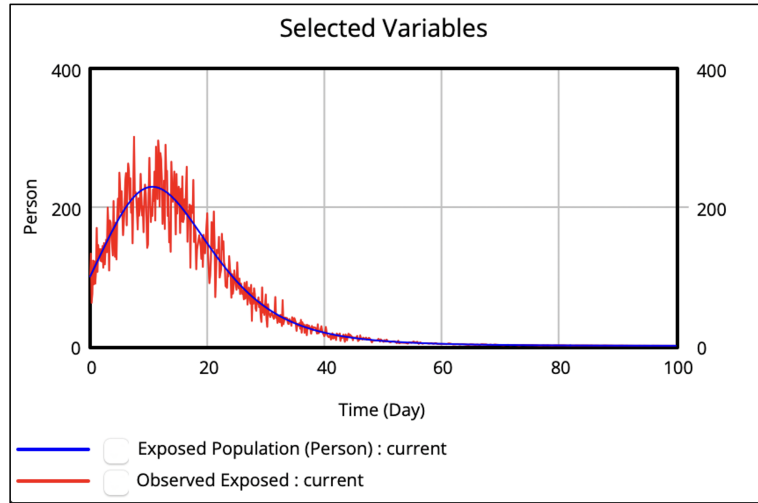
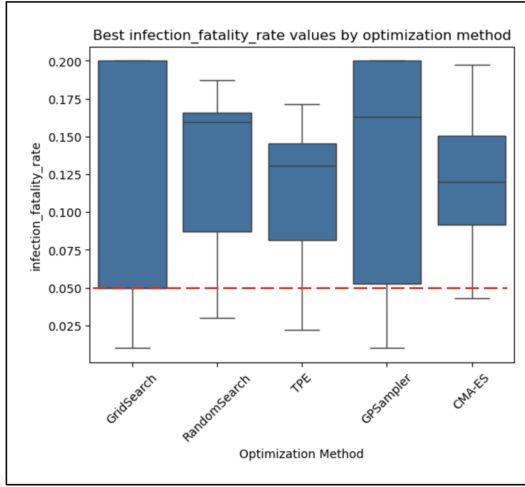
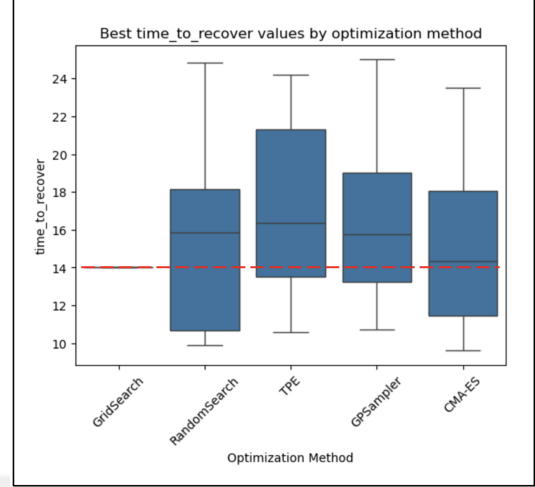


Figure 5.2. Exposed and observed exposed population in pure noise case.

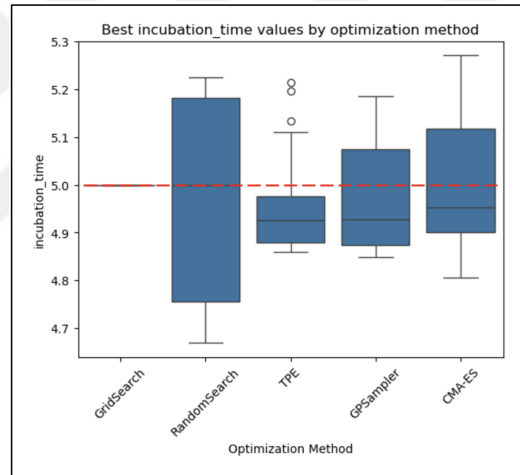
The sets of values of the best parameter values, characterized by the lowest MSE, identified by each algorithm in 25 experiments are presented in the box plot graphs below (Figure 5.3). The red lines indicate the true values of the parameters. Notably, Grid Search accurately identified two of the three parameters across all outputs.



(a) Infection Fatality Rate



(b) Time to Recover



(c) Incubation Time

Figure 5.3. Best parameter sets of (a) infection fatality rate, (b) time to recover, and (c) incubation time in pure noise case.

Figure 5.4 presents a comparison of performance metrics, specifically MSE and vector distance to the actual parameters, across all runs for each optimization method. The algorithms generally exhibit comparable MSE values in this scenario. However, vector distance values show greater variation, especially for Grid Search and GP Sampler.

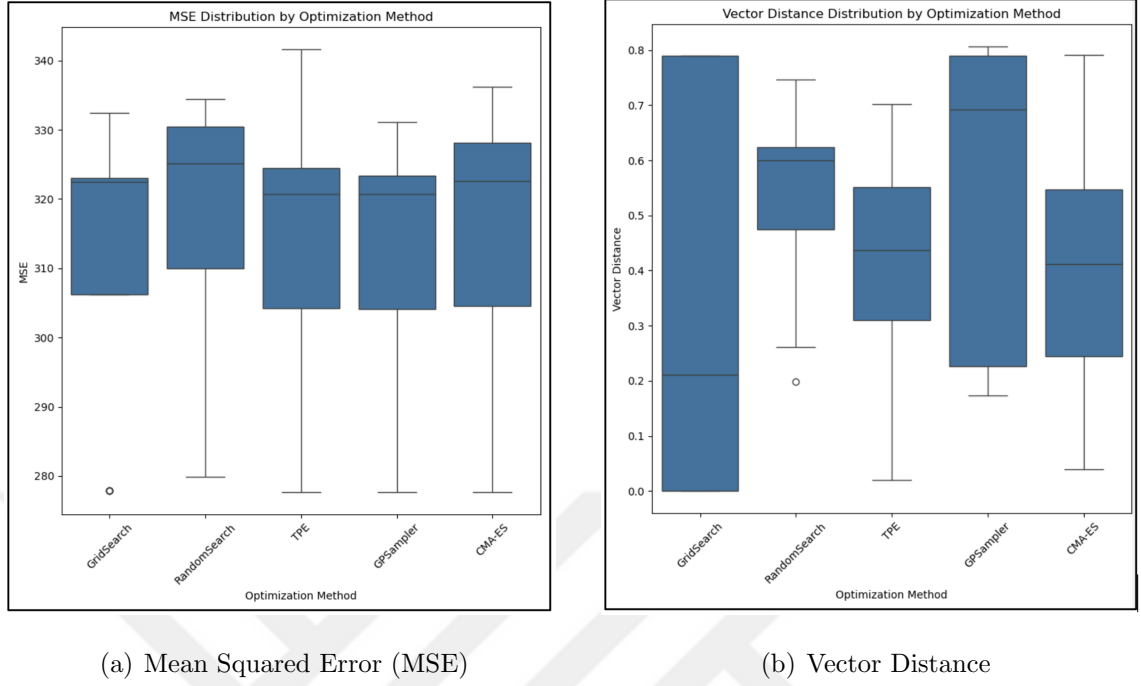


Figure 5.4. (a) Mean squared error (MSE), (b) vector distance results of the best parameter sets in pure noise case – SEIR model.

Figure 5.5 illustrates the dynamic behavior generated by the parameter set with the highest MSE value among the top 25 results for each algorithm. This representation helps assess how closely the algorithm's output aligns with the model's dynamic behavior. As observed, all algorithms demonstrate success in reproducing the overall dynamic behavior of the model.

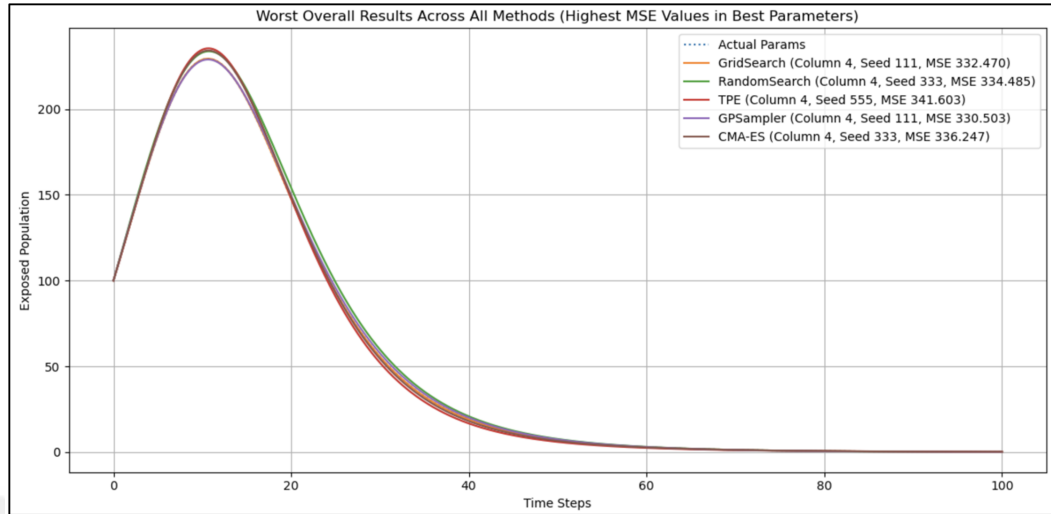


Figure 5.5. Dynamic behaviors from noise seeds that yield the highest MSE values in pure noise case – SEIR model.

The heatmap illustrates the vector distances between the estimated and true parameter sets, classified by various black-box seeds and optimization methods. Grid Search yields consistent results for all seed values, as it is not influenced by the black-box seed value (Figure 5.6).

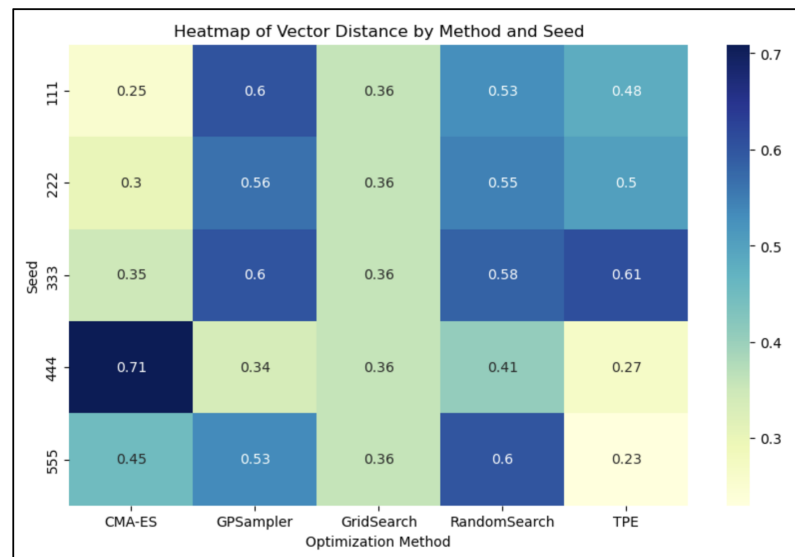


Figure 5.6. Heatmap of vector distance by method and seed in pure noise case - SEIR model.

This histogram illustrates the distribution of the trials in which the best solution was identified for each BBO method (Figure 5.7). In this case, Random Search was able to identify the best solution earlier in the trial process compared to other algorithms.

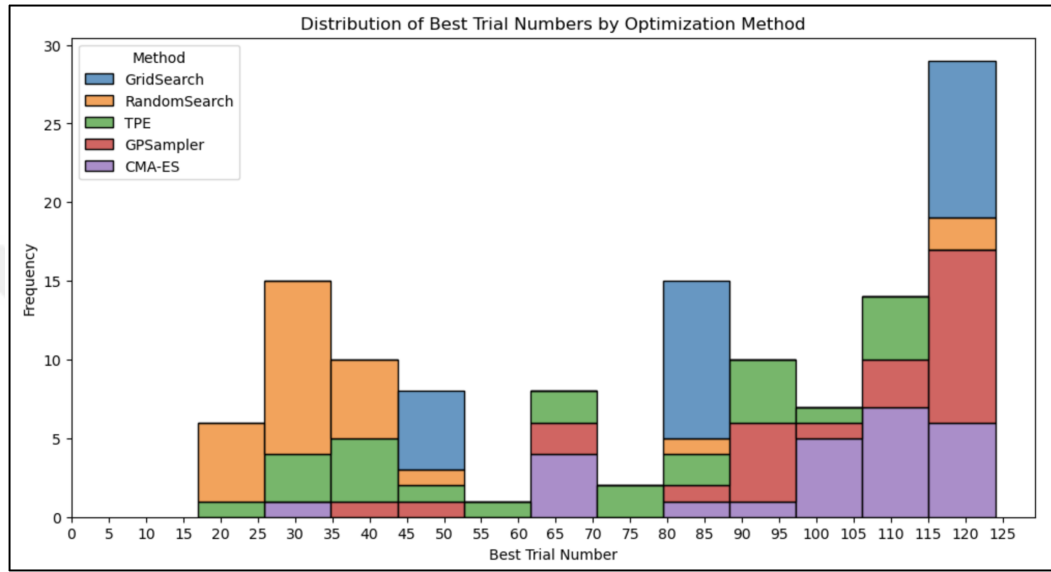


Figure 5.7. Distribution of best trial numbers by optimization method in pure noise case - SEIR model.

Additionally, the processing time of black-box algorithms serves as an important performance measure. Table 5.4 presents the total execution times for each BBO method.

Table 5.4. Execution times for each algorithm in pure noise case - SEIR model.

	Execution Time
Grid Search	56 seconds
Random Search	53 seconds
TPE	1 minute 5 seconds
GP Sampler	13 minutes 11 seconds
CMA-ES	55 seconds

- Strong Pure Noise Case:

Synthetic datasets were generated under the strong pure noise scenario by applying uncorrelated, intensified random noise to simulate substantial real-world measurement errors without temporal dependencies. The experiments focused on evaluating the robustness and accuracy of BBO methods in high-noise environments, excluding autocorrelation. To prevent autocorrelation, *Stime* was set equal to the model's *dt* value. In this instance, the scaling factor *k* is increased to achieve a greater amplitude. Table 5.5 presents the values of the noise generation parameters.

Table 5.5. Smoothing parameters for strong pure noise case - SEIR model.

Parameter	Value
<i>Stime</i>	0.125
<i>k</i>	3

Figure 5.8 illustrates the exposed population in conjunction with the observed exposed population data, utilizing a fixed SD seed value of 1234 within the framework of the strong pure noise scenario.

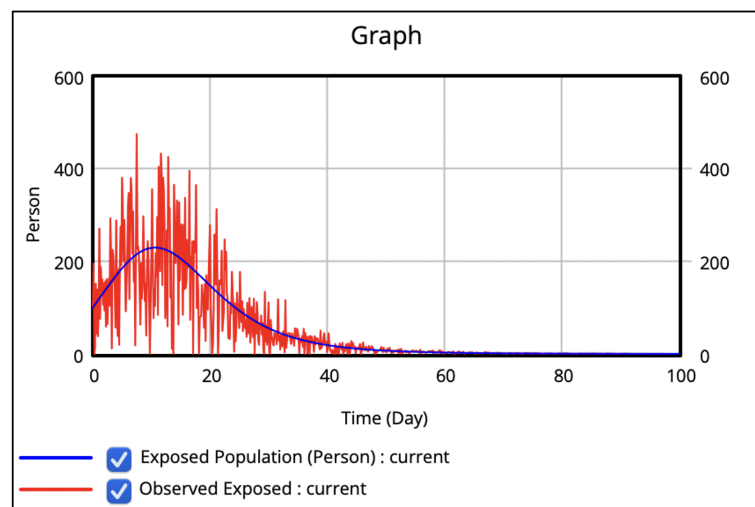


Figure 5.8. Exposed and observed exposed population in strong pure noise case.

Figure 5.9 presents the box plots of the 25 experiments that each algorithm performed to determine the best sets of parameters, as measured by the minimum MSE. GP Sampler showed variability in the estimation of the first two parameters.

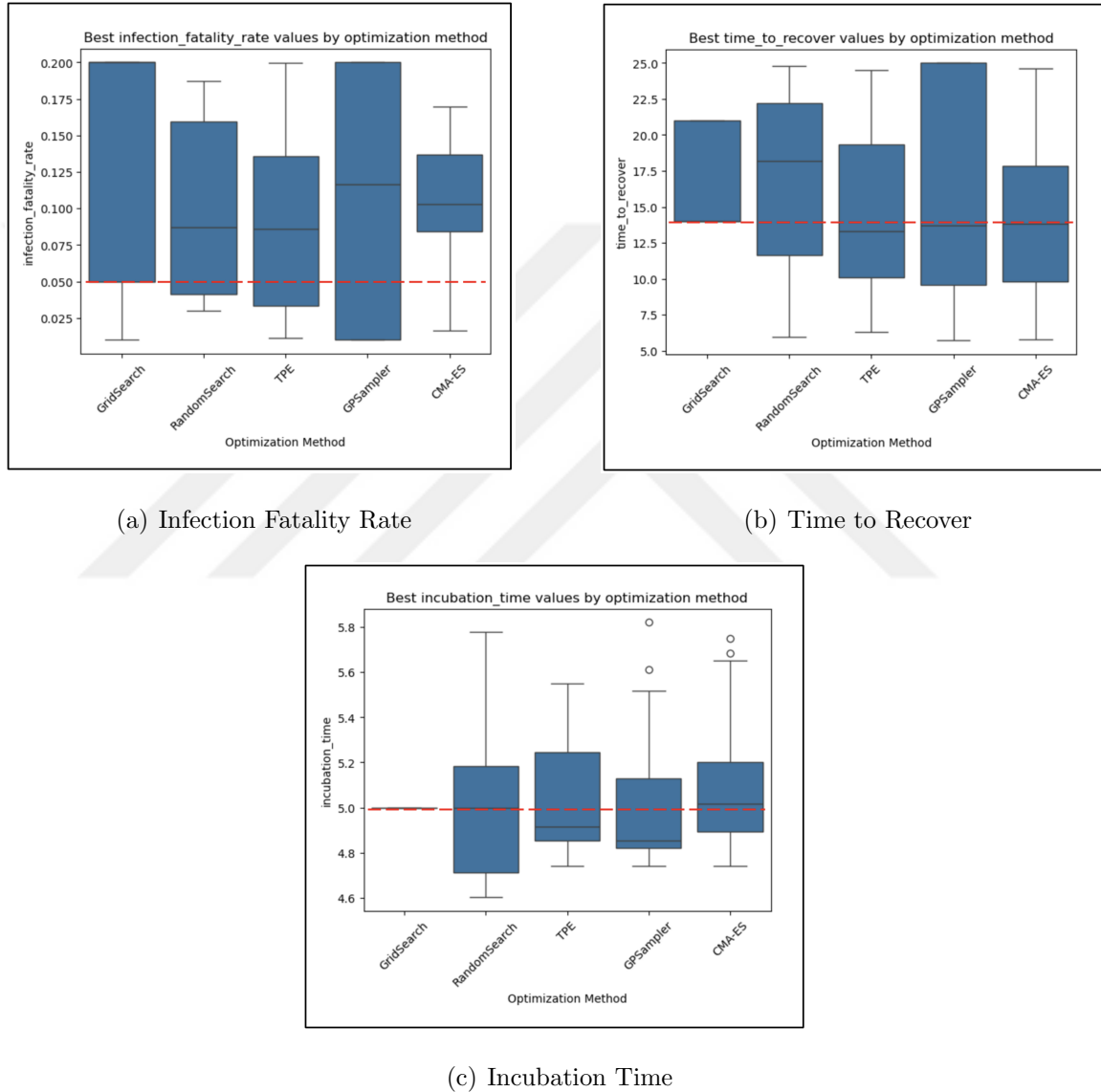


Figure 5.9. Best parameter sets of (a) infection fatality rate, (b) time to recover, and (c) incubation time in strong pure noise case.

Figure 5.10 compares performance metrics, such as MSE and vector distance to the real parameters, over all runs for each optimization method. Although MSE values remain relatively similar in this case, the distribution of vector distance for GP Sampler

is shifted upward compared to other algorithms, indicating a tendency toward higher average parameter deviations.

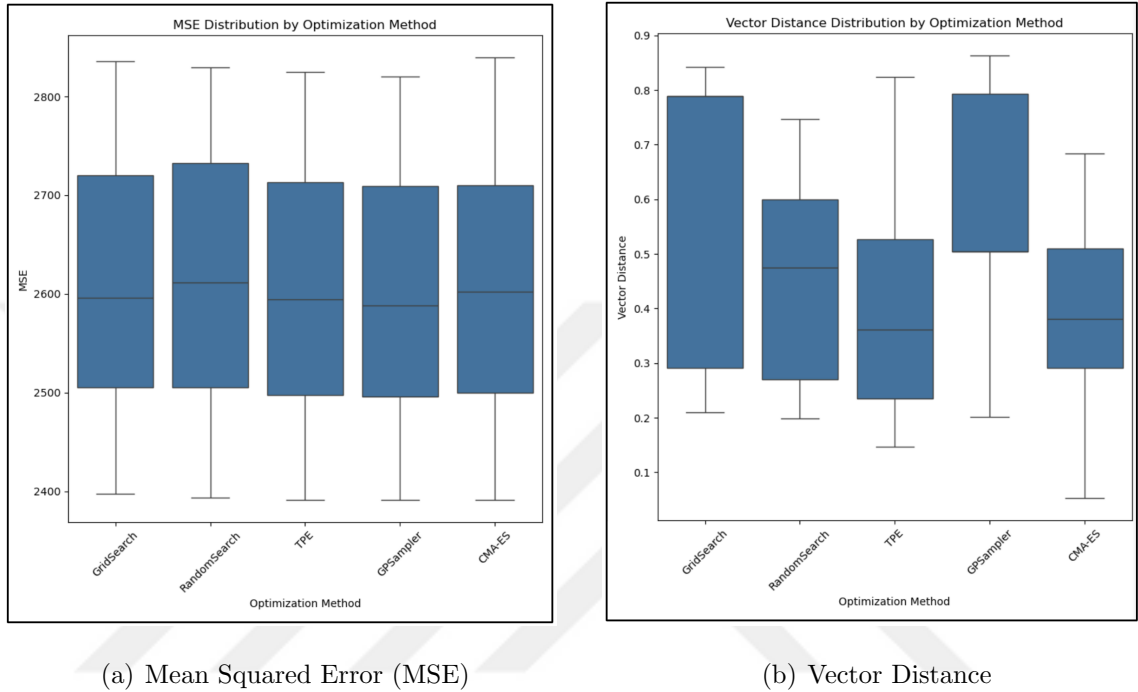


Figure 5.10. (a) Mean squared error (MSE), (b) vector distance results of the best parameter sets in strong pure noise case - SEIR model.

Figure 5.11 demonstrates that the algorithms were generally able to reproduce the model's dynamic behavior, even when using the parameter sets yielding the highest MSE values among their best results.

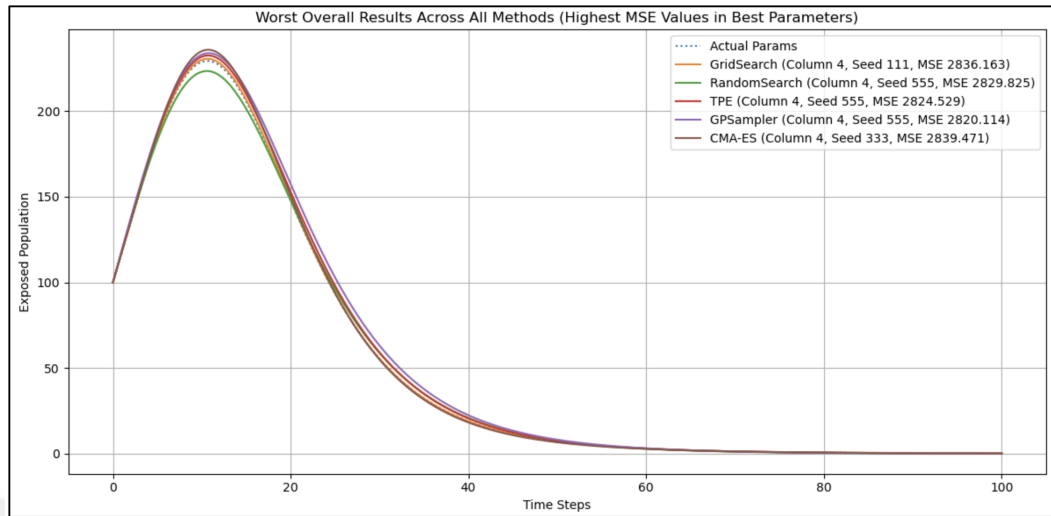


Figure 5.11. Dynamic behaviors from noise seeds that yield the highest MSE values in strong pure noise case - SEIR model.

The heatmap depicts the vector distances between the estimated and true parameter sets, organized by different black-box seeds and optimization techniques in a strong pure noise case (Figure 5.12). Compared to the other algorithms, GP Sampler and Grid Search produced parameter sets with higher vector distances.

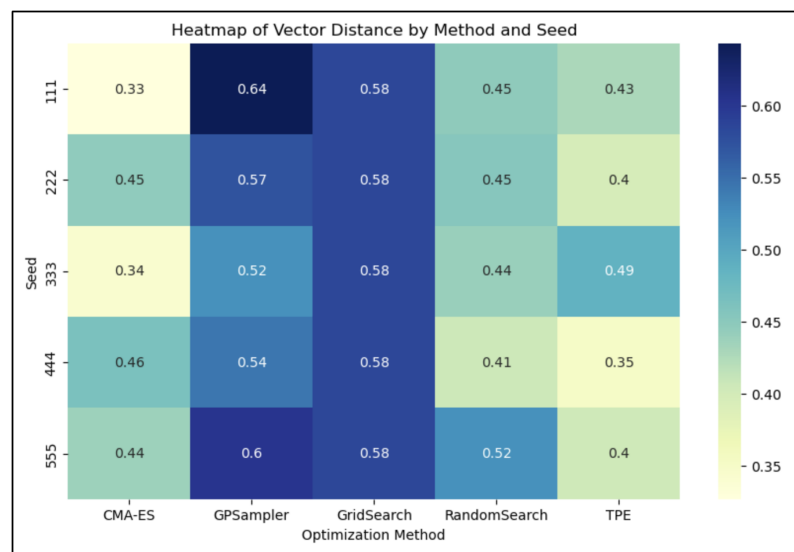


Figure 5.12. Heatmap of vector distance by method and seed in strong pure noise case - SEIR model.

Figure 5.13 illustrates the distribution of trials in which the best solution was identified for each BBO method in strong pure noise case. The best-performing results for GP Sampler were mostly achieved during the final iterations of the trials.

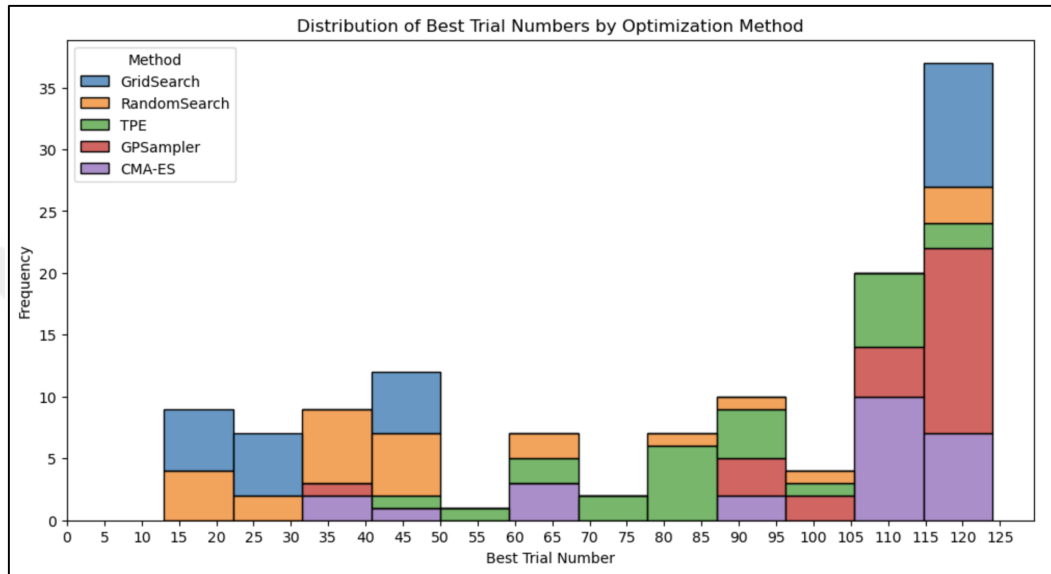


Figure 5.13. Distribution of best trial numbers by optimization method in strong pure noise case - SEIR model.

Table 5.6 displays the cumulative execution durations for each BBO technique in a strong pure noise scenario.

Table 5.6. Execution times for each algorithm in strong pure noise case - SEIR model.

	Execution Time
Grid Search	58 seconds
Random Search	52 seconds
TPE	1 minute 7 seconds
GP Sampler	13 minutes 18 seconds
CMA-ES	56 seconds

- Autocorrelated Noise - Mild Case:

In the autocorrelated noise mild case scenario, synthetic datasets were generated by adding pink noise to the observed values of the SEIR model. This temporally dependent noise was introduced to simulate measurement errors that evolve over time rather than occurring independently. To achieve the desired level of autocorrelation in the added noise, the parameters k and $Stime$ were deliberately selected and adjusted accordingly. Noise generation parameter values are shown in Table 5.7.

Table 5.7. Smoothing parameters for autocorrelated noise - mild case - SEIR model.

Parameter	Value
$Stime$	3
k	3

Figure 5.14 demonstrates the exposed population and the corresponding observed exposed population data under the mild autocorrelated noise condition, generated with a fixed SD seed value of 1234.

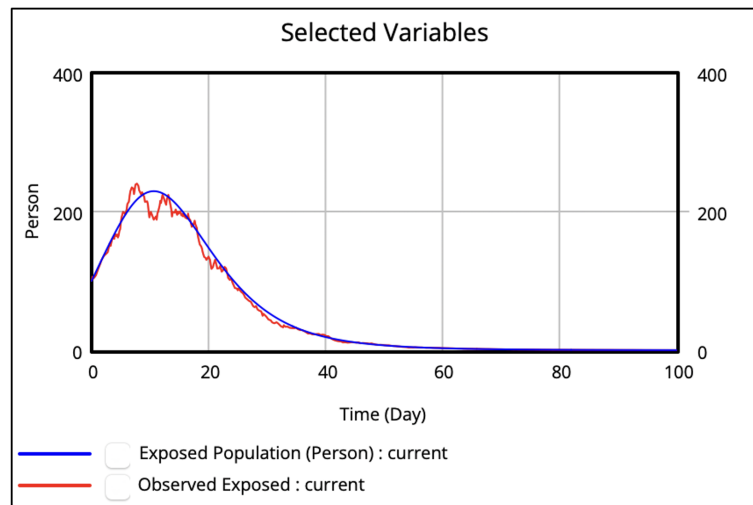
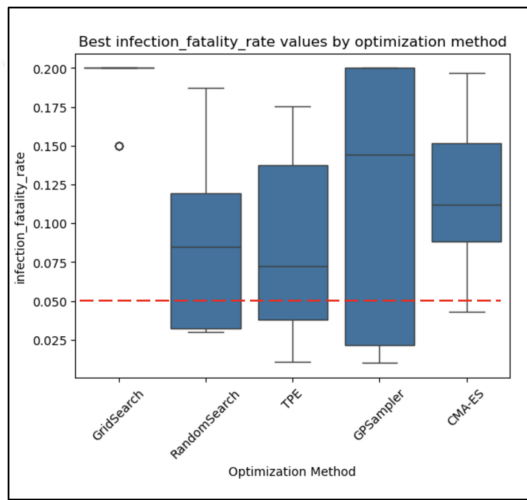
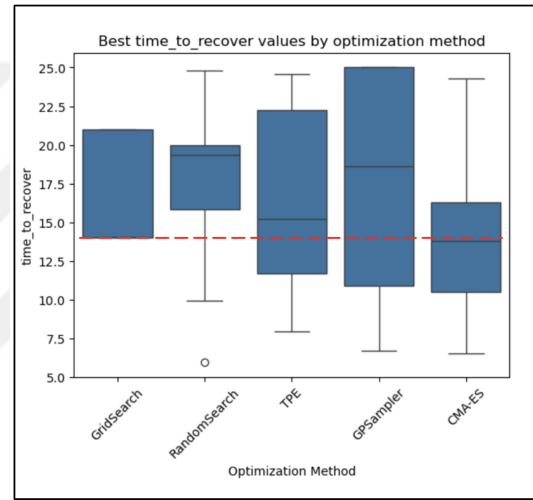


Figure 5.14. Exposed and observed exposed population in autocorrelated noise - mild case.

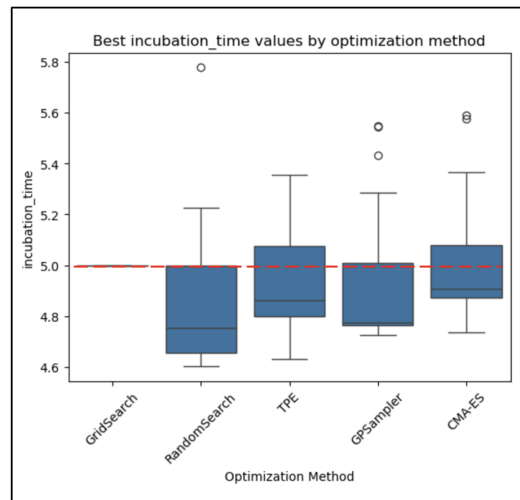
Figure 5.15 provides box plots summarizing the results of 25 experiments conducted for each algorithm, showcasing the estimated parameter sets identified using the minimum MSE as the evaluation criterion. In this noise scenario, although the true value of the Infection Fatality Rate was included among the candidate values, Grid Search failed to identify it correctly, while it successfully estimated another parameter. GP Sampler continued to exhibit variability in its results across trials.



(a) Infection Fatality Rate



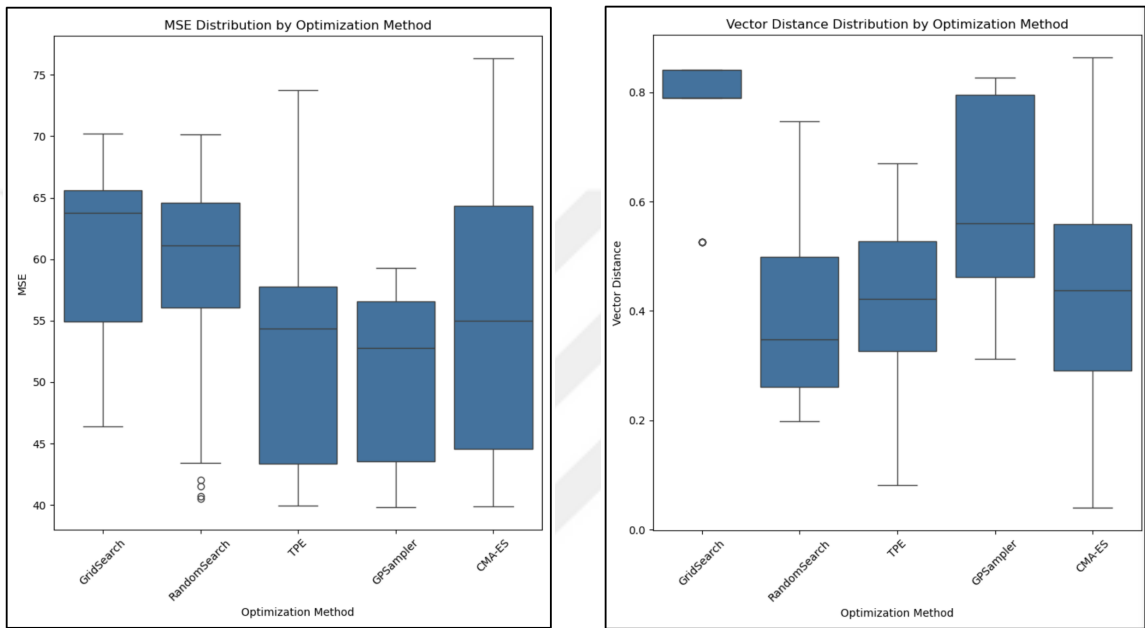
(b) Time to Recover



(c) Incubation Time

Figure 5.15. Best parameter sets of (a) infection fatality rate, (b) time to recover, and (c) incubation time in autocorrelated noise - mild case.

Figure 5.16 visualizes key performance metrics, including MSE and vector distance to the actual parameters, offering a comprehensive comparison across all optimization runs. Grid and Random Search yielded higher MSE values, and Grid Search also produced notably higher vector distances than the other algorithms.



(a) Mean Squared Error (MSE)

(b) Vector Distance

Figure 5.16. (a) Mean squared error (MSE), (b) vector distance results of the best parameter sets in autocorrelated noise - mild case - SEIR model.

As illustrated in Figure 5.17, similar to other cases, the algorithms were able to successfully capture the dynamic behavior in this scenario as well.

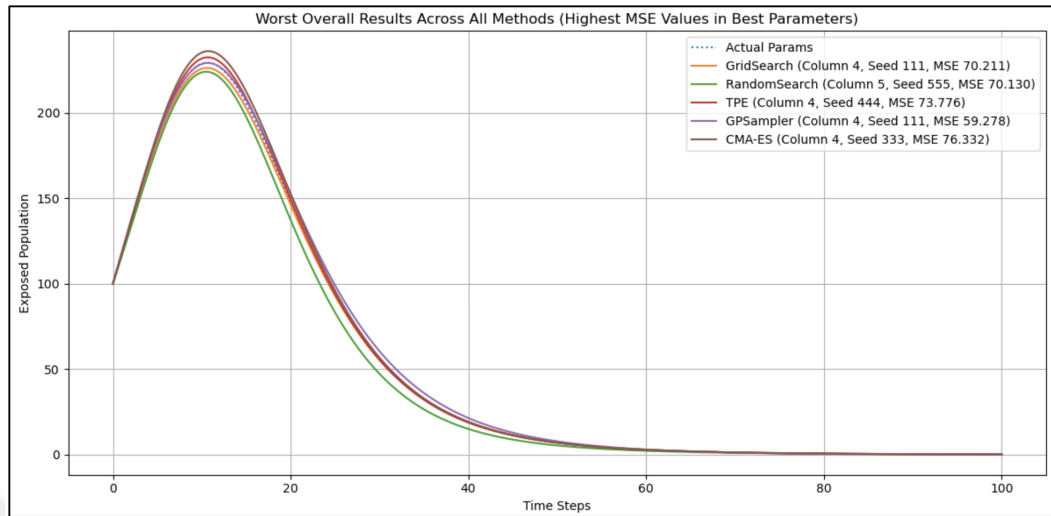


Figure 5.17. Dynamic behaviors from noise seeds that yield the highest MSE values in autocorrelated noise - mild case - SEIR model.

The heatmap in Figure 5.18 illustrates the vector distances between the estimated and true parameter sets, categorized by black-box seeds and algorithms.

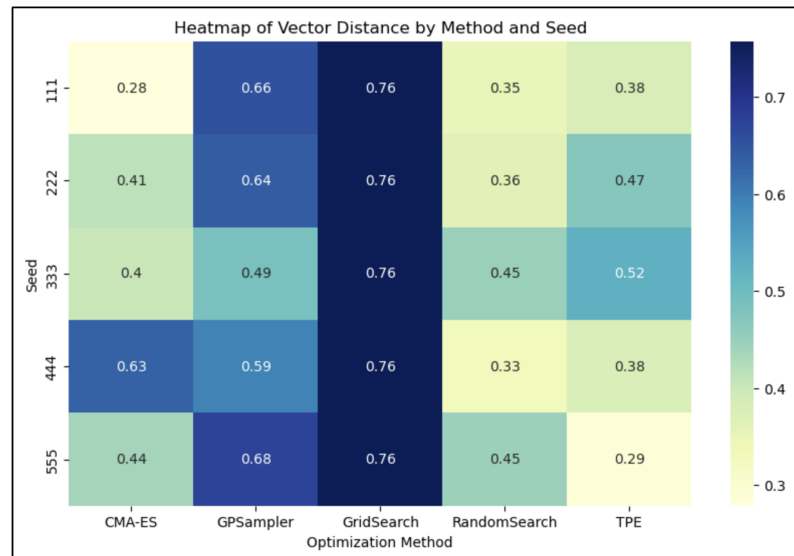


Figure 5.18. Heatmap of vector distance by method and seed in autocorrelated noise - mild case - SEIR model.

Figure 5.19 shows the distribution of trials at which each BBO method located the best solution under mild autocorrelated noise conditions.

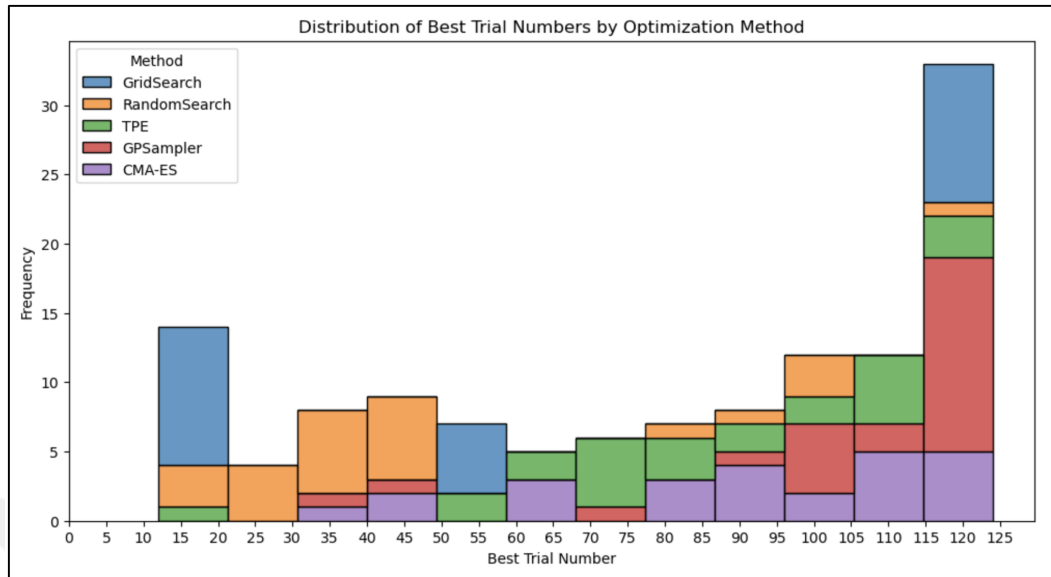


Figure 5.19. Distribution of best trial numbers by optimization method in autocorrelated noise - mild case - SEIR model.

Table 5.8 reports the total execution times for each BBO algorithm, offering insights into computational efficiency in the mild autocorrelated noise case.

Table 5.8. Execution times for each algorithm in autocorrelated noise - mild case - SEIR model.

	Execution Time
Grid Search	1 minute
Random Search	53 seconds
TPE	1 minute 7 seconds
GP Sampler	13 minutes 18 seconds
CMA-ES	56 seconds

- Autocorrelated Noise - Strong Case:

Synthetic datasets were generated under strong autocorrelated noise conditions to simulate real-world measurement errors with significant temporal dependencies. This type of noise, often associated with pink noise, was applied to the observed values of the

SEIR model. Temporal smoothing techniques were employed to amplify the autocorrelated characteristics of the noise, reflecting the complexities often present in dynamic systems. This scenario aims to assess the robustness and accuracy of BBO methods under increased autocorrelation, offering a comparison with simpler noise conditions. In the case of strong autocorrelated noise, both the temporal smoothing constant $Stime$ and the scaling factor k were increased to intensify the temporal dependencies and the amplitude of the noise. Table 5.9 summarizes the parameter values used for noise generation.

Table 5.9. Smoothing parameters for autocorrelated noise - strong case - SEIR model.

Parameter	Value
$Stime$	6
k	9

Figure 5.20 demonstrates the exposed population and the corresponding observed exposed population data under the mild autocorrelated noise condition, generated with a fixed SD seed value of 1234.

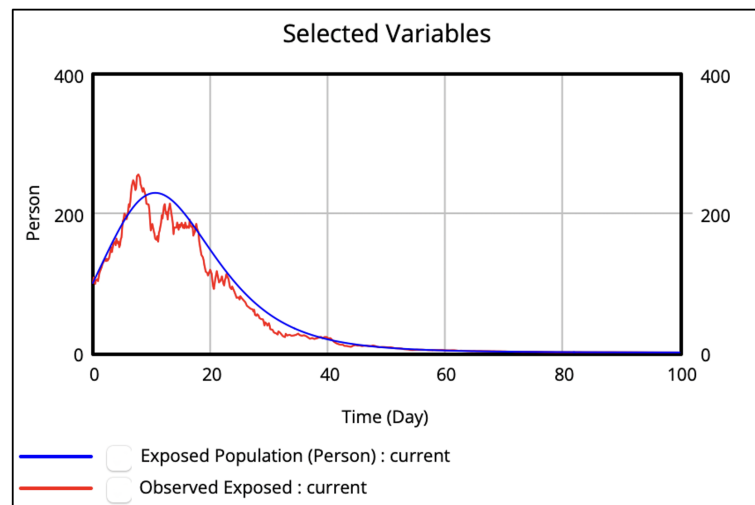


Figure 5.20. Exposed and observed exposed population in autocorrelated noise - strong case.

Figure 5.21 shows box plots of the 25 experiments per algorithm, highlighting the variability and consistency in identifying parameter sets with the lowest MSE. The algorithms produced more consistent and accurate estimates for the Incubation Time parameter, while the predictions for other parameters showed greater variability.

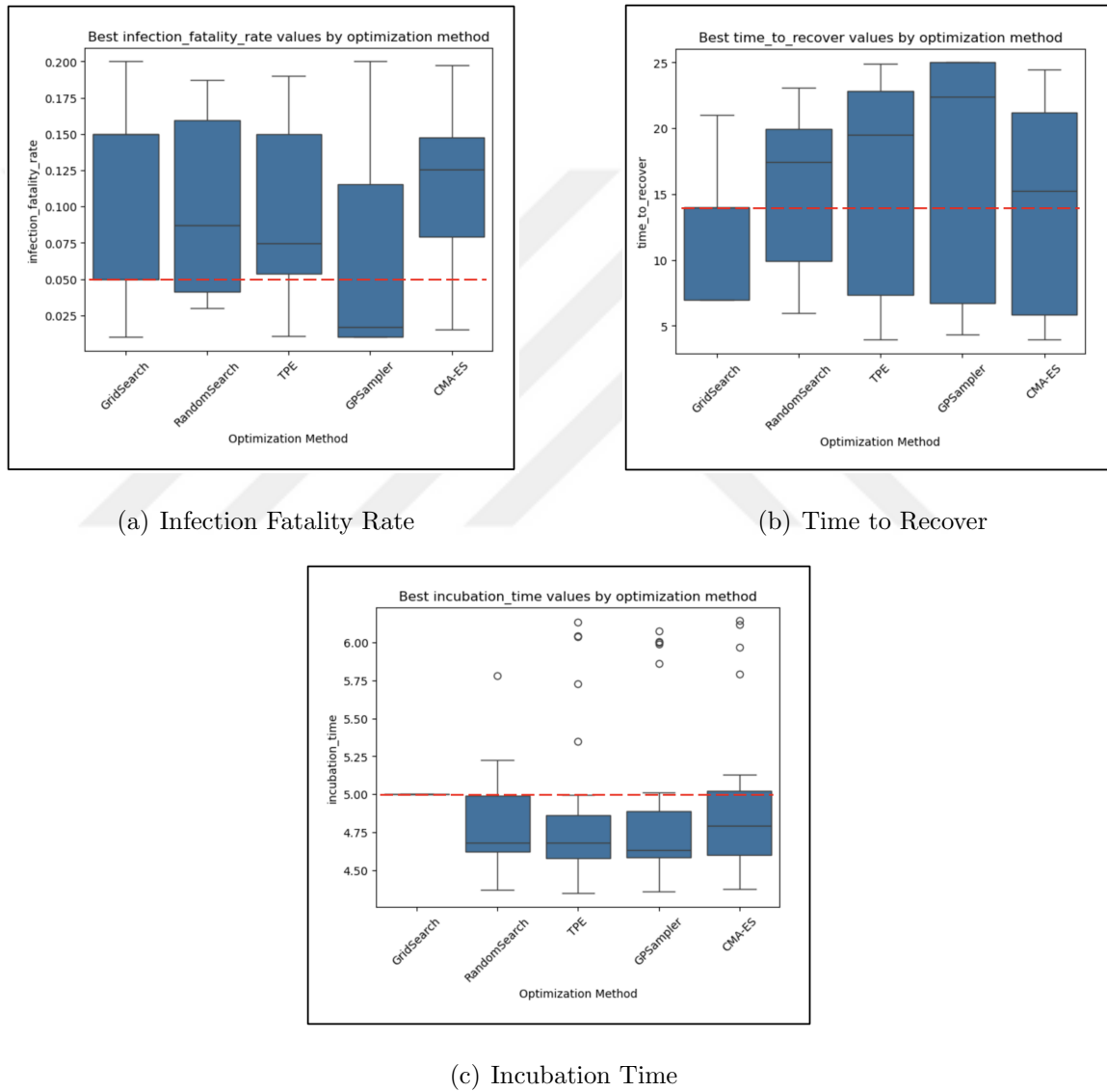


Figure 5.21. Best parameter sets of (a) infection fatality rate, (b) time to recover, and (c) incubation time in autocorrelated noise - strong case.

Figure 5.22 compares the MSE and vector distance performance metrics, providing an aggregated view of how each optimization method performed under strong

autocorrelated noise. In this scenario, TPE, CMA-ES, and GP Sampler obtained comparatively lower MSE values, suggesting better calibration performance than Grid and Random Search.

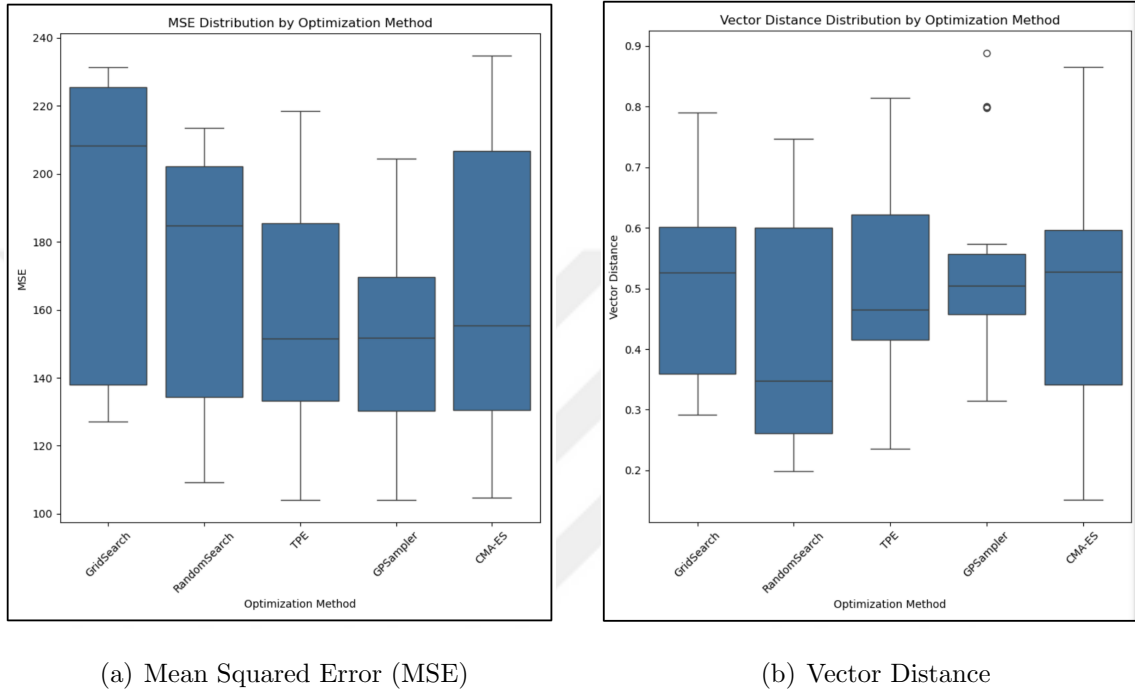


Figure 5.22. (a) Mean squared error (MSE), (b) vector distance results of the best parameter sets in autocorrelated noise - strong case - SEIR model.

In this scenario, unlike the previous cases, Figure 5.23 presents the dynamic behaviors of the parameter sets that yielded the lowest MSE values among the best 25 results for each algorithm. While a slight shift is observed between the trajectories of the estimated and true parameters, the overall dynamic behavior is accurately captured, which is generally sufficient for behavioral validation in SD models.

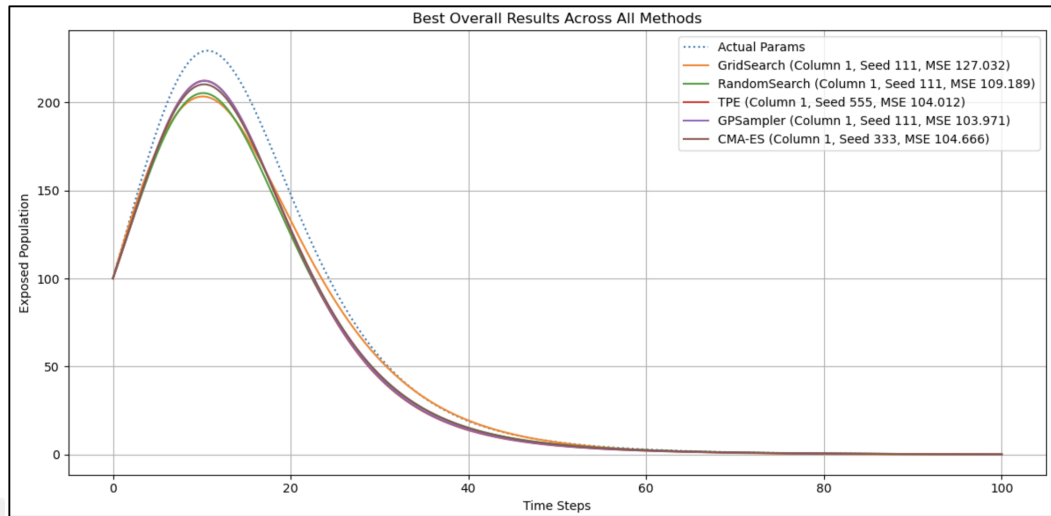


Figure 5.23. Dynamic behaviors from noise seeds that yield the lowest MSE values in autocorrelated noise - strong case - SEIR model.

The heatmap in Figure 5.24 depicts the vector distances between estimated and actual parameters, organized by black-box seeds and optimization techniques in the strong autocorrelated noise scenario. Grid Search produced results with relatively closer vector distances compared to those observed in the mild case.

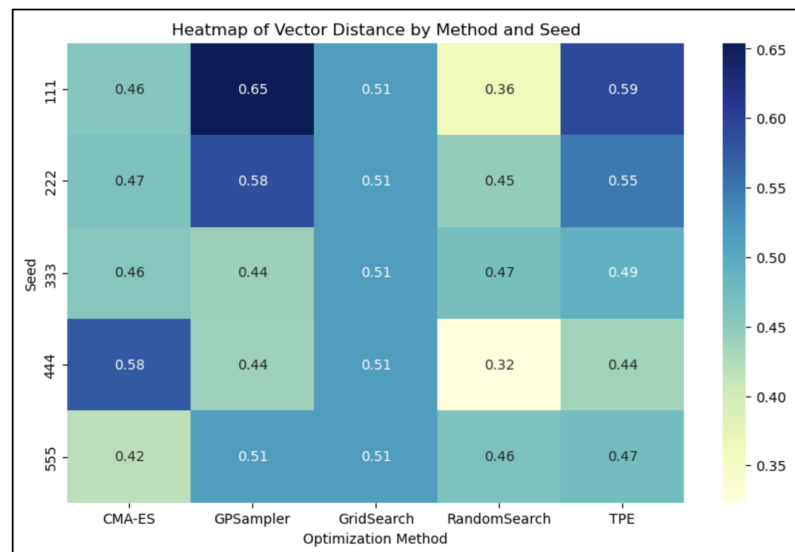


Figure 5.24. Heatmap of vector distance by method and seed in autocorrelated noise - strong case - SEIR model.

Figure 5.25 highlights the number of trials required to identify the best solution for each algorithm under strong autocorrelated noise conditions.

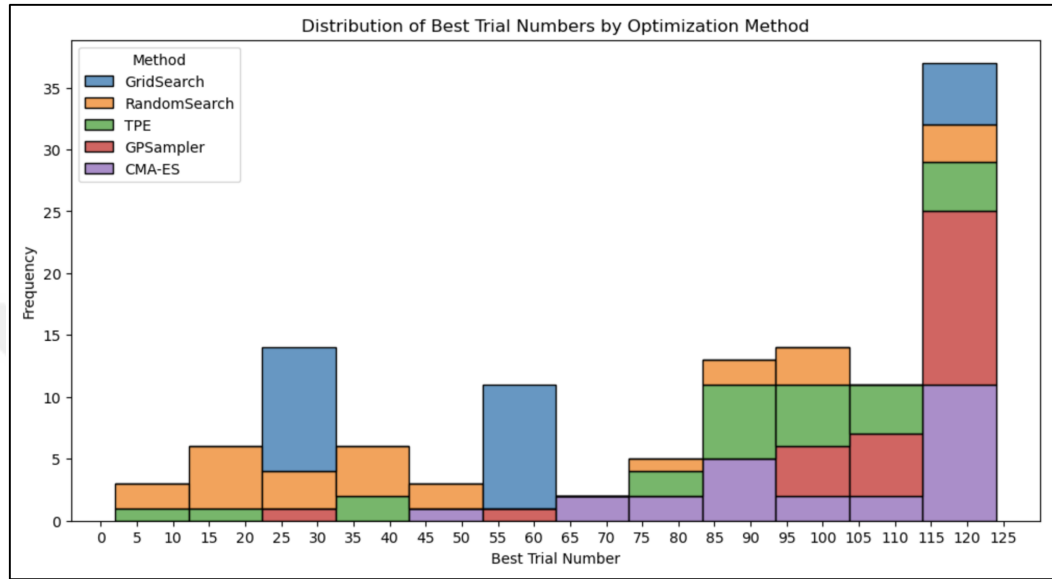


Figure 5.25. Distribution of best trial numbers by optimization method in autocorrelated noise - strong case - SEIR model.

Table 5.10 summarizes the cumulative computational times for each optimization method, enabling a comparison of their efficiency in the strong autocorrelated noise case.

Table 5.10. Execution times for each algorithm in autocorrelated noise - strong case - SEIR model.

	Execution Time
Grid Search	58 seconds
Random Search	54 seconds
TPE	1 minute 7 seconds
GP Sampler	13 minutes 18 seconds
CMA-ES	1 minute

- Strong Pure Noise - Sparse Data Case:

In the strong pure noise - sparse data scenario, synthetic datasets were generated using the same approach as the strong pure noise case. However, the number of data points was significantly reduced by adjusting the simulation settings, resulting in only 10 data points per dataset. Table 5.11 provides the parameter values applied in both noise generation and data sparsification processes.

Table 5.11. Parameters for strong pure noise - sparse data case - SEIR model.

Parameter	Value
<i>Stime</i>	0.125
<i>k</i>	3
<i>SAVEPER</i>	5
<i>FINAL TIME</i>	50

Figure 5.26 compares the exposed and observed exposed population data in the strong pure noise - sparse data case, generated using a fixed SD seed value of 1234.

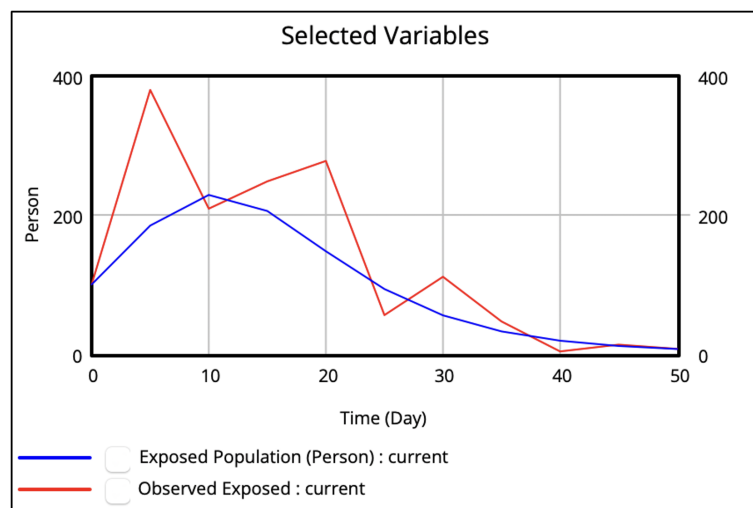
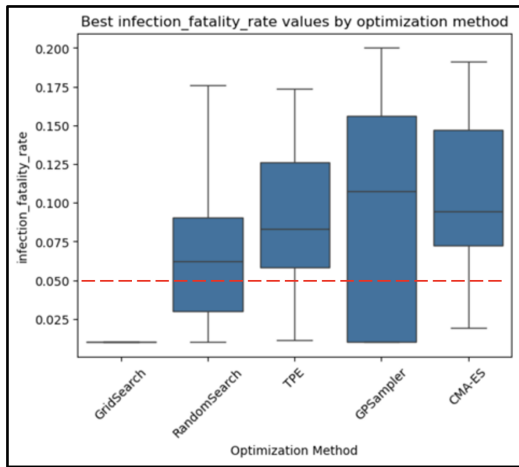
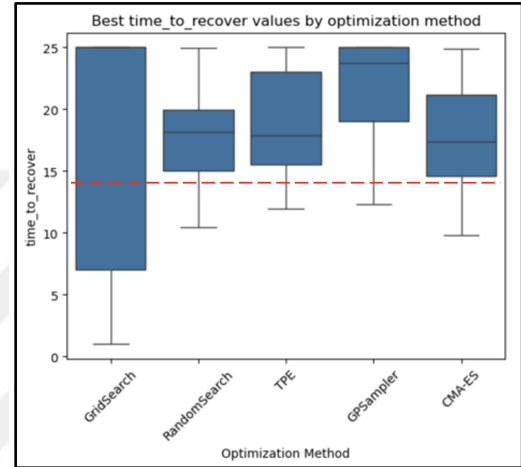


Figure 5.26. Exposed and observed exposed population in strong pure noise - sparse data case.

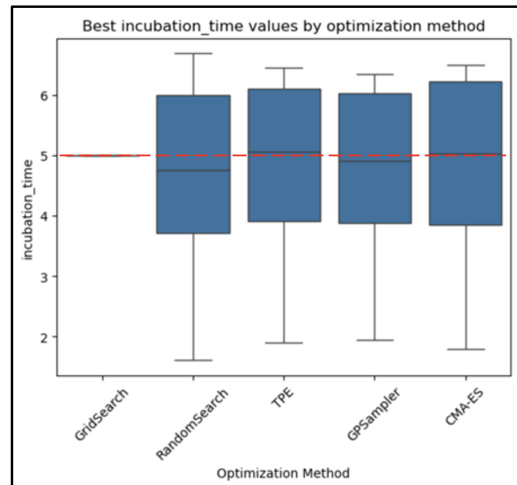
Figure 5.27 presents box plots that summarize the results of 25 experimental runs for each algorithm, focusing on the estimated parameter sets derived using the minimum MSE as the evaluation metric. Most algorithms tended to overestimate the Infection Fatality Rate and Time to Recover compared to the true parameter values.



(a) Infection Fatality Rate



(b) Time to Recover



(c) Incubation Time

Figure 5.27. Best parameter sets of (a) infection fatality rate, (b) time to recover, and (c) incubation time in strong pure noise - sparse data case.

Figure 5.28 showcases the performance metrics, including MSE and vector distance to the actual parameters, providing a comparative analysis across all runs under

limited data conditions. In this case, MSE values are generally higher and more variable across methods.

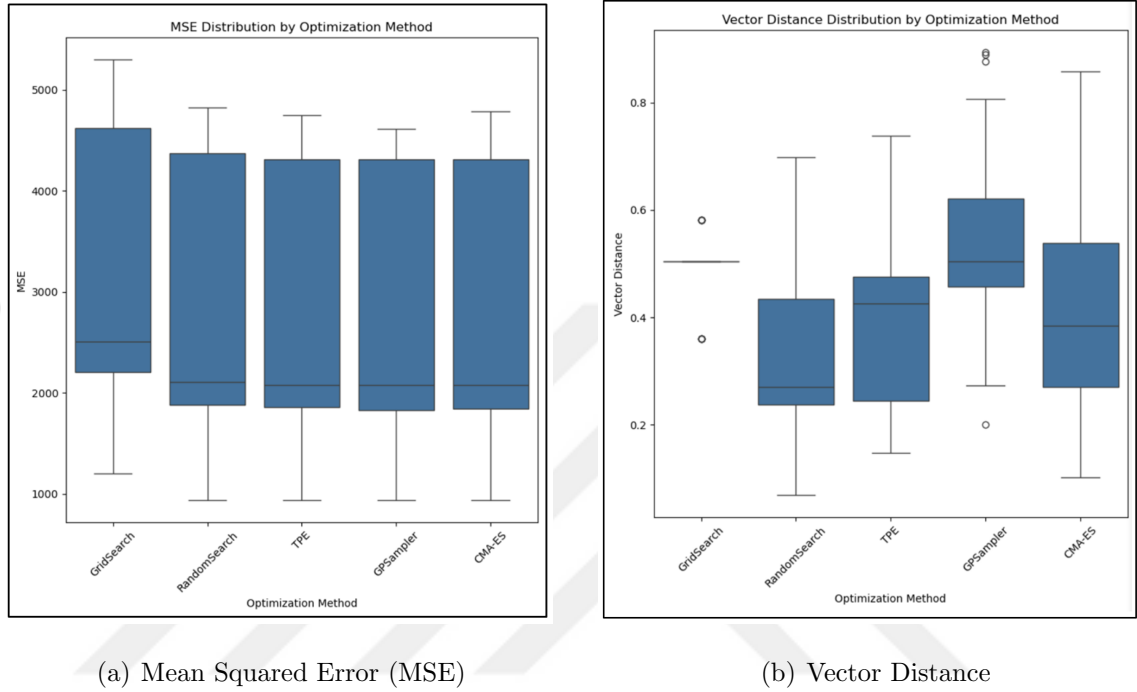


Figure 5.28. (a) Mean squared error (MSE), (b) vector distance results of the best parameter sets in strong pure noise - sparse data case - SEIR model.

Figure 5.29 presents the dynamic behaviors generated by the best parameter sets that yielded the highest MSE on a representative synthetic dataset in the Strong Pure Noise – Sparse Data case (specifically, results from Column 1). Despite slight deviations from the trajectory of the actual parameters, the algorithms were still able to accurately capture the system’s dynamic behavior, even under these challenging conditions.

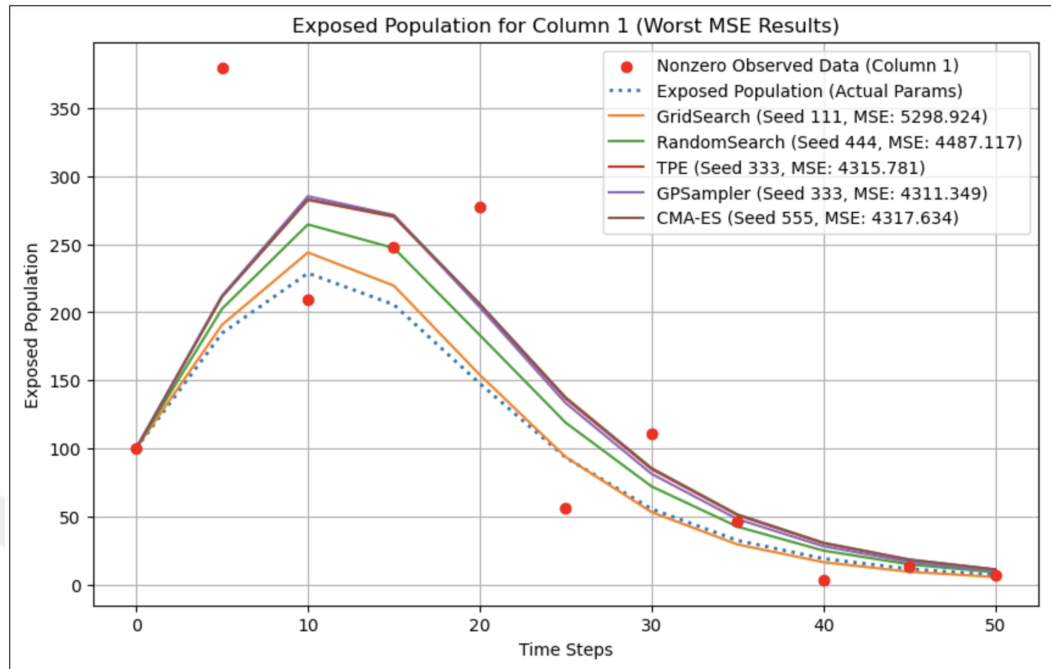


Figure 5.29. Dynamic behaviors from black-box seeds that yield the highest MSE values in strong pure noise - sparse data case - SEIR model.

The heatmap in Figure 5.30 visualizes the vector distances between estimated and true parameters, segmented by black-box seeds and algorithms. GP Sampler yielded higher vector distances compared to the other algorithms.

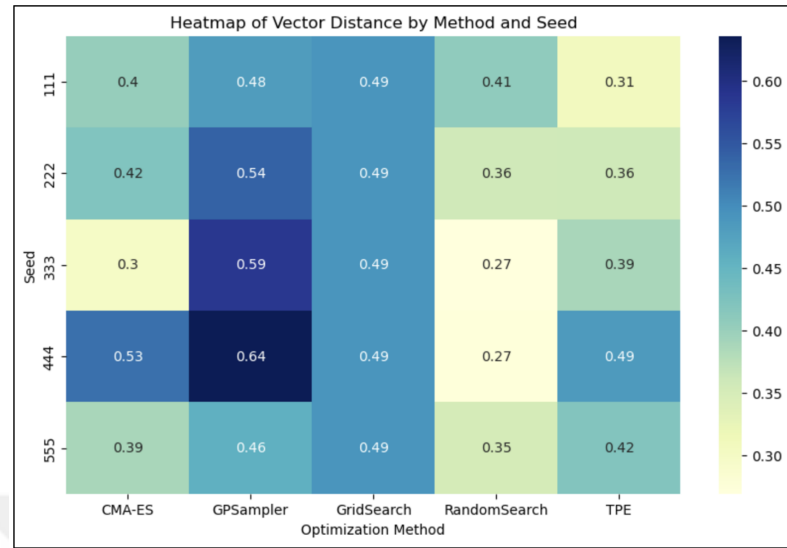


Figure 5.30. Heatmap of vector distance by method and seed in strong pure noise - sparse data case - SEIR model.

Figure 5.31 illustrates the distribution of trials required to identify the best solution for each algorithm. While Random Search typically identified the best result in earlier trials, under the sparse data condition, its performance was more dispersed.

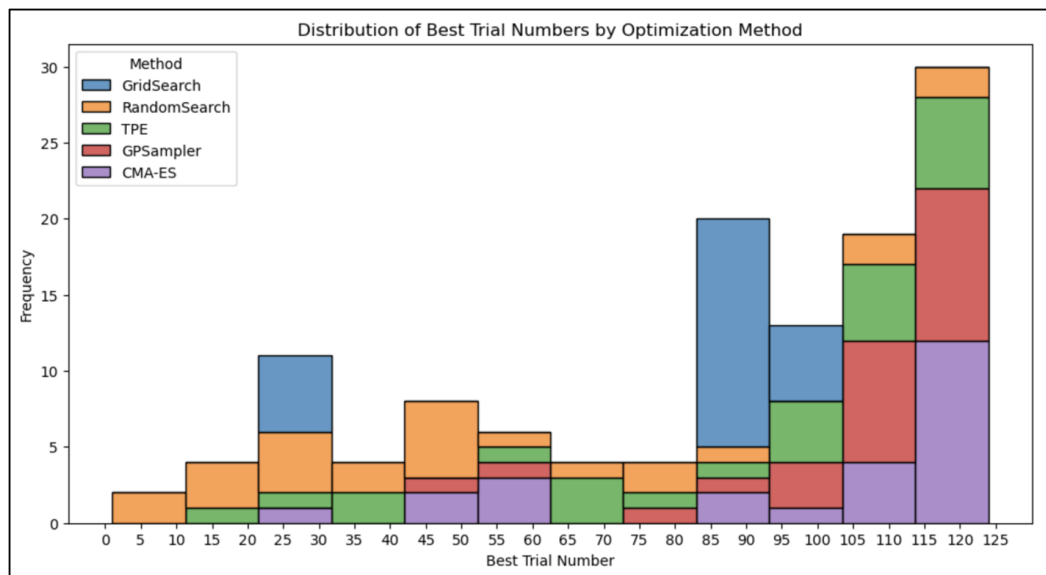


Figure 5.31. Distribution of best trial numbers by optimization method in strong pure noise - sparse data case - SEIR model.

Table 5.12 details the cumulative execution times for each optimization method, providing insights into their computational efficiency when handling strong pure noise with reduced data availability.

Table 5.12. Execution times for each algorithm in strong pure noise - sparse data case - SEIR model.

	Execution Time
Grid Search	20 seconds
Random Search	16 seconds
TPE	29 seconds
GP Sampler	12 minutes 29 seconds
CMA-ES	19 seconds

A detailed evaluation of the results, including comparisons across different models and dynamic scenarios, will be presented in the following sections after the stock management model experiment results.

5.2. Stock Management Model

The stock management model used in this study will be described in the following section, including its structure, key parameters, and flow dynamics. After presenting the model, the procedure for generating synthetic data will be explained along with the experimental setup detailing the scenarios and input parameters. Finally, the experimental results will be presented.

5.2.1. Model Description

The stock management model serves as a classical example in system dynamics, demonstrating key concepts such as feedback control, inventory management, and resource flow regulation. The stock management model, examined through the lens of system dynamics by Forrester in his work on industrial dynamics [2], has been ap-

plied as a modeling approach for analyzing supply chain behaviors and assessing the potential impact of new regulations. The ability of the stock management model to analyze critical phenomena such as demand amplification, where slight fluctuations in consumer demand result in exaggerated variations upstream, has been helpful for understanding supply chain dynamics and identifying potential improvements [87].

Researchers use the stock management model as a tool for investigating complex dynamics such as multilevel supply chains [87] and general supply chain behaviors, while practitioners apply its principles to develop and enhance inventory management strategies [2]. The model's simplicity, adaptability, and resilience align with key principles of SD, such as feedback-driven behaviors, handling nonlinear interactions, and enabling the examination of long-term consequences. These attributes contribute to its continued relevance as a tool for both theoretical investigation and practical application in supply chain management.

The stock management model often exhibits oscillatory dynamics resulting from the interaction of feedback loops, delays, and decision-making rules within the supply chain. These oscillations are induced by efforts to align inventory levels with consumer demand while accounting for manufacturing and replenishment delays [3]. For example, when demand increases unexpectedly, decision makers may over-react by significantly increasing production, resulting in excess inventory. This overshoot leads to corrective measures, such as production reduction, which in turn creates cycles of overstocking and understocking.

Although inventory levels may fluctuate, under certain conditions where production rates are gradually increased to meet rising cumulative demand, the overall behavior of the system may temporarily resemble an S-shaped (sigmoid) growth curve. However, the model more commonly exhibits oscillatory dynamics due to feedback loops and delays.

This research utilizes a traditional stock management model to examine the dynamic relationships among production, inventory, and sales in a supply chain context.

The model encompasses inventory and WIP stages, along with auxiliary stocks such as Stock1 and Expected Sales, which are modified through flows (Flow1, Flow2) and estimated delays (EstDel). These stocks contribute to the model's ability to represent supply chain processes by accounting for anticipated sales and delivery performance. The integration of feedback loops and flow adjustments allows the model to capture some of the complexities present in supply chain dynamics. As such, the model serves as a useful tool for analyzing supply chain operations and evaluating optimization algorithms. The main assumptions of the model are described below:

- **Inventory Dynamics:** Customer demand is assumed to be constant during the simulation, representing a simplified scenario where demand remains unaffected by external factors. Production is dynamically adjusted based on the difference between current inventory levels and the target inventory goal. This adjustment aims to align inventory with expected sales, thereby attempting to mitigate the risks of overstocking and understocking.
- **Standardized Processing:** Work-In-Process (WIP) items progress through the production pipeline with fixed delay times assigned to each stage. This assumption implies that all items experience consistent processing delays, promoting uniform throughput during each stage. The model assumes that production resources are consistently applied across WIP stages, aiming to prevent bottlenecks and maintain steady progress.
- **Flow and Stock Updates:** Most flows, including production, sales, and inventory adjustments, are regulated by equations that are primarily linear. The general linearity of the system simplifies the analysis of oscillatory dynamics and delays. Continuous updates of stocks, inventory, work-in-progress stages, and expected sales allow for dynamic adjustments in production and inventory levels to address potential oscillatory behaviors. The model operates under deterministic conditions, where flows and stocks are calculated accurately based on predefined equations, without incorporating stochastic variation.

The stock management model is represented by equations that describe the interactions among stocks, flows, and auxiliary variables. These equations capture the

dynamics of production, inventory, and sales processes, providing a structured approach for analyzing the model's behavior. The model is defined by the following equations.

The model assumes a constant customer demand, which drives the supply chain operations. The sales flow depends on the available inventory and a sales fraction parameter (*SalesT*), which determines the proportion of inventory that can be sold in each time period. Sales are calculated as the minimum of available inventory and customer demand

$$Sales = Min(Inventory \times SalesT, Demand). \quad (5.11)$$

This relationship indicates that sales are constrained by inventory availability and customer demand, reflecting practical operational limitations within the model.

Inventory dynamics play a crucial role in balancing production and sales. Inventory is updated based on the inflow from the final stage of production (*Finishing3*) and the outflow resulting from sales

$$\frac{d(Inventory)}{dt} = Finishing3 - Sales. \quad (5.12)$$

An inventory goal establishes a target level for production adjustments, aiming to align inventory levels with constant customer demand. While demand remains constant, sales may fluctuate depending on inventory availability, and the inventory goal serves to mitigate the risk of shortages or excess accumulation.

The production process is influenced by current inventory levels and expected sales. Production requirements are determined by evaluating the inventory deficit and aligning production with anticipated sales to maintain the inventory goal

$$DesProduction = (InvGoal - Inventory) \times IAT + Expected\ Sales. \quad (5.13)$$

The inventory adjustment time determines the rate at which inventory discrepancies are addressed; in other words, it represents how quickly the system reacts to deviations from the desired inventory level. While the production rate aims to reduce inventory deficits and restore balance, it is inherently constrained by the maximum available production capacity

$$Production = \text{Max}(\text{Min}(PCapac, DesProduction), 0). \quad (5.14)$$

This guarantees that production levels remain within sustainable limits defined by production capacity and are consistent with inventory and demand objectives.

The model includes a multi-stage work-in-process (WIP) system, representing the sequential steps of production. Each stage is associated with a specific processing delay ($PDel1$, $PDel2$, $PDel3$), and the flow between stages is calculated as follows

$$\frac{d(WIP1)}{dt} = Production - Finishing1, \quad (5.15)$$

$$Finishing1 = \frac{WIP1}{PDel1}, \quad (5.16)$$

$$\frac{d(WIP2)}{dt} = Finishing1 - Finishing2, \quad (5.17)$$

$$Finishing2 = \frac{WIP2}{PDel2}, \quad (5.18)$$

$$\frac{d(WIP3)}{dt} = Finishing2 - Finishing3, \quad (5.19)$$

$$Finishing3 = \frac{WIP3}{PDel3}. \quad (5.20)$$

The equations capture the processing delays and dependencies within the production process, which play a crucial role in generating the oscillatory behavior observed in inventory levels and production rates.

The model utilizes expected sales to estimate inventory adjustments and guide production planning, aiming to maintain alignment between production levels and anticipated sales. Expected sales are continuously updated through

$$\frac{d(\textit{Expected Sales})}{dt} = \textit{Flow2} = \frac{\textit{Stock1} - \textit{Expected Sales}}{\textit{EstDel}}. \quad (5.21)$$

where *EstDel* denotes the duration necessary for adjusting sales expectations. This dynamic aims to align production planning with anticipated sales levels, considering the delayed response to inventory changes.

The equations collectively describe the model's behavior, capturing the interactions among production, inventory, and sales processes within a supply chain system.

The stock management model provides a structured representation for analyzing production, inventory, and sales dynamics within a supply chain system. It tracks the flow of items through various production stages, represented by WIP stocks, until they reach inventory and satisfy customer demand. The model's behavior is shaped by feedback loops, inventory objectives, and delay mechanisms, which together create dynamic patterns such as oscillations in inventory levels and production rates. Nonlinearities arising from production adjustments and capacity constraints further complicate the system's dynamics. This combination of interacting components results in a complex optimization landscape, making the model a suitable testbed for evaluating BBO methods.

5.2.2. Experimental Design and Results

The experiments with the stock management model aimed to calibrate key operational parameters, PDel1, PDel2, and PDel3, to ensure the model outputs accurately reflected synthetic datasets generated under controlled noise conditions. Similar to the SEIR model experiments, noise was introduced to simulate real-world observational uncertainties, particularly affecting the Observed Inventory variable. Different noise patterns, including white and pink noise, were generated by varying the smoothing

constant $Stime$ and the scaling factor k . These patterns allowed for the evaluation of BBO methods under various noise conditions to assess their robustness and accuracy.

As in the SEIR experiments, five synthetic datasets were generated for each noise scenario using five random seed values to account for variability. This setup resulted in a total of 25 experimental runs per case, enabling a comprehensive assessment of algorithm performance across different initial conditions. All runs were conducted using the Optuna library (v4.0.0), and GP Sampler employed the EI acquisition function, consistent with the SEIR model configuration. The Observed Inventory variable served as the benchmark for calculating MSE, which was used to measure the accuracy of parameter calibration. The stock management model was treated as a black box, with BBO methods aimed at minimizing the discrepancies between noisy observed data and model outputs to evaluate their effectiveness.

The parameter ranges used for Grid Search and other algorithms in the stock management model are presented in Table 5.13. The actual parameter values used to generate the synthetic datasets are listed in Table 5.14.

Table 5.13. Parameter ranges for stock management model.

Parameter	Grid Search	Other Algorithms
$PDel1$	[0.5, 5, 10, 15, 20]	(0.5, 20)
$PDel2$	[0.5, 5, 10, 15, 20]	(0.5, 20)
$PDel3$	[0.5, 5, 10, 15, 20]	(0.5, 20)

Table 5.14. Actual parameter values for stock management model.

Parameter	Actual Value
$PDel1$	6
$PDel2$	5
$PDel3$	5

These parameter values served as reference points for evaluating the accuracy of the optimization results, following a similar approach to that used in the SEIR model.

Visualizations were employed to assess the performance of BBO algorithms across various noise scenarios, following a similar procedure as the SEIR model experiments. Box plots of parameter value distributions, MSE, and vector distances were used to evaluate the accuracy, consistency, and robustness of the algorithms in approximating actual parameter values. Vector distance heat maps revealed calibration accuracy patterns across noise scenarios and seed values, while histograms of trials to the best solutions indicated convergence efficiency by highlighting the speed of identifying best parameter sets. Execution times were also compared to evaluate computational efficiency. This consistent evaluation approach enabled direct performance comparisons across different models.

The study employs a consistent evaluation procedure and metrics, including MSE, vector distance, convergence efficiency, and execution time, across both models. This approach ensures comparability and coherence in assessing BBO techniques under diverse conditions, including varying noise scenarios and seed values. The subsequent sections examine the detailed results and case analyses related to the stock management model.

- Pure Noise Case:

Synthetic datasets were generated under the pure noise scenario using uncorrelated white noise to simulate real-world measurement errors without temporal dependencies. This noise was uniformly applied to the Observed Inventory variable in the stock management model to create purely stochastic conditions. The objective of these experiments was to evaluate the effectiveness of BBO methods in a baseline scenario, providing a reference point for comparison with more complex noise conditions involving autocorrelation. To eliminate autocorrelation and maintain pure white noise characteristics, the smoothing constant $Stime$ was set equal to the model's dt value. This configuration ensures that noise is updated at each simulation step independently,

without introducing temporal dependencies. The values of the noise generation parameters are listed in Table 5.15.

Table 5.15. Smoothing parameters for pure noise case - stock management model.

Parameter	Value
$Stime$	0.125
k	1

Figure 5.33 illustrates the inventory levels alongside the observed inventory data, generated using a fixed SD seed value of 1234 under the pure noise scenario.

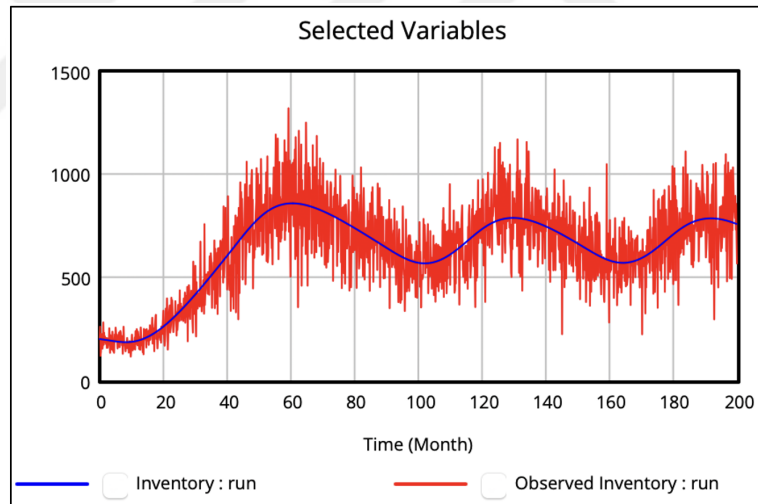


Figure 5.33. Inventory and observed inventory in pure noise case.

Figure 5.34 presents the box plots of the 25 experiments conducted by each algorithm to identify the best parameter sets, determined by the MSE. Grid Search successfully estimated two of the parameters and consistently arrived at the same result across all trials. Meanwhile, Random Search exhibited noticeably higher variability compared to the other algorithms.

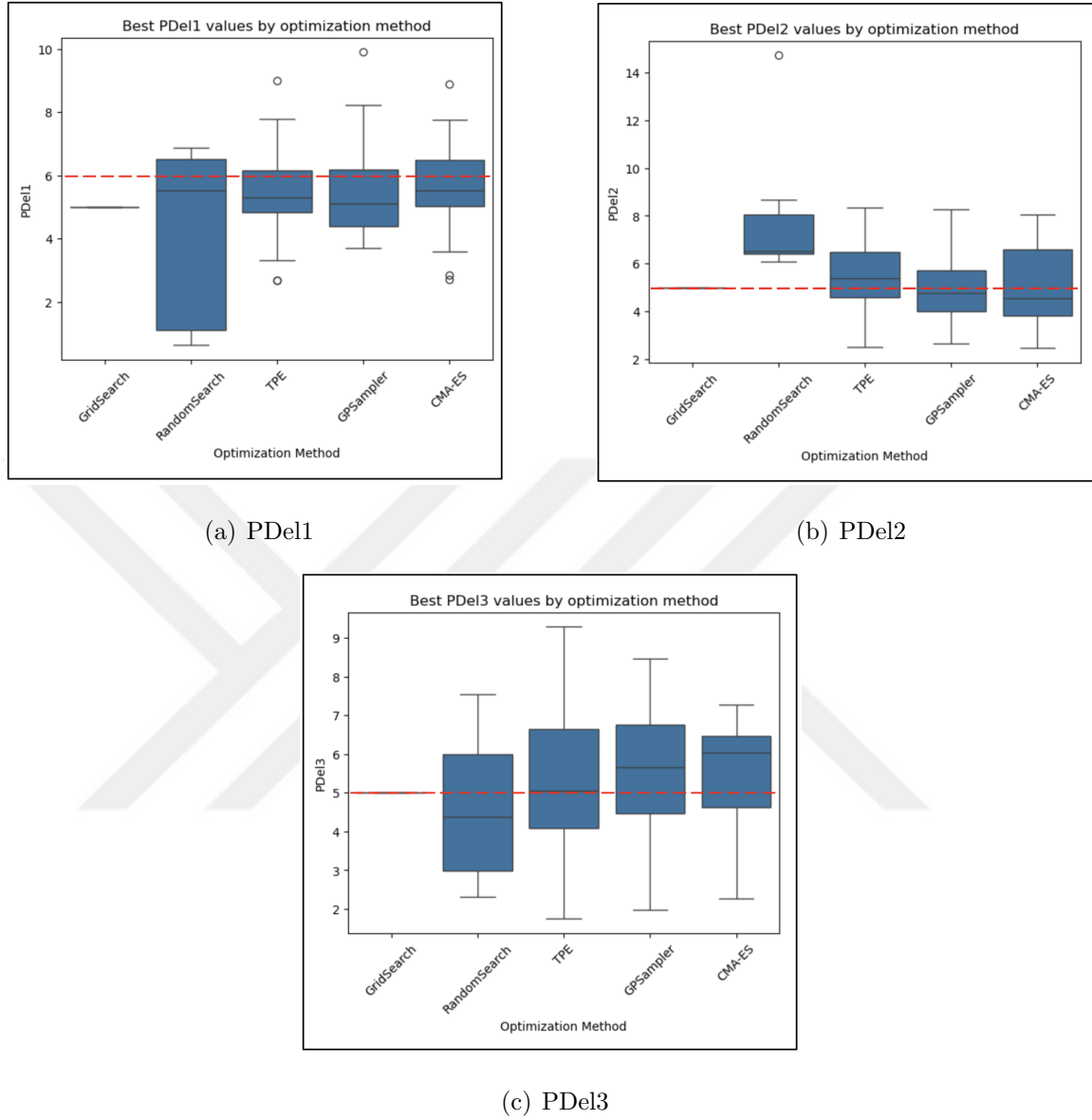


Figure 5.34. Best parameter sets of (a) PDel1, (b) PDel2, and (c) PDel3 in pure noise case.

Figure 5.35 compares key performance metrics, including MSE and vector distance to the actual parameter values, across all runs for each BBO method. While Grid Search achieved low vector distances, these parameter sets were often linked to very high MSE values. Random Search produced more variable results and generally higher MSE values compared to the other methods.

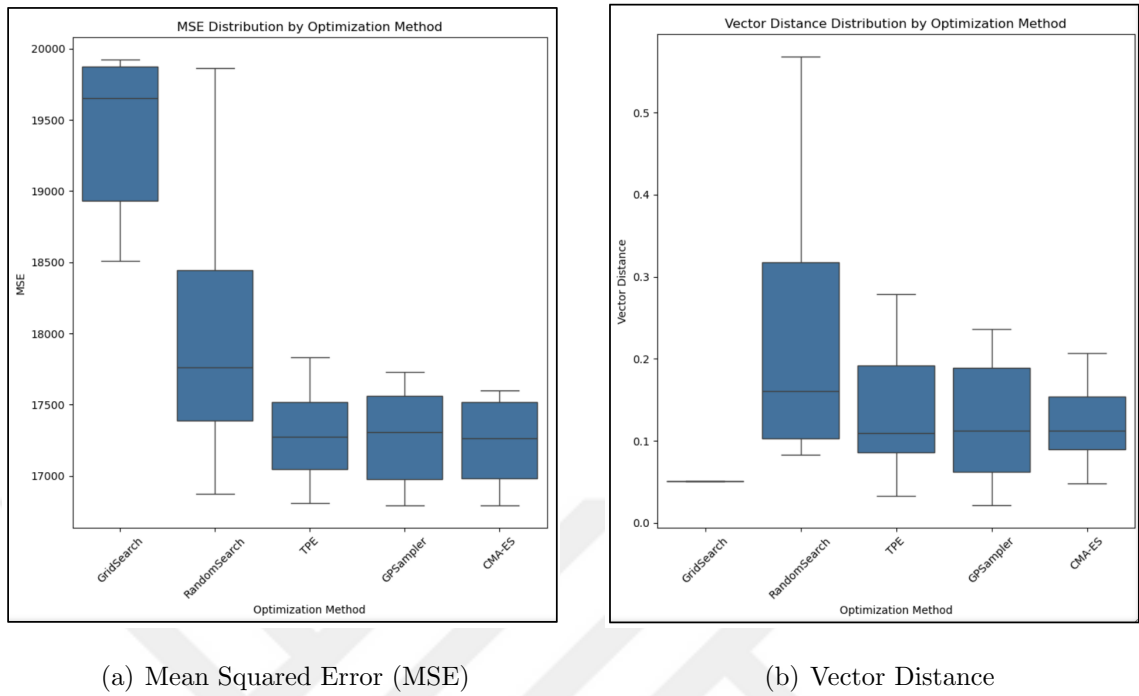


Figure 5.35. (a) Mean squared error (MSE), (b) vector distance results of the best parameter sets in pure noise case - stock management model.

Figure 5.36 illustrates the dynamic behavior of the parameter sets with the highest MSE values among the top 25 results for each algorithm. Unlike in the SEIR model, not all methods were able to accurately replicate the system's dynamic behavior. While the oscillatory pattern is captured, the results of Grid Search and Random Search show visible deviations from the actual system behavior.

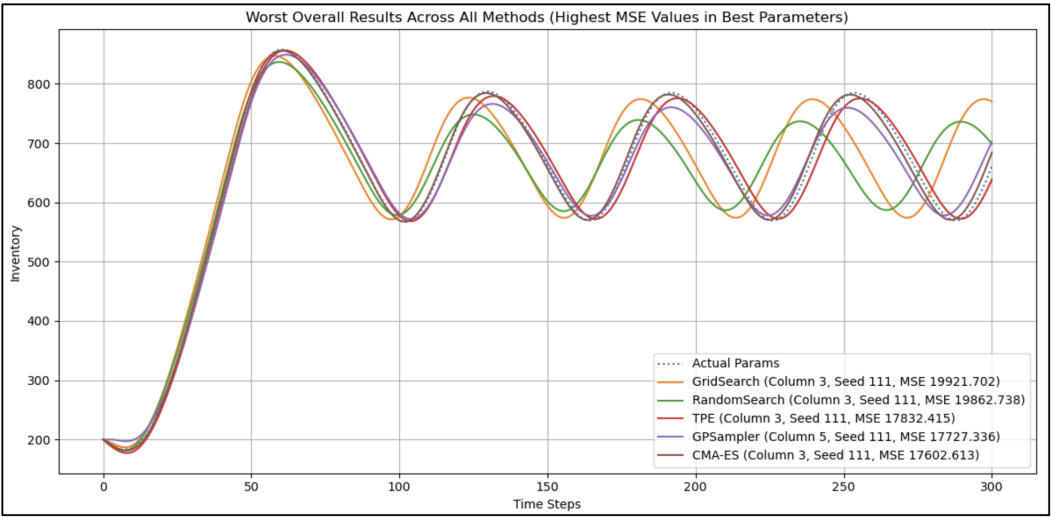


Figure 5.36. Dynamic behaviors from noise seeds that yield the highest MSE values in pure noise case - stock management model.

The heatmap depicts the vector distances between the estimated and actual parameter sets (Figure 5.37). All methods except Random Search consistently achieved low vector distances across different seeds. Random Search exhibited notably higher variation and larger deviations.

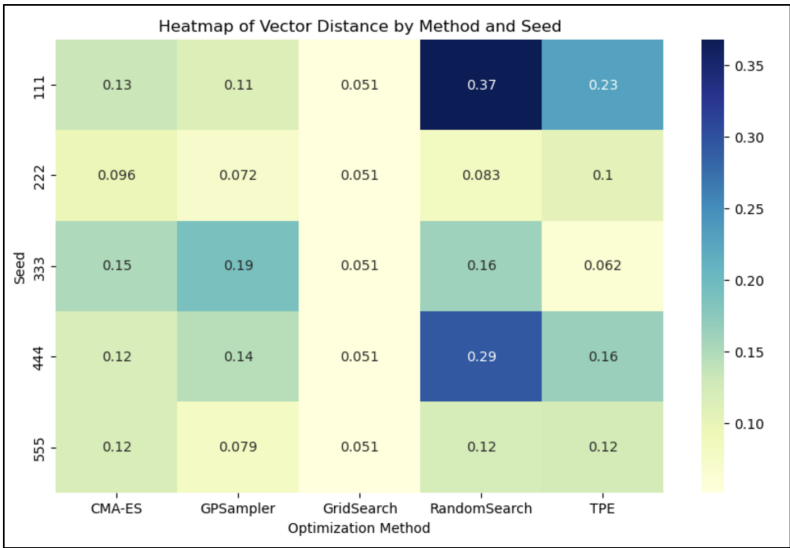


Figure 5.37. Heatmap of vector distance by method and seed in pure noise case - stock management model.

Figure 5.38 displays the distribution of trials at which each BBO method identified the best solution within the pure noise scenario.

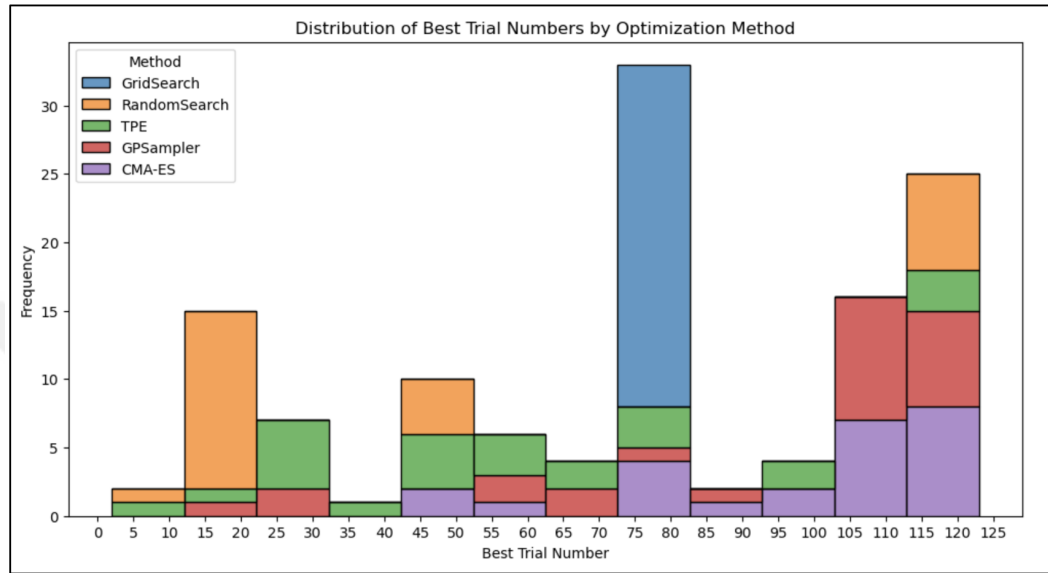


Figure 5.38. Distribution of best trial numbers by optimization method in pure noise case - stock management model.

Table 5.16 summarizes the cumulative execution times for each BBO method, highlighting their computational performance in the pure noise scenario.

Table 5.16. Execution times for each algorithm in pure noise case - stock management model.

	Execution Time
Grid Search	1 minute 59 seconds
Random Search	1 minute 56 seconds
TPE	2 minutes 7 seconds
GP Sampler	11 minutes 7 seconds
CMA-ES	1 minute 59 seconds

- Strong Pure Noise Case:

In the strong pure noise scenario, synthetic datasets were generated by applying amplified white noise to the Observed Inventory variable. The purpose of this approach was to evaluate the robustness and accuracy of BBO methods under heightened noise levels. To generate amplified noise, the scaling factor k was increased to enhance the intensity of the white noise. To prevent temporal dependencies and ensure pure white noise characteristics, the smoothing constant $Stime$ was set equal to the model's dt value. Table 5.17 presents the values of the noise generation parameters.

Table 5.17. Smoothing parameters for strong pure noise case - stock management model.

Parameter	Value
$Stime$	0.125
k	1.5

Figure 5.39 shows the inventory values alongside the observed inventory data, generated under the strong pure noise scenario with a fixed SD seed value of 1234.

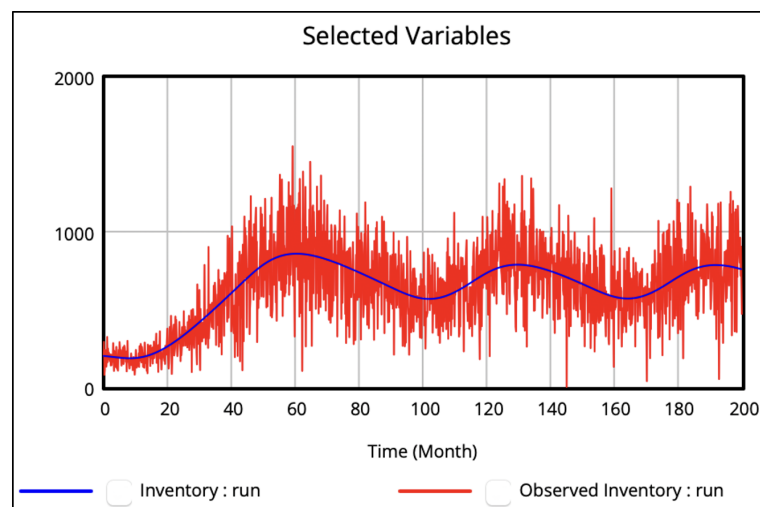
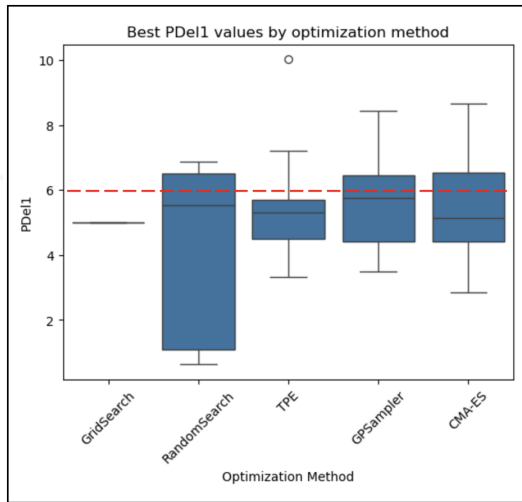
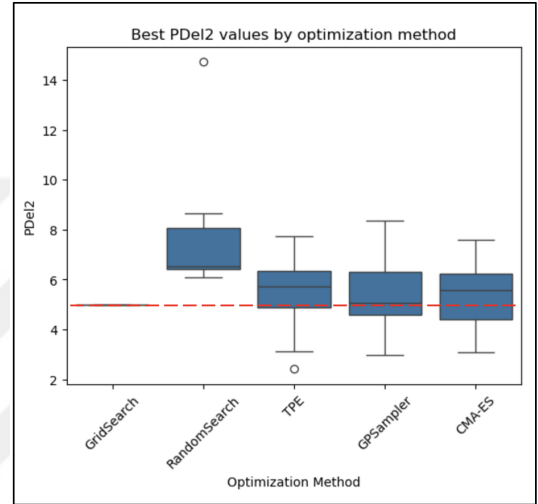


Figure 5.39. Inventory and observed inventory in strong pure noise case.

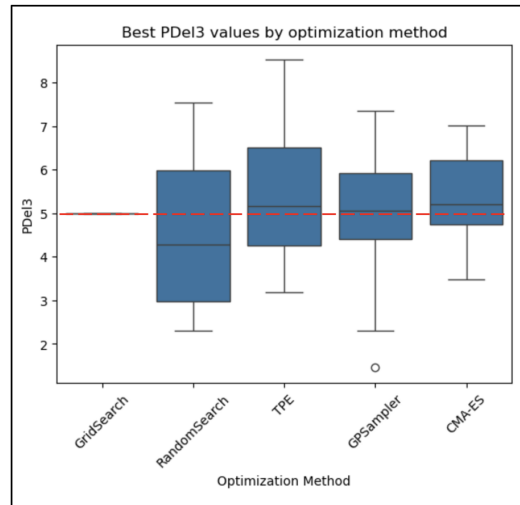
Figure 5.40 presents the box plots summarizing the results of 25 runs for each algorithm, identifying the best parameter sets based on the MSE. Grid Search again produced identical results across all trials, while Random Search maintained its variability.



(a) PDe1



(b) PDe2



(c) PDe3

Figure 5.40. Best parameter sets of (a) PDe1, (b) PDe2, and (c) PDe3 in strong pure noise case.

Figure 5.41 provides a comparison of MSE and vector distance to the actual parameters. The results show similarities with those observed in the pure noise case.

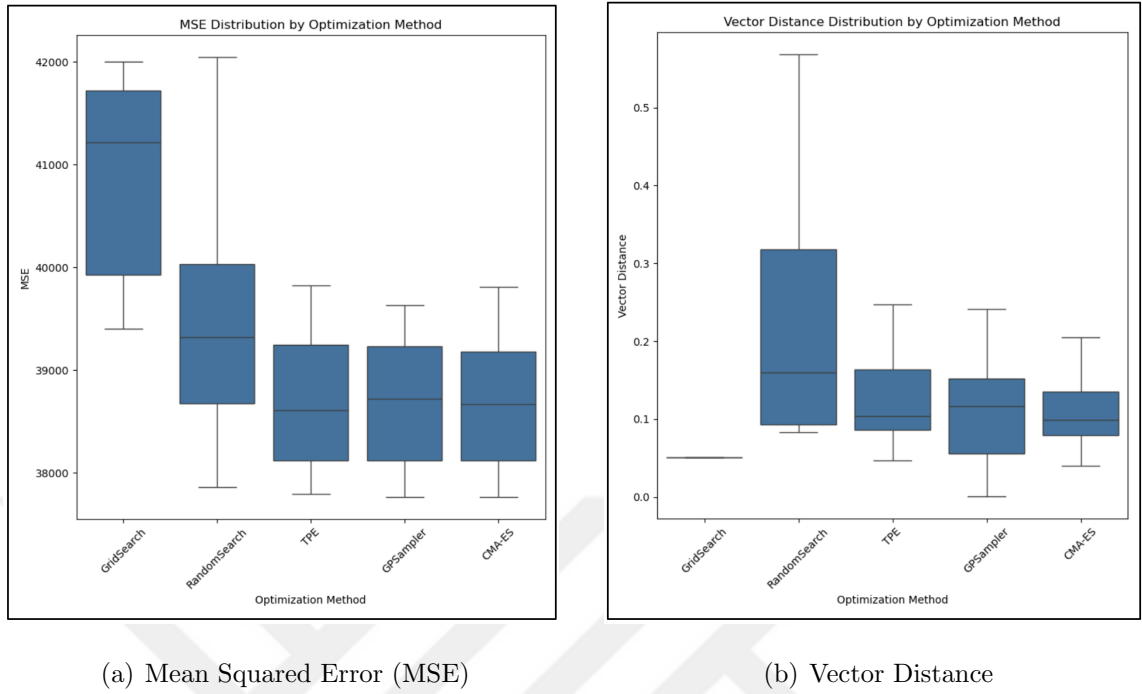


Figure 5.41. (a) Mean squared error (MSE), (b) vector distance results of the best parameter sets in strong pure noise case - stock management model.

As seen in Figure 5.42, the dynamic behavior results closely resemble those observed in the pure noise case. Similarly to the previous scenario, although the overall oscillatory pattern is preserved, noticeable deviations from the true system behavior were still observed, particularly in the outputs of Grid Search and Random Search.

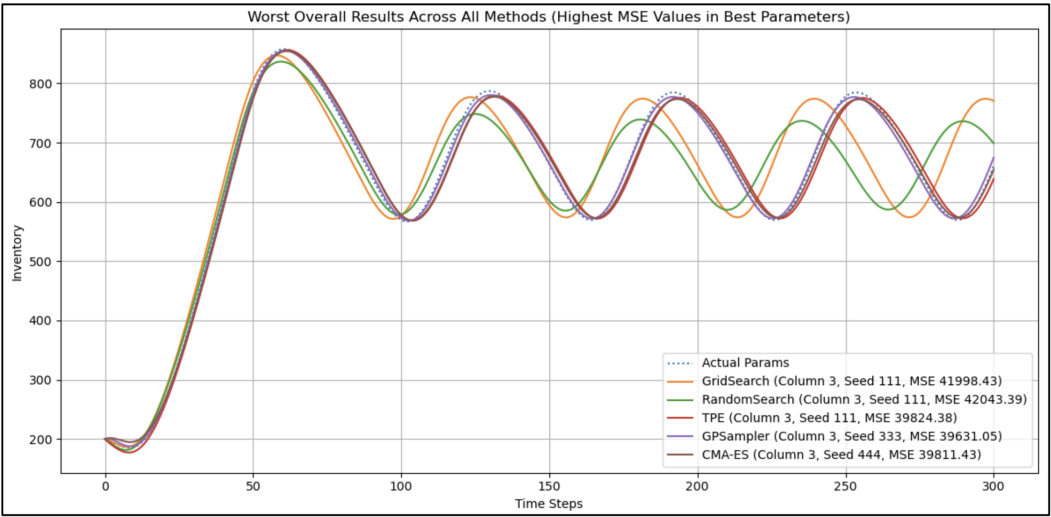


Figure 5.42. Dynamic behaviors from noise seeds that yield the highest MSE values in strong pure noise case - stock management model.

The heatmap in Figure 5.43 visualizes the vector distances between the estimated and true parameter sets. Aside from Grid Search, CMA-ES performed well in this case.

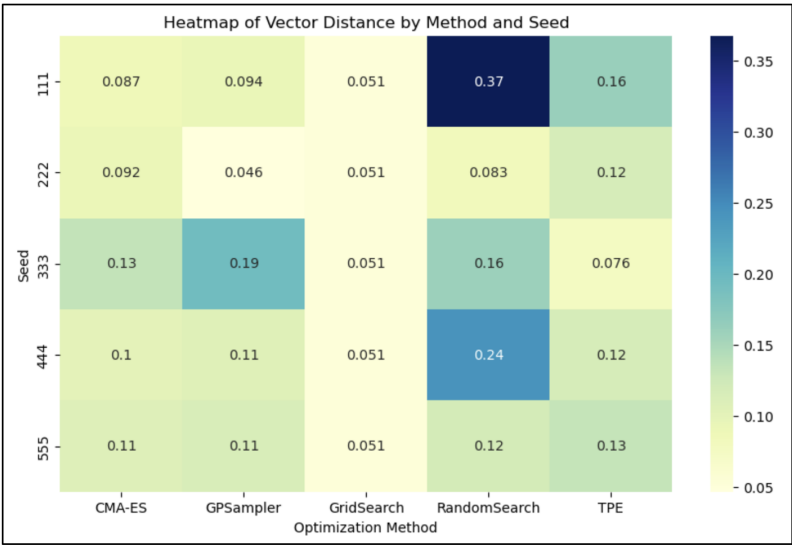


Figure 5.43. Heatmap of vector distance by method and seed in strong pure noise case - stock management model.

Figure 5.44 shows the distribution of trials required to locate the best parameters. Unlike Grid Search, the best trials vary across runs for other methods.

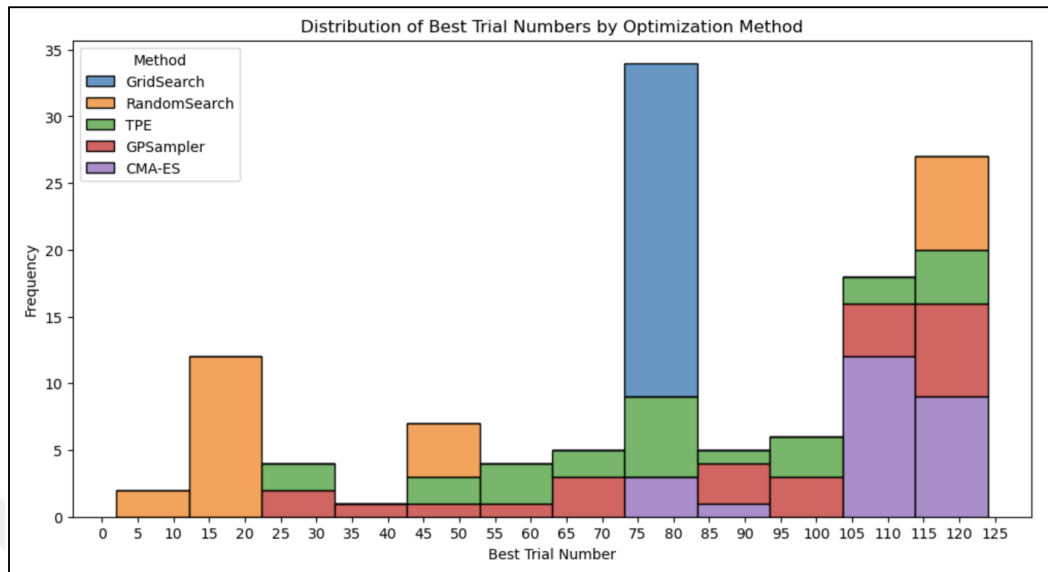


Figure 5.44. Distribution of best trial numbers by optimization method in strong pure noise case - stock management model.

Table 5.18 details the cumulative execution durations for each BBO algorithm, providing insights into their computational efficiency in the strong pure noise case. Among the tested algorithms, Random Search achieved the shortest execution time while GP Sampler required a significantly longer runtime compared to the other algorithms.

Table 5.18. Execution times for each algorithm in strong pure noise case - stock management model.

	Execution Time
Grid Search	1 minute 59 seconds
Random Search	1 minute 55 seconds
TPE	2 minutes 6 seconds
GP Sampler	11 minutes 8 seconds
CMA-ES	2 minutes

- Autocorrelated Noise - Mild Case:

In the autocorrelated noise scenario, synthetic datasets were generated using noise with temporal dependencies. This type of noise, often referred to as pink noise, was applied to the Observed Inventory variable. The parameters k and $Stime$ were deliberately adjusted to generate autocorrelated noise by introducing temporal smoothing. The scaling factor k controlled the intensity of the noise, while the smoothing constant $Stime$ regulated the extent of temporal dependencies. Noise generation parameter values are shown in Table 5.19.

Table 5.19. Smoothing parameters for autocorrelated noise - mild case - stock management model.

Parameter	Value
$Stime$	3
k	3

Figure 5.45 illustrates the inventory dynamics alongside the observed inventory data, generated using mild autocorrelated noise with a fixed SD seed value of 1234.

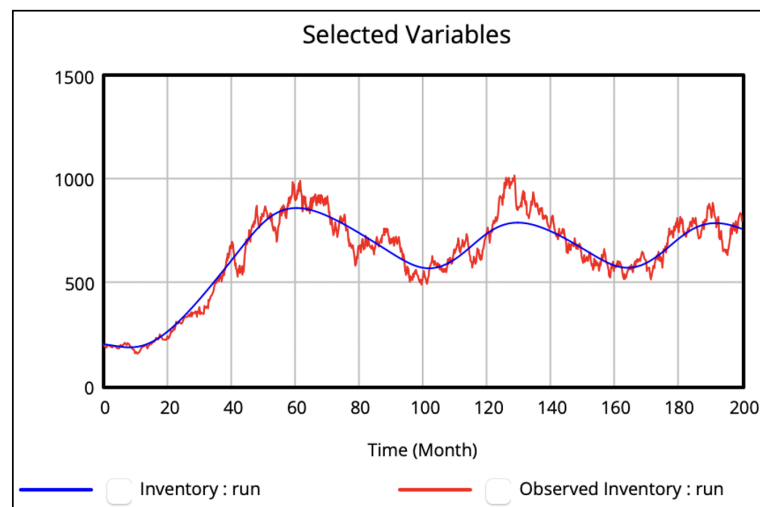
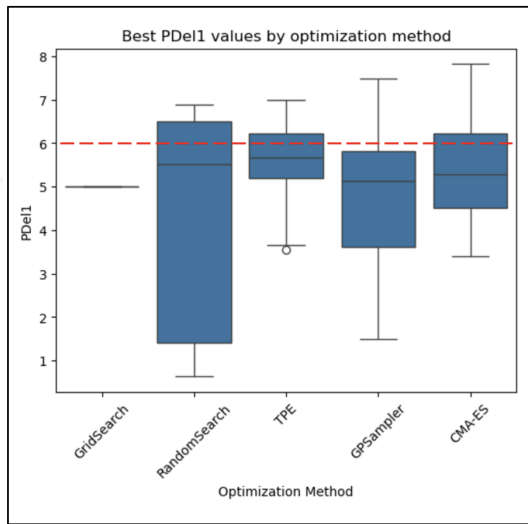
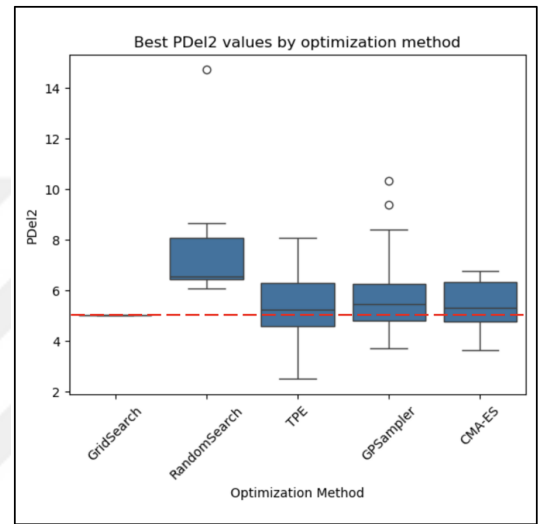


Figure 5.45. Inventory and observed inventory in autocorrelated noise - mild case.

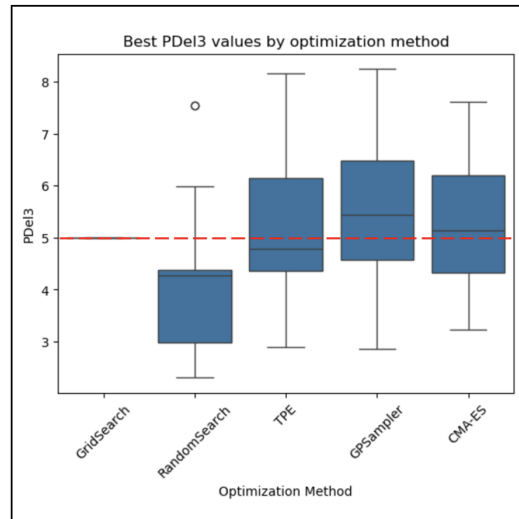
Figure 5.46 showcases the box plots summarizing the parameter estimates from 25 experiments conducted by each optimization method, evaluated based on MSE. Grid Search maintained identical outputs across runs, and both CMA-ES and TPE continued to yield favorable results.



(a) PDel1



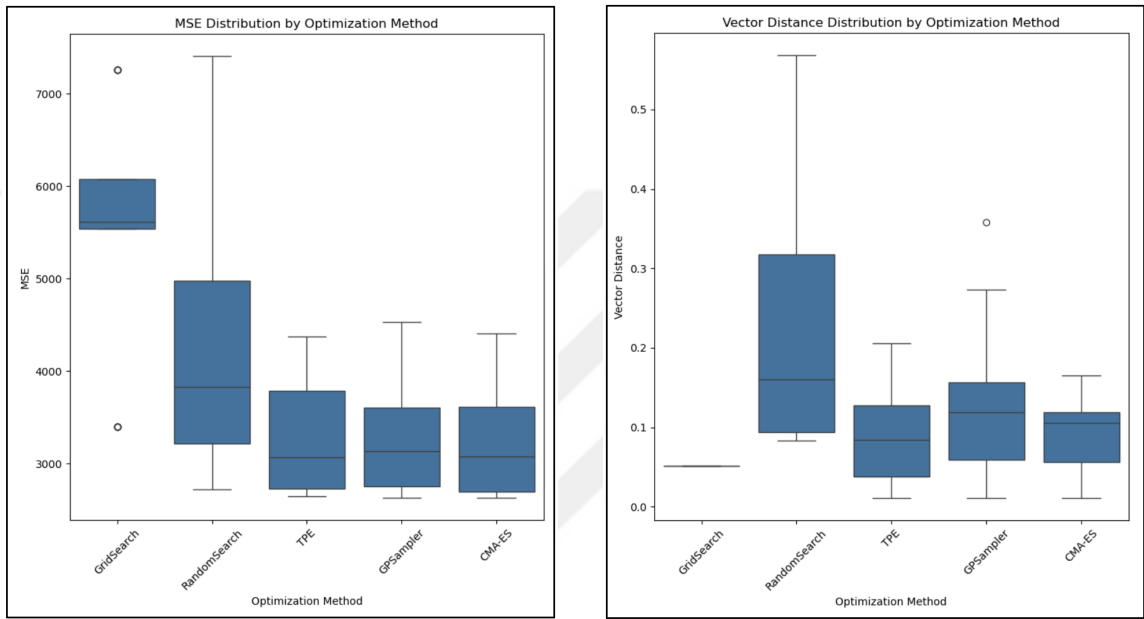
(b) PDel2



(c) PDel3

Figure 5.46. Best parameter sets of (a) PDel1, (b) PDel2, and (c) PDel3 in autocorrelated noise - mild case.

Figure 5.47 compares performance metrics, such as MSE and vector distance to the true parameters, across all BBO methods under mild autocorrelated noise conditions. It is noteworthy that GP Sampler produced more consistent results in this model compared to the SEIR model.



(a) Mean Squared Error (MSE)

(b) Vector Distance

Figure 5.47. (a) Mean squared error (MSE), (b) vector distance results of the best parameter sets in autocorrelated noise - mild case - stock management model.

Among the parameter sets with the highest MSE values within the best 25 results for each algorithm, CMA-ES, TPE, and GP Sampler were able to successfully replicate the model's dynamic behavior (Figure 5.48). In contrast, although Random Search and Grid Search captured the oscillatory pattern, they exhibited visible deviations similar to the other cases.

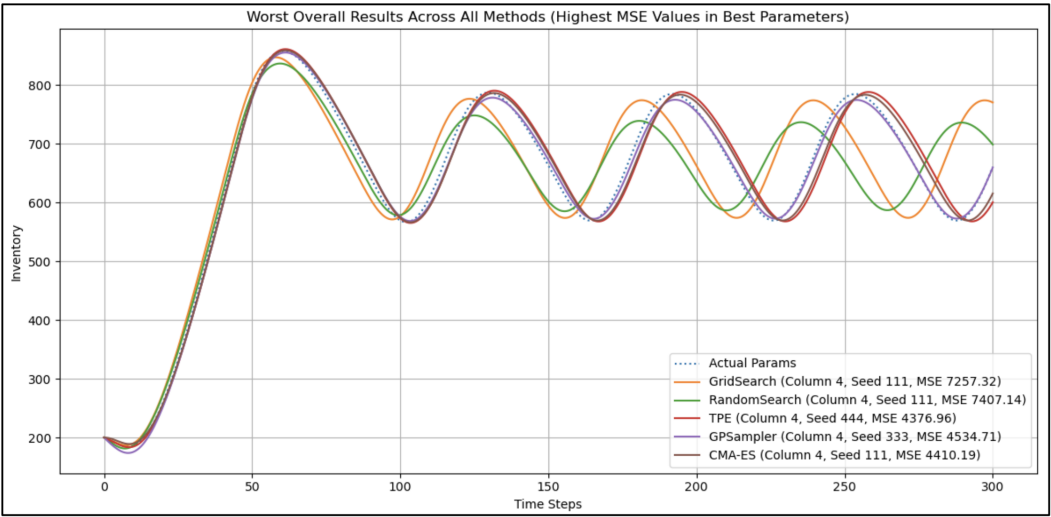


Figure 5.48. Dynamic behaviors from noise seeds that yield the highest MSE values in autocorrelated noise - mild case - stock management model.

The heatmap in Figure 5.49 displays the vector distances between the calibrated and actual parameter sets, segmented by black-box seeds and optimization techniques. Alongside Grid Search, both CMA-ES and TPE also achieved favorable results.

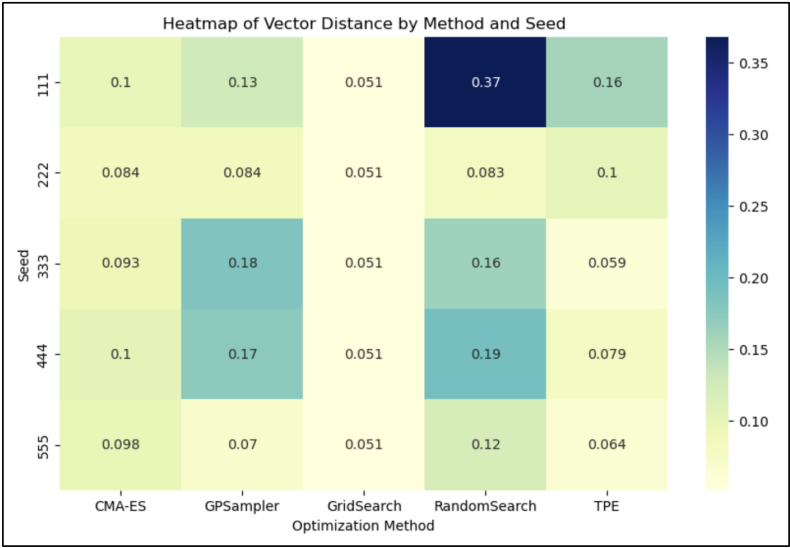


Figure 5.49. Heatmap of vector distance by method and seed in autocorrelated noise - mild case - stock management model.

Figure 5.50 highlights the trial distribution for each BBO method to achieve the best parameter set in the mild autocorrelated noise scenario.

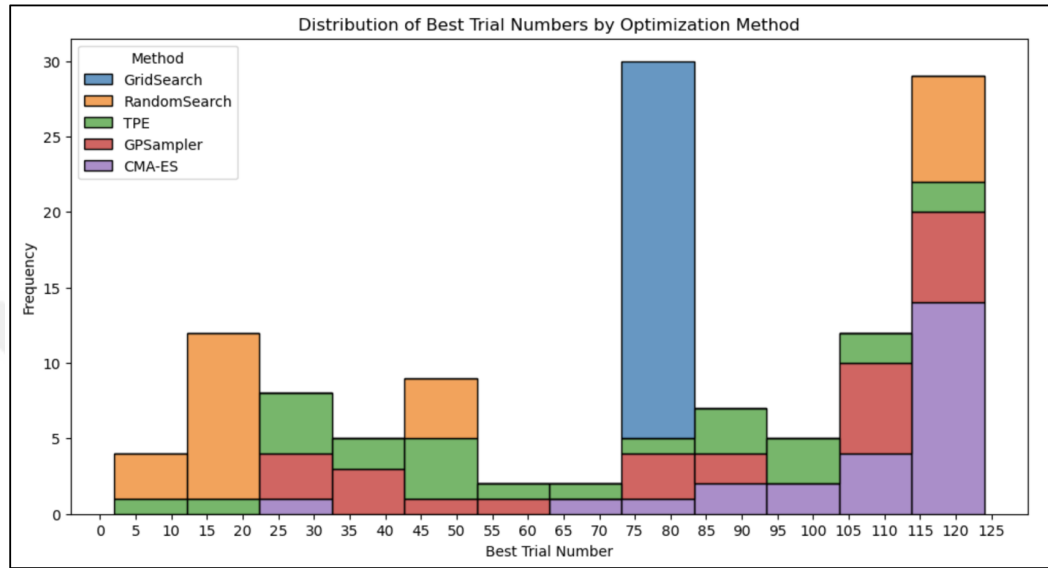


Figure 5.50. Distribution of best trial numbers by optimization method in autocorrelated noise - mild case - stock management model.

Table 5.20 captures the cumulative execution times for each BBO method, providing a comparative overview of computational performance in the mild autocorrelated noise case.

Table 5.20. Execution times for each algorithm in autocorrelated noise - mild case - stock management model.

	Execution Time
Grid Search	2 minutes 3 seconds
Random Search	1 minute 58 seconds
TPE	2 minutes 11 seconds
GP Sampler	11 minutes 6 seconds
CMA-ES	2 minutes 2 seconds

- Autocorrelated Noise - Strong Case:

Synthetic datasets were generated with strong autocorrelated noise. In this configuration, temporal dependencies were increased by introducing stronger autocorrelation (with higher *Stime*) and amplifying the noise intensity (with higher *k*). The resulting noise pattern created a more challenging environment for parameter calibration by generating substantial stochastic variability and introducing temporal correlations, which can potentially complicate the optimization process. Table 5.21 summarizes the parameter values used for noise generation.

Table 5.21. Smoothing parameters for autocorrelated noise - strong case - stock management model.

Parameter	Value
<i>Stime</i>	6
<i>k</i>	9

Figure 5.51 shows the inventory levels and the corresponding observed inventory data, reflecting strong autocorrelated noise, with a fixed SD seed value of 1234.

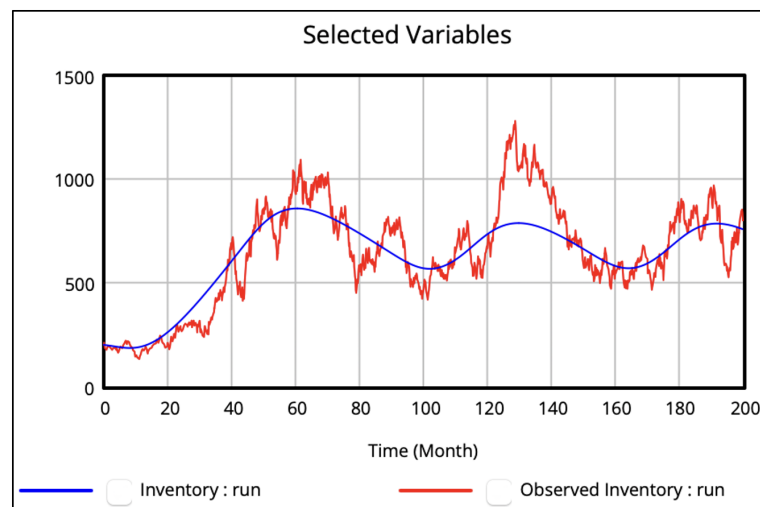
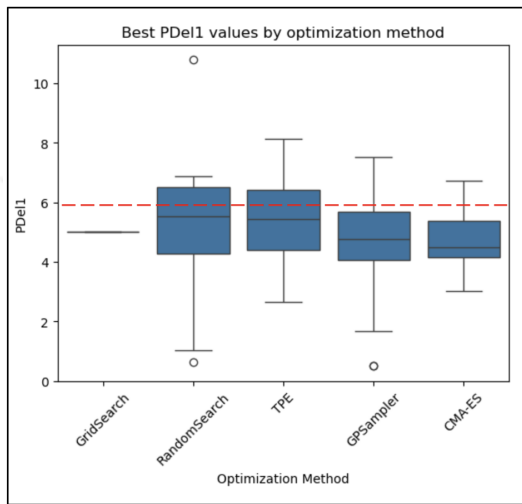
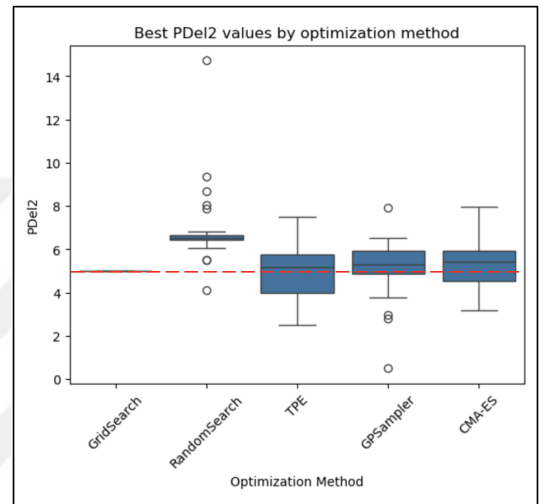


Figure 5.51. Inventory and observed inventory in autocorrelated noise - strong case.

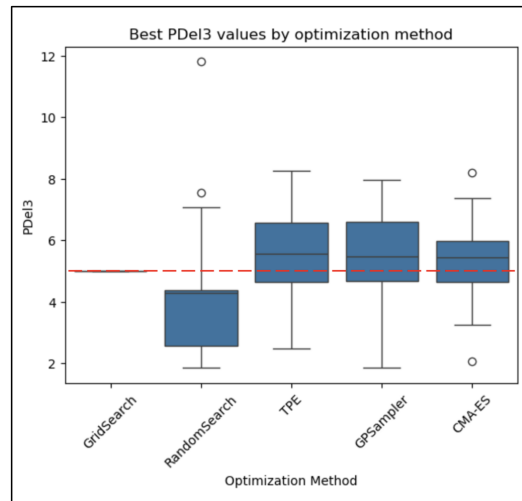
Figure 5.52 presents box plots summarizing the calibrated parameter estimates from 25 experimental runs by each algorithm, evaluated based on the minimum MSE. Random Search exhibited slightly less variability in this particular case compared to previous ones.



(a) PDel1



(b) PDel2



(c) PDel3

Figure 5.52. Best parameter sets of (a) PDel1, (b) PDel2, and (c) PDel3 in autocorrelated noise - strong case.

Figure 5.53 compares MSE and vector distance to the actual parameter sets for all BBO methods. When both MSE and vector distance are considered together, CMA-ES, TPE, and GP Sampler demonstrate comparatively better performance.

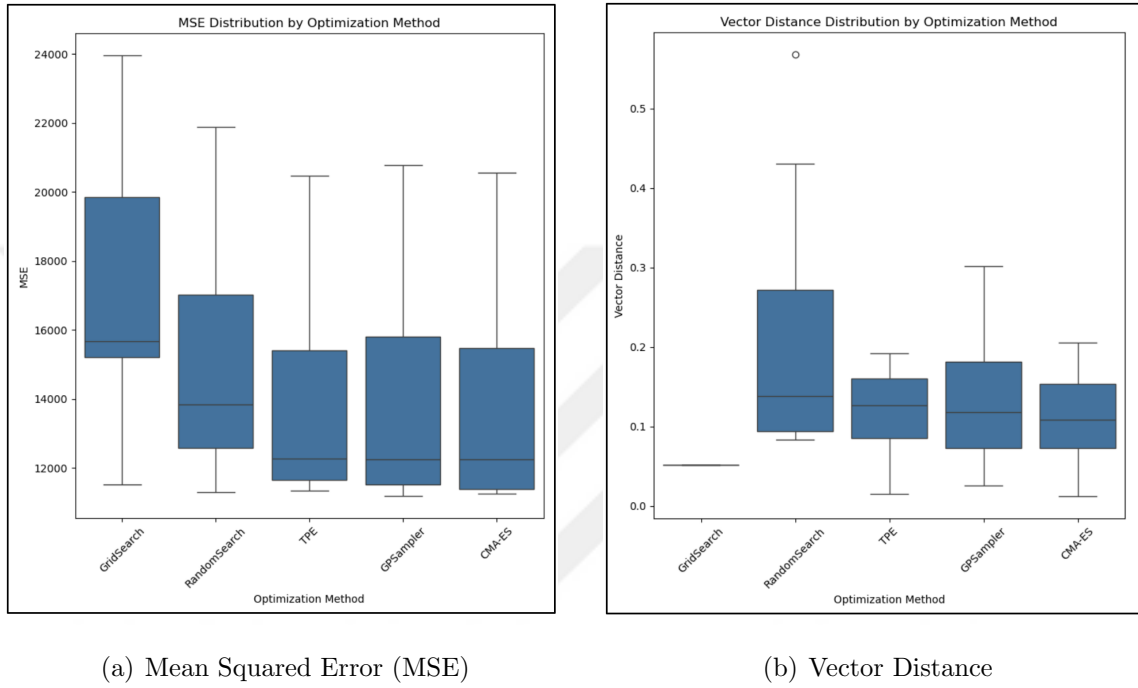


Figure 5.53. (a) Mean squared error (MSE), (b) vector distance results of the best parameter sets in autocorrelated noise - strong case - stock management model.

As the intensity of autocorrelated noise in the data increases, the dynamic behaviors associated with the parameter sets yielding the highest MSE among the best-performing results became more diverse (Figure 5.54). TPE provided the result that most closely aligns with the model's dynamic behavior. Unlike in the other cases, although CMA-ES and GP Sampler were able to capture the oscillatory pattern, their outputs showed noticeable deviations from the true system dynamics.

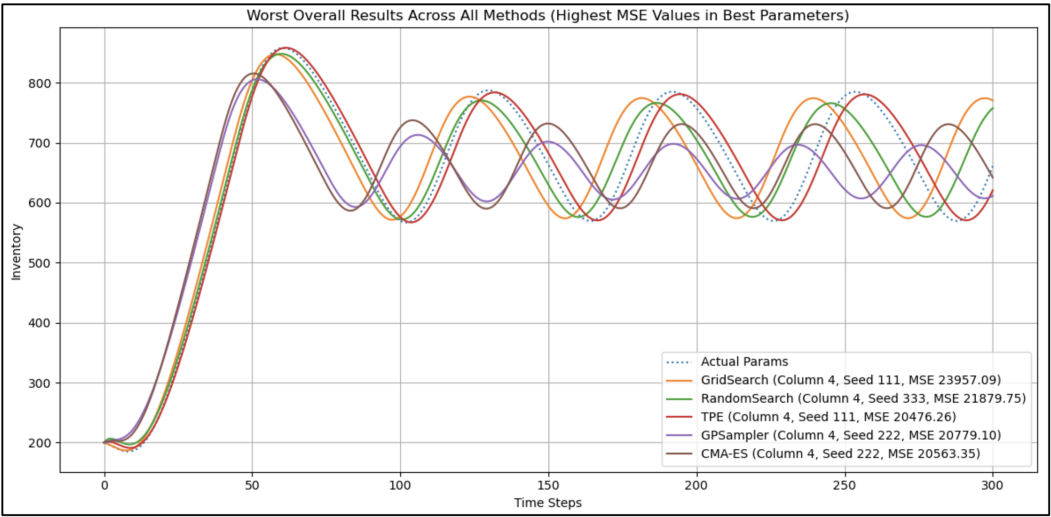


Figure 5.54. Dynamic behaviors from noise seeds that yield the highest MSE values in autocorrelated noise - strong case - stock management model.

Figure 5.55 provides a heatmap of vector distances between the calibrated and actual parameter sets, categorized by black-box seeds and optimization algorithms in the strong autocorrelated noise scenario. In this case, the algorithms generally yielded low vector distances.

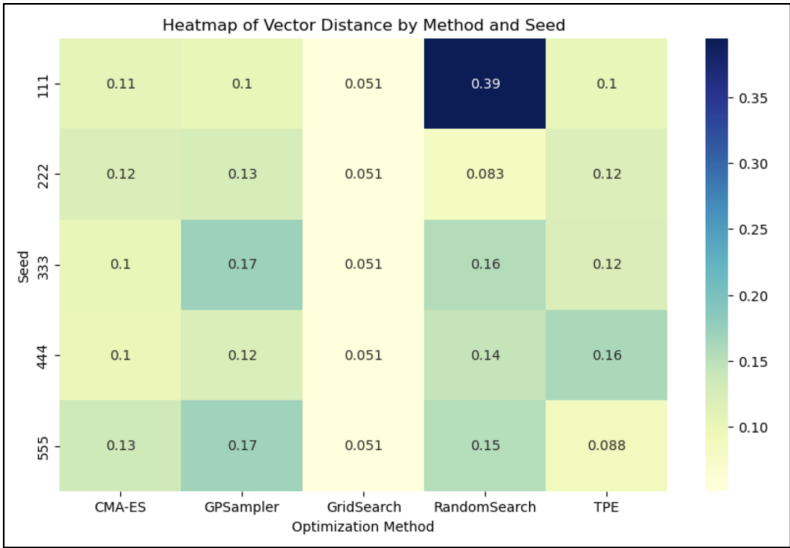


Figure 5.55. Heatmap of vector distance by method and seed in autocorrelated noise - strong case - stock management model.

Figure 5.56 illustrates the trial distribution for identifying the best solution across BBO methods under strong autocorrelated noise conditions. The results in this case are generally similar to those observed in the other scenarios.

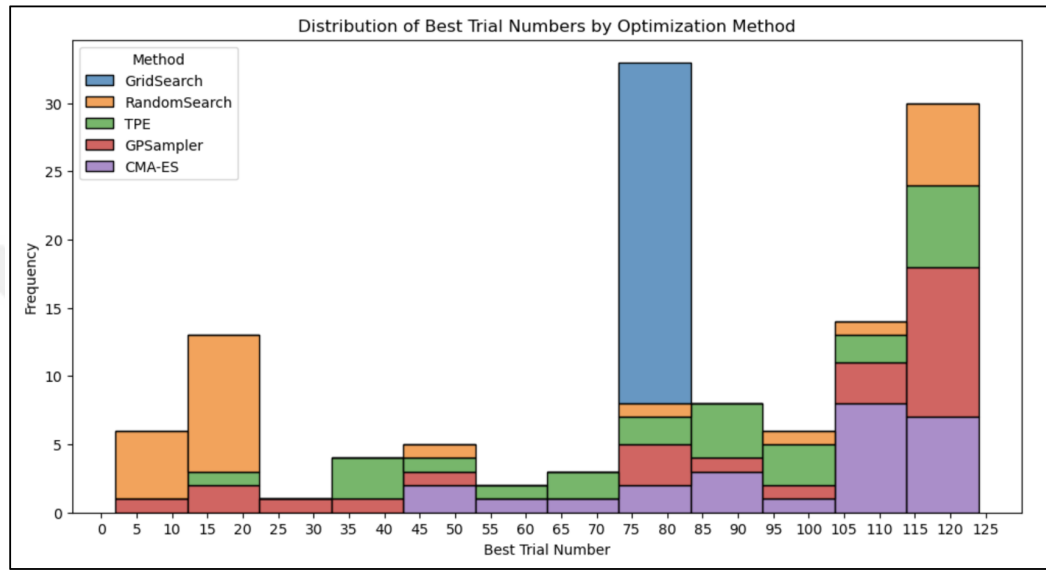


Figure 5.56. Distribution of best trial numbers by optimization method in autocorrelated noise - strong case - stock management model.

Table 5.22 details the execution durations for each BBO method, showcasing computational efficiency in the strong autocorrelated noise case.

Table 5.22. Execution times for each algorithm in autocorrelated noise - strong case - stock management model.

	Execution Time
Grid Search	2 minutes 3 seconds
Random Search	2 minutes
TPE	2 minutes 11 seconds
GP Sampler	15 minutes 26 seconds
CMA-ES	2 minutes 2 seconds

- Strong Pure Noise - Sparse Data Case:

In this case, synthetic datasets were generated using the same formulation as the strong pure noise scenario. However, the number of data points was significantly reduced to only 10 per dataset by modifying the simulation parameters. This case evaluates the performance of optimization algorithms under strong noise and limited data. Table 5.23 shows the parameter values used for noise generation and data sparsification.

Table 5.23. Parameters for strong pure noise - sparse data case - stock management model.

Parameter	Value
<i>Stime</i>	0.125
<i>k</i>	1.5
<i>SAVEPER</i>	15
<i>FINAL TIME</i>	150

Figure 5.57 illustrates the inventory and observed inventory data, generated under the strong pure noise sparse data scenario with a fixed SD seed value of 1234.

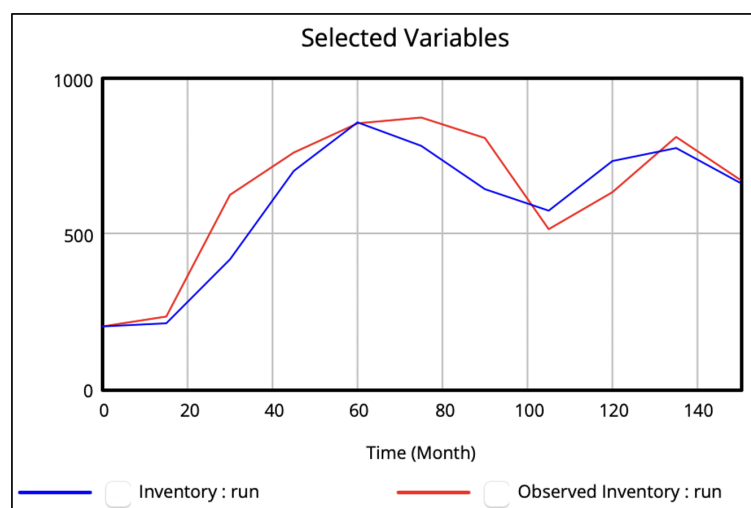


Figure 5.57. Inventory and observed inventory in strong pure noise - sparse data case.

Figure 5.58 presents the box plots summarizing the results of 25 experiments per algorithm, evaluating the best parameter sets based on the minimum MSE. Under limited data conditions, Grid Search no longer produced identical parameter sets and instead yielded more variable results. In fact, an overall increase in variability was observed across all algorithms.

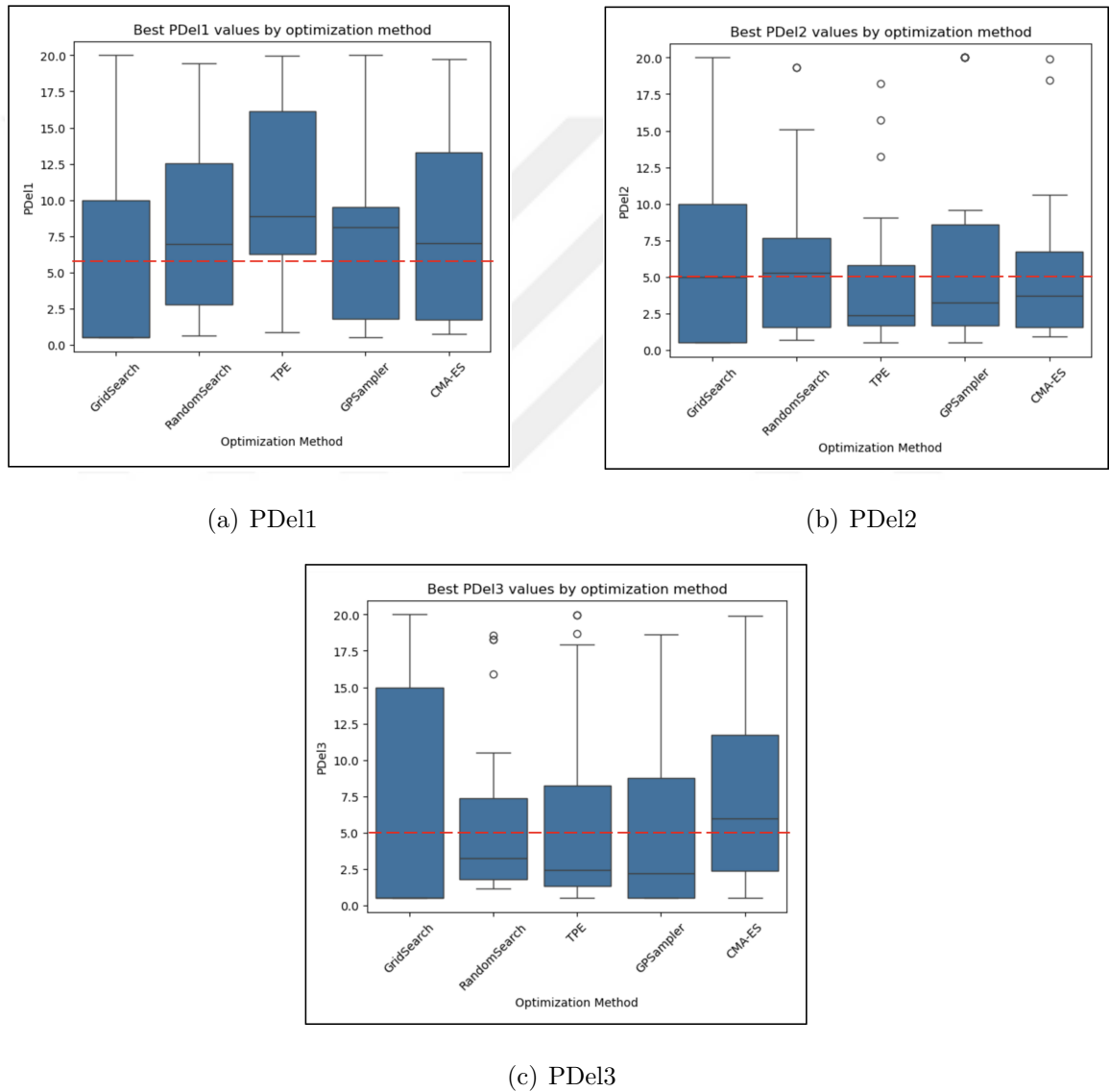


Figure 5.58. Best parameter sets of (a) PDe11, (b) PDe12, and (c) PDe13 in strong pure noise - sparse data case.

Figure 5.59 compares MSE and vector distance across BBO methods. Despite similar MSE values, Grid Search produced more distant parameter sets.

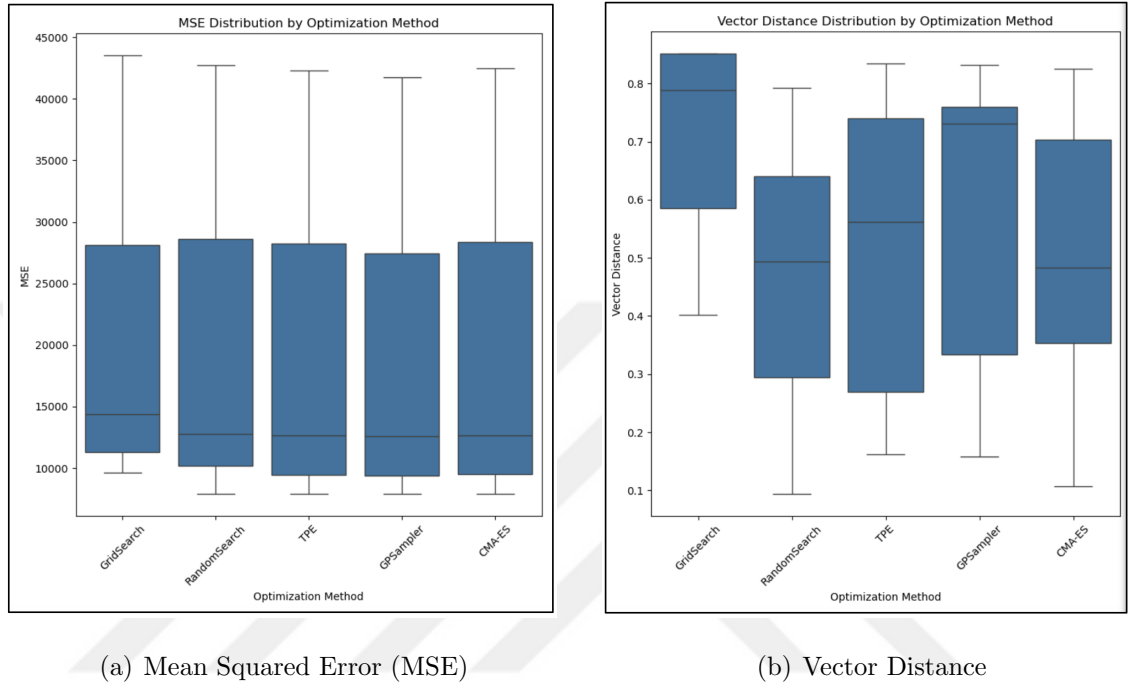


Figure 5.59. (a) Mean squared error (MSE), (b) vector distance results of the best parameter sets in strong pure noise - sparse data case - stock management model.

Figure 5.60 illustrates the dynamic behaviors based on the Strong Pure Noise – Sparse Data scenario, using a representative synthetic dataset (results from Column 4). In contrast to the relatively stable behavior of the SEIR model, the stock management model continued to produce more variable results.

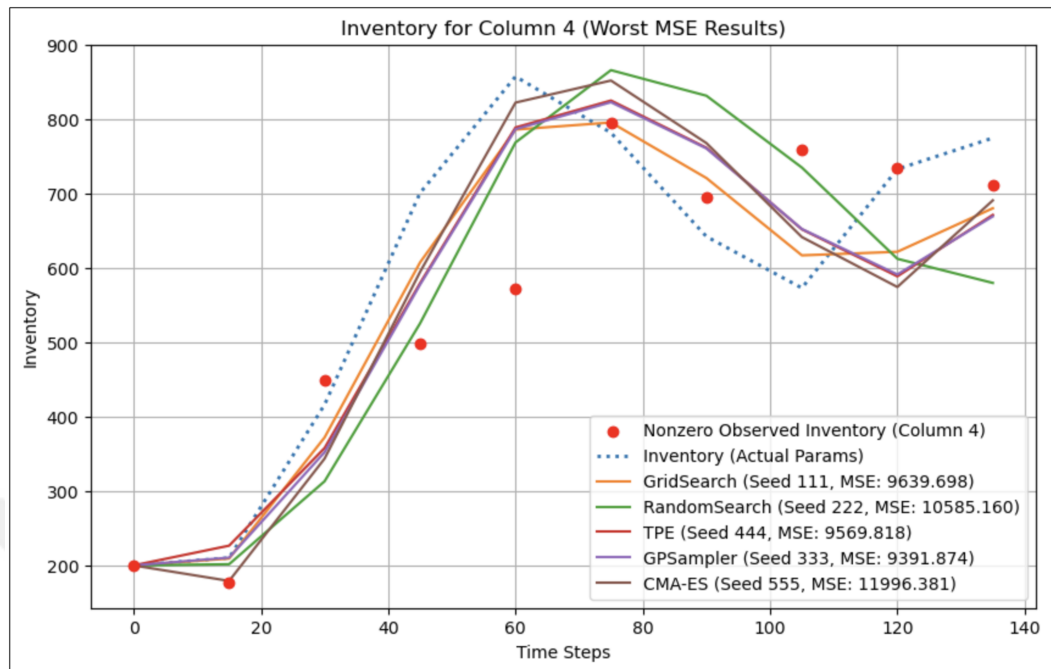


Figure 5.60. Dynamic behaviors from black-box seeds that yield the highest MSE values in strong pure noise - sparse data case - stock management model.

Figure 5.61 provides a heatmap of vector distances between the estimated and actual parameter sets, categorized by black-box seeds and optimization techniques under strong pure noise sparse data conditions.

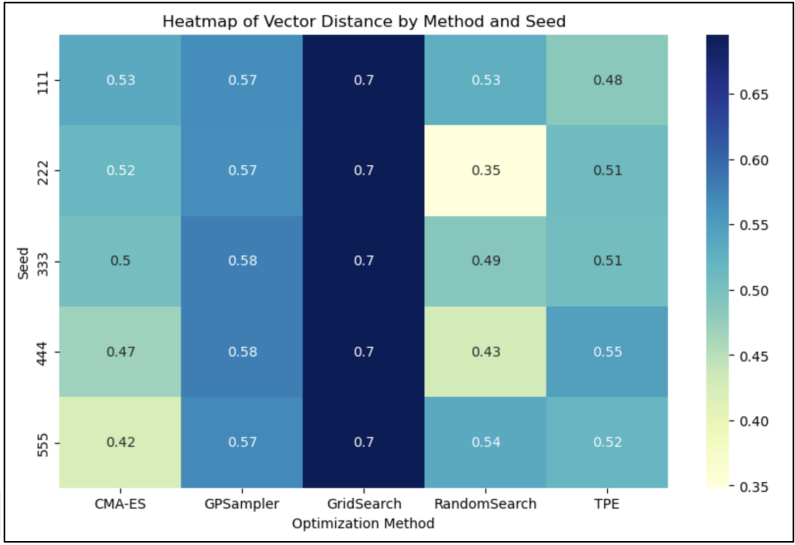


Figure 5.61. Heatmap of vector distance by method and seed in strong pure noise - sparse data case - stock management model.

Figure 5.62 displays the distribution of trials in which each BBO method identified the best solution. Unlike in the other cases, Grid Search exhibited variability, which also affected the diversity of trials yielding the best parameter sets.

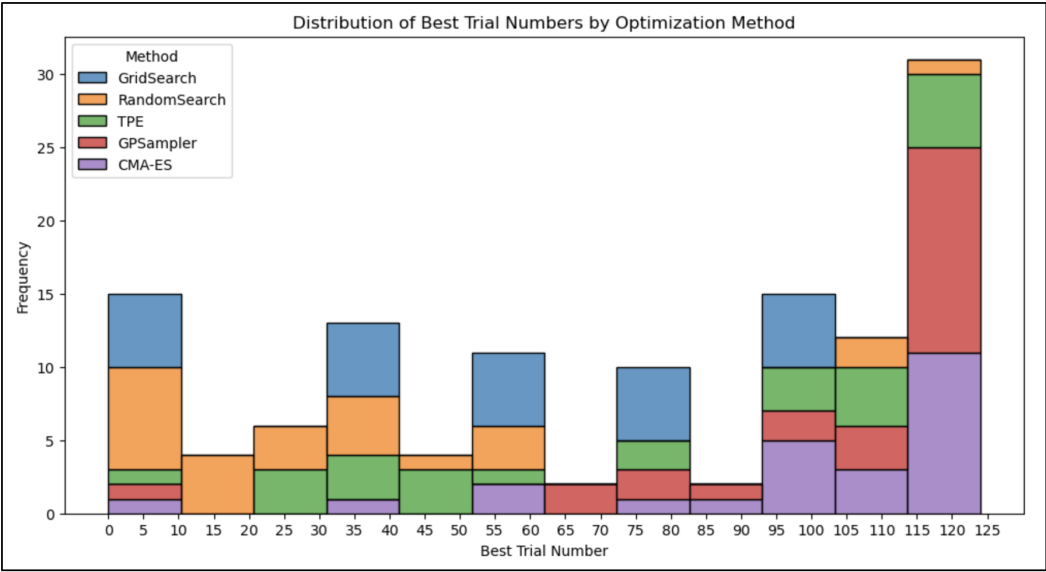


Figure 5.62. Distribution of best trial numbers by optimization method in strong pure noise - sparse data case - stock management model.

Table 5.24 reports the cumulative execution times for each BBO method, highlighting their computational performance under the constraints of strong noise and limited data availability.

Table 5.24. Execution times for each algorithm in strong pure noise - sparse data case - stock management model.

	Execution Time
Grid Search	54 seconds
Random Search	59 seconds
TPE	55 seconds
GP Sampler	10 minutes 45 seconds
CMA-ES	54 seconds

5.2.3. Extreme Condition Test

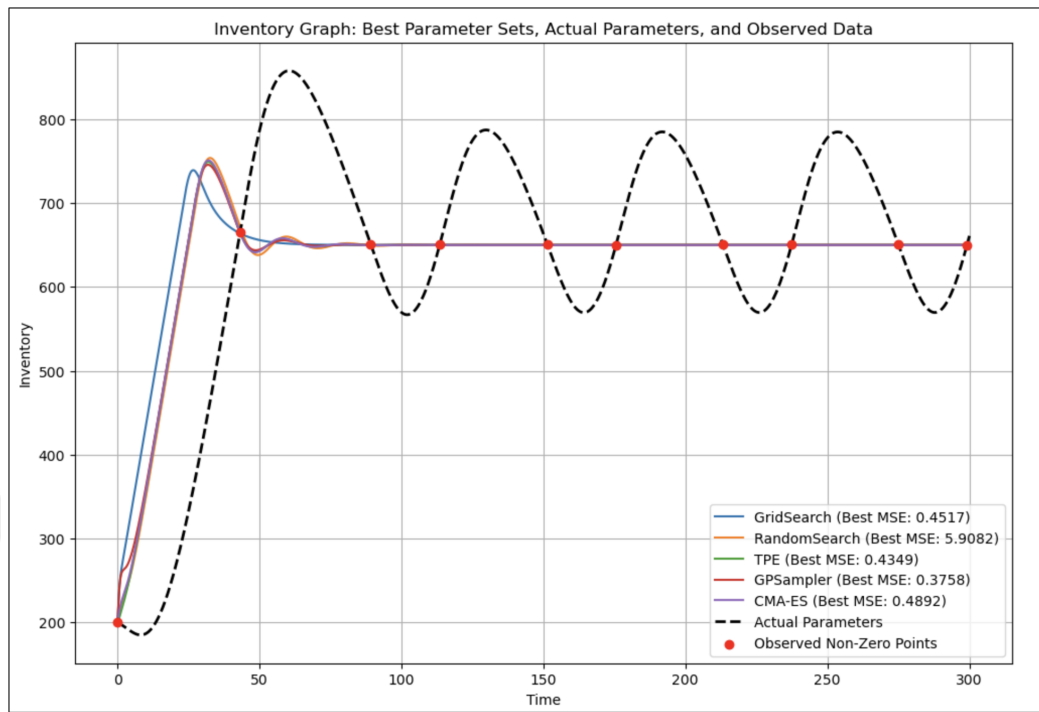
In this study, the SEIR model demonstrated robustness in maintaining its characteristic dynamic behavior across the defined parameter space. Despite deviations from the true parameter values, most parameter sets consistently produced the expected S-shaped curve, characterized by an initial overshoot followed by a gradual decline. This consistency suggests that the SEIR model's structure inherently supports stable dynamics. In contrast, the stock management model exhibited greater variability, even under noiseless conditions, with distinct parameter sets generating notably different dynamic behaviors. This variability underscores the challenges associated with calibrating models exhibiting oscillatory dynamics, where small changes in parameter values can significantly alter the system's behavior.

To assess the adaptability of the optimization algorithms, an extreme test case was introduced. In this scenario, 10 data points were selected to form a nearly linear pattern, requiring the algorithms to fit the model to a limited dataset. Despite the extreme nature of this test, some algorithms demonstrated their exploration capabilities by successfully detecting oscillatory behavior. This was observed even in synthetic

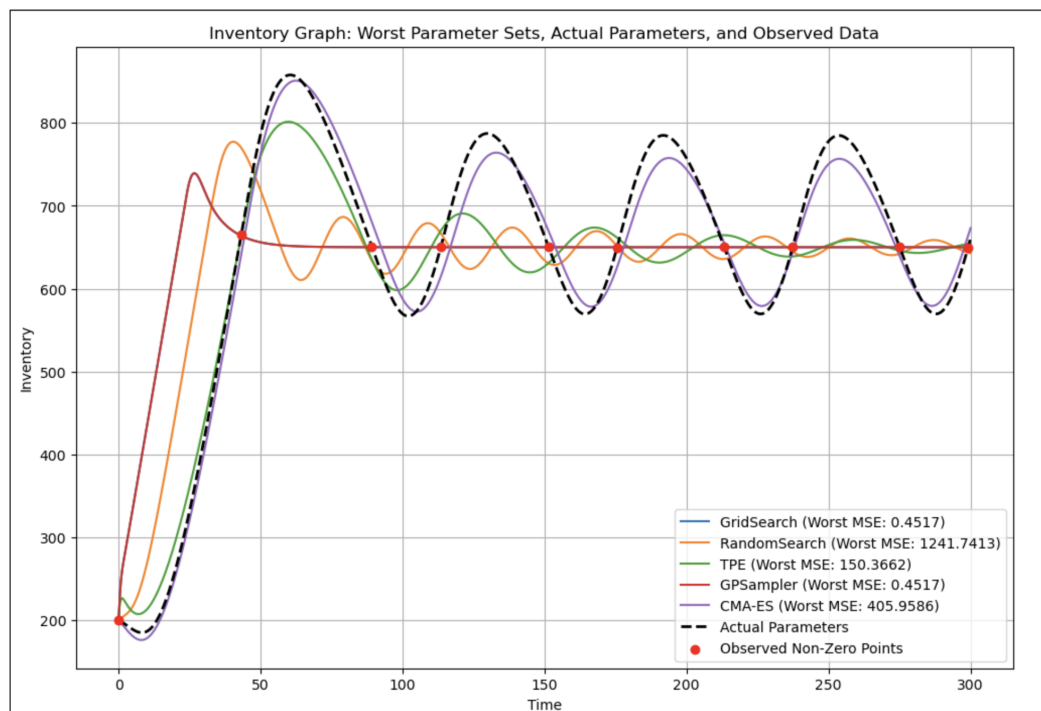
data that initially showed no oscillations, highlighting their ability to identify complex dynamics under challenging conditions.

Each algorithm was executed with 20 different black-box seed values, leading to variations in optimization outcomes across the runs. Since Grid Search is a deterministic method that systematically explores the parameter space, it produced consistent results regardless of the seed values. Although Optuna introduces randomness to the order in which combinations are tried, the number of trials in this study was set to ensure that Grid Search tested all combinations.

To illustrate the variability in optimization outcomes across the runs, the parameter sets yielding the lowest and highest MSE values among the 20 best results obtained by each algorithm are presented in Figure 5.63. This comparison highlights the performance range of each method when applied under different seed values. However, it is important to note that in this extreme scenario, even parameter sets associated with lowest MSE values failed to reproduce the model's true dynamic behavior. This highlights a potential risk in relying solely on statistical error metrics: a seemingly good fit may still yield misleading or invalid system dynamics. Although this case is synthetic and deliberately challenging, similar inconsistencies may also arise in more realistic settings, particularly when working with real-world data or structurally complex models.



(a) Best mean squared error (MSE)



(b) worst mean squared error (MSE)

Figure 5.63. (a) Best MSE, (b) worst MSE results of the best parameter sets in extreme condition test - stock management model.

On the other hand, to identify the diversity in dynamic behavior, a practical approach inspired by the principles of Behavior Analysis and Testing Software (BATS) [88] has been employed in this study to detect and classify the dynamic patterns observed in the results. The approach involves analyzing time series dynamics by extracting key features such as trend (slope), oscillation patterns (peak detection and amplitude decay), and stability over time (variance analysis). These extracted features are then used to categorize the time series into predefined behavioral patterns, including ‘oscillation’, ‘damped oscillation’, ‘exponential increase and decline’, and ‘stable’.

Based on the results obtained:

- Grid Search systematically explores a specified parameter space, rendering it unaffected by black-box seed values. Consequently, it identified a singular dynamic behavior characterized by an exponential increase and decline.
- Random Search identified a mix of oscillation (70%) and damped oscillation (30%).
- TPE achieved a more balanced distribution, detecting oscillation (45%) and damped oscillation (55%).
- GP Sampler classified the series mostly as damped oscillation (70%) but also identified some exponential increase and decline (30%).
- CMA-ES predominantly detected the oscillation (80%) with a smaller proportion of the damped oscillation (20%).

The results in this section provide a summary of the dynamic behaviors captured by each algorithm. These findings help reveal how effectively the methods can explore different time-series patterns. In addition, although metrics like MSE offer useful information about statistical fit, they are not always sufficient to assess whether a model reproduces the expected dynamic behavior. This limitation becomes more critical in complex or real-world applications. Therefore, a two-phase evaluation approach is recommended. First, simulation outputs should be checked against the expected behavioral patterns using automated classification tools such as BATS [88] or Indirect

Structure Testing Software (ISTS) [89]. Second, numerical metrics should be applied only to those simulations that pass this initial check. Although this study relies on synthetic cases, the results suggest that this approach could be valuable in practical modeling settings. In the following sections, a comprehensive evaluation of the experimental results will be provided, along with an analysis of their implications and broader context.

5.3. Comparative Analysis Across Models

The comparison between the SEIR and stock management models reveals different calibration challenges. These stem mainly from their distinct dynamic structures and the varying performance of the optimization algorithms. The SEIR model involves nonlinear epidemiological dynamics driven by parameters such as Incubation Time, Time to Recover, and Infection Fatality Rate. These parameters interact to produce characteristic S-shaped behavior, overshoot, and decline. In contrast, the stock management model features production delays (PDel1, PDel2, and PDel3) that induce oscillatory behavior. The presence of these delays makes the model sensitive to parameter variations, with the added noise in observed data further complicating the identification of optimal parameter values. Additionally, the limited availability of data introduces challenges, particularly when algorithms try to distinguish genuine dynamic patterns from noise-related distortions in the observations.

The primary distinction between the two models arises from the dynamic behaviors produced by the parameter sets identified during the optimization phase. In the SEIR model, even when the algorithms generated parameter sets that deviated from the true values, the resulting dynamics consistently exhibited an S-shaped pattern characterized by an initial overshoot followed by a gradual decline (Figures 5.5, 5.11, 5.17, 5.23). The model, by its nature, can exhibit different behaviors depending on the parameter sets. However, when the algorithms are executed to minimize MSE, the resulting dynamics closely align with those of the synthetic data, with no significant deviations in overall behavior. Even under the most challenging conditions, Strong Pure Noise - Sparse Data case, the algorithms effectively capture the overall system

dynamics (Figure 5.29). This consistency highlights the robustness of the SEIR model's dynamic behavior within the defined parameter space, as well as the effectiveness of the algorithms in identifying suitable parameter sets.

The experiments in this study reveal an important issue related to relying only on numerical fit during model calibration. In the SEIR model, under extreme conditions such as strong noise and limited data, some algorithms produced parameter sets that matched the synthetic data well. However, as seen in Figure 5.64, which is based on a specific synthetic dataset (Column 3), these parameter sets failed to reflect the actual dynamic behavior of the model. In such situations, a good numerical fit can be misleading, as the resulting dynamics may differ significantly from the system's expected behavior.

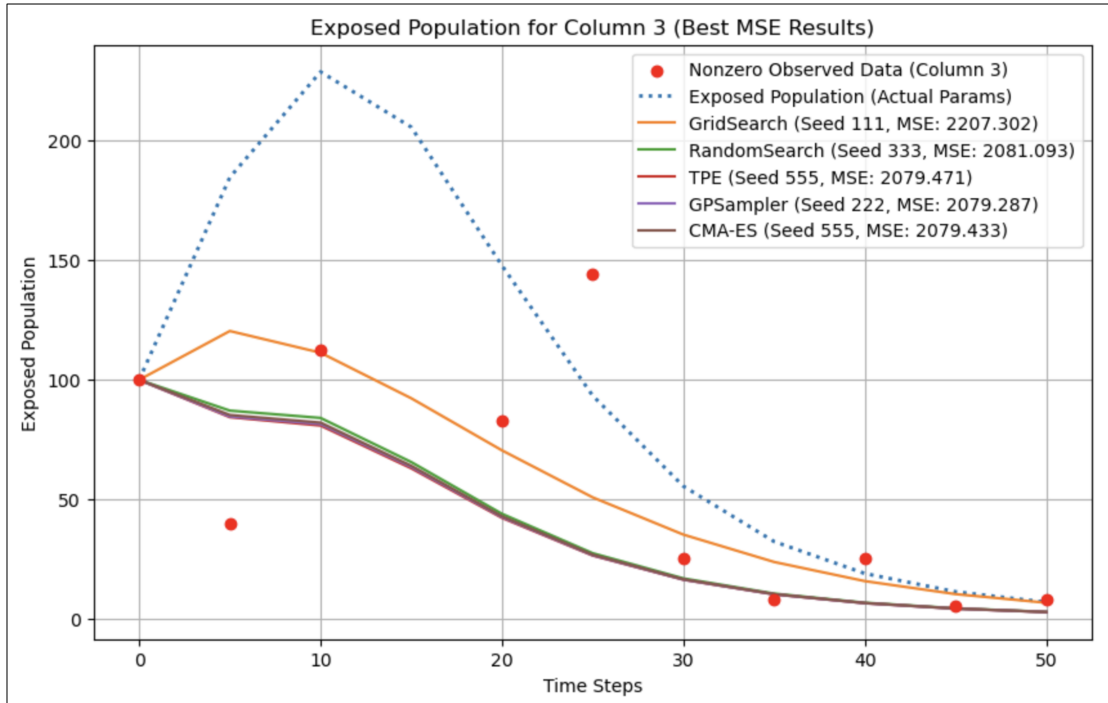


Figure 5.64. Best mean squared error results of an extreme condition in strong pure noise - sparse data case - SEIR model.

Conversely, the stock management model demonstrated greater variability (Figures 5.36, 5.42, 5.48, 5.54). The output of the model under the Strong Pure Noise

– Sparse Data scenario illustrates that, unlike the SEIR model, it does not exhibit a consistent dynamic behavior type (Figure 5.60). Instead, because of oscillations, the model can produce highly variable patterns in the outputs. Even without noise, the parameter sets resulted in diverse dynamic behaviors, with noticeable differences in oscillatory patterns. This study highlights the complexity of parameter calibration in models with oscillatory dynamics, where even small variations in parameters can lead to noticeably different system behaviors. In contrast to the SEIR model, the dynamics of the stock management model are more sensitive to parameter fluctuations, making it more challenging to obtain accurate calibration results.

On the other hand, comparing the results between the two models, particularly for Grid Search, is important. In the SEIR model, where the actual parameter values were included within the search space, Grid Search could theoretically evaluate all 125 parameter sets within 125 trials and identify the true parameters. However, the presence of varying noise in the observed data prevented this outcome. Under Pure Noise, Grid Search accurately identified two of the parameters, while the Infection Fatality Rate showed variability (Figure 5.3). When Strong Pure Noise was introduced, the Time to Recover parameter could no longer be precisely identified, and the overall variability increased (Figure 5.9). In the Autocorrelated Mild Noise case, the outputs displayed the highest vector distance (Figure 5.18). When all cases are considered collectively, despite the true parameter values being included in the search space, Grid Search exhibited the lowest overall performance among the algorithms. For the stock management model, Grid Search produced the same results across all experiments except for the Strong Pure Noise - Sparse Data case (Figures 5.34, 5.40, 5.46, 5.52, 5.58). Although it achieved results with very close vector distances, it consistently recorded the highest MSE values (Figures 5.35, 5.41, 5.47, 5.53). These findings indicate that Grid Search struggled to adapt under increasing noise conditions. In the Strong Pure Noise - Sparse Data case, Grid Search displayed variability and lost its position as the best-performing method in terms of vector distance, even producing the highest distances (Figure 5.55). However, it managed to achieve MSE values comparable to those of the other algorithms (Figure 5.59).

The presence of autocorrelated noise in the observed data of the SEIR model complicates the calibration process, as algorithms must account for temporal dependencies within the noisy observations of the exposed population. GP Sampler exhibited greater variability in accurately identifying the true parameter values, particularly for the parameters Infection Fatality Rate and Time to Recover (Figures 5.15, 5.21).

The stock management model involves interdependencies between inventory dynamics and production delays (PDel1, PDel2, and PDel3). The calibration of these delay parameters is interconnected, as inaccuracies in one can influence the others, potentially affecting the overall dynamic. This interdependence may complicate the optimization task compared to models with more independently acting parameters. Based on the box plots of the best parameter sets presented for all cases (Figures 5.34, 5.40, 5.46, 5.52), CMA-ES and TPE demonstrated better performance in handling these interdependencies. In contrast, Grid Search struggled to converge accurately for certain parameters, possibly because the true value of one parameter was not included in the predefined search space. However, GP Sampler, with its complex computational structure, struggled with the SEIR model and produced inconsistent outcomes (Figures 5.9, 5.15, 5.21). In contrast, for the stock management model, this complexity has contributed positively to the calibration process by facilitating the identification of interdependent parameters more effectively. This resulted in relatively more robust results compared to its performance in the SEIR model. Random Search, while performing well in certain cases, exhibited considerable variability across different SD and black-box seed values. The variability observed in Random Search's performance suggests a sensitivity to initial configurations. This may reduce the consistency of its results, particularly when compared to more deterministic methods like Grid Search.

The reduced data scenarios increased the difficulty of parameter calibration by substantially limiting the available observational data. In both models, all methods achieved similar MSE values, but Grid Search and GP Sampler yielded comparatively worse results in terms of vector distance (Figures 5.28, 5.59). In stock management model, Grid Search struggled to adapt to the reduced data, resulting in increased variability in parameter estimates (Figure 5.58).

5.4. Analysis and Discussion of Results

The comparative analysis of BBO algorithms highlights their varying performance across several evaluation criteria, including accuracy, computational efficiency, convergence speed, and robustness to noise. When both calibration accuracy (MSE) and vector distance are considered, CMA-ES performed best in most scenarios. It was able to produce parameter sets that aligned well with the synthetic data. TPE also performed well, producing results comparable to those of CMA-ES. Although Grid Search often produced parameter sets with low vector distances, these did not translate into accurate model outputs, as it frequently resulted in high MSE values. Random Search performed well in certain specific scenarios (Figures 5.27, 5.28), but its general results were inconsistent. GP Sampler produced more variable results with higher vector distances in the SEIR model. However, in the stock management model, its performance in terms of both MSE and vector distance was comparable to CMA-ES and TPE.

CMA-ES offered a good balance of speed and accuracy, completing the runs in reasonable time with reliable performance. Although TPE took longer to run than CMA-ES, it typically reached its best result in fewer trials than CMA-ES. Grid Search, due to its exhaustive approach, provided comprehensive exploration. However, its convergence varied depending on the randomness introduced by Optuna. The best results were sometimes achieved early in the trials, and other times later. While this method can be computationally expensive, especially in high-dimensional spaces, it showed acceptable execution times in the current experiments. Random Search was the fastest algorithm in terms of execution time. It often reached its best results in early trials. However, as the noise complexity increased in the SEIR model, from pure to strong autocorrelated noise and then to limited data, the trials in which the best results were obtained became more scattered and shifted to later stages. GP Sampler consistently had the longest execution times, often exceeding 10 minutes per case. It also tended to reach its best results toward the later stages of the trials in most cases.

CMA-ES generally produced stable results across different noise types. However, it showed slightly more variability in the strong autocorrelated noise scenario of

the SEIR model (Figure 5.21) and the strong pure noise sparse data scenario of the stock management model (Figure 5.58). TPE exhibited similar robustness, but showed slightly reduced stability under high noise conditions compared to CMA-ES (Figures 5.9, 5.52). Grid Search generally produced stable results, except in the strong pure noise sparse data case of the stock management model (Figure 5.58). In most scenarios, it returned a single consistent result for certain parameters. Although these parameter sets were close to the true values in terms of vector distance, they often failed to produce accurate model outputs, as reflected by high MSE values. This behavior is clearly illustrated in Figures 5.36, 5.42, 5.48, and 5.54, where the resulting dynamics deviate from the system behavior despite parameter proximity. As a result, the reliability of Grid Search in reproducing dynamic behavior may be limited under such conditions. Random Search was sensitive to both noise and initial seed values, which resulted in inconsistent performance across different scenarios. GP Sampler showed greater variability in its results for the SEIR model. In contrast, it showed more consistent performance across most noise types in the stock management model, with outcomes comparable to those of CMA-ES and TPE. However, its variability increased in the strong pure noise sparse data scenario of the stock management model (Figure 5.58).

Furthermore, the Extreme Condition Test conducted with the stock management model offered observations regarding the exploration capabilities of CMA-ES, TPE, and Random Search (Figure 5.63). While Random Search generally exhibited variable performance throughout the experiments, its stochastic nature proved beneficial in this case, successfully identifying oscillatory behavior despite the absence of inherent oscillations in the synthetic dataset. Similarly, CMA-ES and TPE demonstrated their exploration capabilities. These results show that some algorithms are capable of uncovering dynamic patterns that are not clearly present in the data. While this is useful in revealing hidden system behaviors, it also supports the earlier point that statistical fit alone is not enough. A separate behavioral check is still needed to ensure that the detected patterns are meaningful within the model's structure.

6. CONCLUSION

This study investigates the effectiveness of black-box optimization algorithms for parameter calibration in two generic system dynamics models, the SEIR and stock management models. Systematic experimentation under various noisy data conditions and data availability scenarios revealed findings, offering theoretical insights and practical implications for the application of black-box optimization in system dynamics models.

The experiments demonstrated that black-box optimization algorithms can be effectively applied to calibrate medium-scale system dynamics models, even under challenging conditions involving strong noise present in the observed data and limited data availability. Covariance Matrix Adaptation Evolution Strategy (CMA-ES) consistently demonstrated robust and reliable performance, achieving high accuracy with relatively low MSE values and vector distances in most scenarios. Tree-structured Parzen Estimator (TPE) achieved results comparable to Covariance Matrix Adaptation Evolution Strategy (CMA-ES), but with slightly higher variability in scenarios involving high noise levels. Grid Search, despite its exhaustive approach, showed limited accuracy, especially in the stock management model, where insufficient parameter space coverage affected its performance. Random Search demonstrated inconsistent outcomes. While it performed reasonably well in the SEIR model, its results in the stock management model were notably more variable. Gaussian Process (GP) Sampler was effective in calibrating interdependent parameters, such as the delays in the stock management model. However, its high computational requirements resulted in lower robustness and inconsistent results when applied to the SEIR model. Although Gaussian Process (GP) Sampler achieved good calibration accuracy, its high computational cost and lengthy processing time limited its practicality even in the medium-scale system dynamics models evaluated in this study.

The comparative analysis highlighted the strengths and limitations of each optimization method. It showed the importance of balancing accuracy, computational

efficiency, and adaptability across different modeling scenarios. The extreme condition tests provided additional insights, emphasizing the importance of thorough search space exploration for effective calibration. Relying exclusively on Mean Squared Error (MSE) minimization risks overlooking critical insights, including emergent dynamic behaviors that could be highly relevant for real-world applications. By extending the evaluation approach to include dynamic behavior classification, practitioners can uncover complex and unexpected system responses, offering a more comprehensive perspective on model calibration.

This study highlights the importance of supplementing traditional error-based calibration metrics with dynamic behavior analysis. Combining these approaches can improve the robustness of parameter calibration. It also helps to reveal edge cases and complex behaviors that might otherwise go unnoticed. The results from the extreme condition tests show that examining dynamic behavior can improve the calibration process and lead to a deeper understanding of how the system works. A two-phase approach that combines traditional error-based metrics with dynamic behavior classification may help improve calibration results. If the desired dynamic behavior is clearly defined, the calibration process can be directed to focus on parameter sets that reproduce this behavior while excluding those that lead to unrelated dynamics. This can result in cleaner and more consistent outputs by ensuring that the model captures not only numerical accuracy but also the expected system behavior.

Future research could focus on developing hybrid optimization strategies that combine the strengths of various algorithms to enhance scalability, robustness, and computational efficiency. Expanding these methods to include larger and more complex models, as well as real-world datasets, would be valuable for assessing their broader applicability and effectiveness. Additionally, advanced techniques such as using machine learning methods to identify and account for different types of noise in the observed data could enhance black-box optimization processes in system dynamics, making them more resilient to noise-related challenges. Future work could also explore an automated implementation of the two-phase calibration approach discussed in this study.

In conclusion, this study contributes to the field of system dynamics by systematically evaluating black-box optimization algorithms for parameter calibration under various scenarios involving noisy observed data and limited data availability. The findings, derived from both theoretical analysis and practical experimentation, offer insights into the strengths and limitations of different algorithms in handling noisy observations and limited data. The findings also suggest areas for further investigation to enhance calibration accuracy and efficiency in complex modeling environments.



REFERENCES

1. Bala, B. K., F. M. Arshad and K. M. Noh, *System Dynamics: Modelling and Simulation*, Springer, Singapore, 2017.
2. Forrester, J. W., *Industrial Dynamics*, MIT Press, Cambridge, Massachusetts, 1961.
3. Sterman, J. D., *Business Dynamics: Systems Thinking and Modeling for a Complex World*, Irwin/McGraw-Hill, Boston, 2000.
4. Oliva, R., “Model Calibration as a Testing Strategy for System Dynamics Models”, *European Journal of Operational Research*, Vol. 151, No. 3, pp. 552–568, 2003.
5. Schmiester, L., D. Weindl and J. Hasenauer, “Efficient Gradient-Based Parameter Estimation for Dynamic Models Using Qualitative Data”, *Bioinformatics*, Vol. 37, No. 23, pp. 4493–4500, 2021.
6. Yücel, G. and Y. Barlas, “Automated Parameter Specification in Dynamic Feedback Models Based on Behavior Pattern Features”, *System Dynamics Review*, Vol. 27, No. 2, pp. 195–215, 2011.
7. Dangerfield, B. and J. Duggan, “System Dynamics Models, Optimization of”, R. Meyers (Editor), *Encyclopedia of Complexity and Systems Science*, pp. 1–14, Springer, Berlin, Heidelberg, 2019.
8. Gardiner, C. W., *Stochastic Methods: A Handbook for the Natural and Social Sciences*, Springer, Berlin, 4th edn., 2009.
9. Wang, Z. and M. Ierapetritou, “A Novel Surrogate-Based Optimization Method for Black-Box Simulation with Heteroscedastic Noise”, *Industrial and Engineering Chemistry Research*, Vol. 56, No. 38, pp. 10720–10732, 2017.

10. Miller, J. J. and S. Mak, “Targeted Variance Reduction: Robust Bayesian Optimization of Black-Box Simulators With Noise Parameters”, arXiv preprint arXiv:2403.03816, 2024.
11. Bergstra, J. and Y. Bengio, “Random Search for Hyper-Parameter Optimization”, *Journal of Machine Learning Research*, Vol. 13, No. 1, pp. 281–305, 2012.
12. Shahriari, B., K. Swersky, Z. Wang, R. P. Adams and N. de Freitas, “Taking the Human Out of the Loop: A Review of Bayesian Optimization”, *Proceedings of the IEEE*, Vol. 104, No. 1, pp. 148–175, 2016.
13. Hansen, N. and A. Ostermeier, “Completely Derandomized Self-Adaptation in Evolution Strategies”, *Evolutionary Computation*, Vol. 9, No. 2, pp. 159–195, 2001.
14. Audet, C. and W. Hare, *Derivative-Free and Blackbox Optimization*, Springer, Cham, 2017.
15. Homer, J. B. and G. B. Hirsch, “System Dynamics Modeling for Public Health: Background and Opportunities”, *American Journal of Public Health*, Vol. 96, No. 3, pp. 452–458, 2006.
16. Currie, D. J., C. Smith and P. Jagals, “The Application of System Dynamics Modelling to Environmental Health Decision-Making and Policy-A Scoping Review”, *BMC Public Health*, Vol. 18, pp. 1–11, 2018.
17. Andersen, D. F., E. Rich and R. MacDonald, “Public Policy, System Dynamics Applications To”, R. A. Meyers (Editor), *Encyclopedia of Complexity and Systems Science*, pp. 7545–7559, Springer, New York, 2009.
18. Forrester, J. W., “System Dynamics and Learner-Centered-Learning in Kindergarten Through 12th Grade Education”, Text of Remarks Delivered December, 1992.

19. Richardson, G. P., “Feedback Thought in Social Science and Systems Theory”, *System Dynamics Review*, Vol. 7, No. 1, pp. 5–20, 1991.
20. Meadows, D. H., D. L. Meadows, J. Randers and W. W. Behrens, *The Limits to Growth*, Universe Books, New York, 1972.
21. Kermack, W. O. and A. G. McKendrick, “A Contribution to the Mathematical Theory of Epidemics”, *Proceedings of the Royal Society A: Mathematical, Physical and Engineering Sciences*, Vol. 115, No. 772, pp. 700–721, 1927.
22. Struben, J., “The Coronavirus Disease (COVID-19) Pandemic: Simulation-Based Assessment of Outbreak Responses and Postpeak Strategies”, *System Dynamics Review*, Vol. 36, No. 3, pp. 247–293, 2020.
23. Leontitsis, A., A. Senok, A. Alsheikh-Ali, Y. A. Nasser, T. Loney and A. Alshamsi, “Seahir: A Specialized Compartmental Model for COVID-19”, *International Journal of Environmental Research and Public Health*, Vol. 18, No. 5, p. 2667, 2021.
24. Dayan, F., N. Ahmed, M. Rafiq, A. Raza, I. Khan and E. M. T. Eldin, “A Reliable Numerical Investigation of an SEIR Model of Measles Disease Dynamics With Fuzzy Criteria”, *Scientific Reports*, Vol. 13, No. 1, p. 15840, 2023.
25. Lavine, J. S., M. Poss and B. T. Grenfell, “Directly Transmitted Viral Diseases: Modeling the Dynamics of Transmission”, *Trends in Microbiology*, Vol. 16, No. 4, pp. 165–172, 2008.
26. Inthamoussou, F. A., F. Valenciaga, S. Núñez and F. Garelli, “Extended SEIR Model for Health Policies Assessment Against the COVID-19 Pandemic: The Case of Argentina”, *Journal of Healthcare Informatics Research*, Vol. 6, No. 1, pp. 91–111, 2022.
27. Hou, Y. and H. Bidkhori, “Multi-Feature SEIR Model for Epidemic Analysis and Vaccine Prioritization”, *PLOS ONE*, Vol. 19, No. 3, p. e0298932, 2024.

28. He, S., Y. Peng and K. Sun, “SEIR Modeling of the COVID-19 and Its Dynamics”, *Nonlinear Dynamics*, Vol. 101, No. 3, pp. 1667–1680, 2020.
29. Sterman, J. D., “Modeling Managerial Behavior: Misperceptions of Feedback in a Dynamic Decision-Making Experiment”, *Management Science*, Vol. 35, No. 3, pp. 259–385, 1989.
30. Lee, H. L., V. Padmanabhan and S. Whang, “The Bullwhip Effect in Supply Chains”, *MIT Sloan Management Review*, Vol. 38, No. 3, pp. 93–102, 1997.
31. Li, X., A. Ghadami, J. M. Drake, P. Rohani and B. I. Epureanu, “Mathematical Model of the Feedback Between Global Supply Chain Disruption and COVID-19 Dynamics”, *Scientific Reports*, Vol. 11, No. 1, p. 15450, 2021.
32. Menner, M., K. Berntorp and S. Di Cairano, “Automated Controller Calibration by Kalman Filtering”, *IEEE Transactions on Control Systems Technology*, Vol. 31, No. 6, pp. 2350–2364, 2023.
33. Hazelbag, C. M., J. Dushoff, E. M. Dominic, Z. E. Mthombothi and W. Delva, “Calibration of Individual-Based Models to Epidemiological Data: A Systematic Review”, *PLoS Computational Biology*, Vol. 16, No. 5, p. e1007893, 2020.
34. Sun, N.-Z. and A. Sun, *Model Calibration and Parameter Estimation: For Environmental and Water Resource Systems*, Springer, Berlin, Heidelberg, 2015.
35. Adekitan, A. I. and O. Omoruyi, “Stock Keeping Accuracy: A Data Based Investigation of Storage Tank Calibration Challenges”, *Data in Brief*, Vol. 19, p. 2155, 2018.
36. Gupta, H. V., K. J. Beven and T. Wagener, “Model Calibration and Uncertainty Estimation”, M. G. Anderson (Editor), *Encyclopedia of Hydrological Sciences*, John Wiley & Sons, Chichester, 2006.

37. Hosseinichimeh, N., H. Rahmandad, M. S. Jalali and A. K. Wittenborn, “Estimating the Parameters of System Dynamics Models Using Indirect Inference”, *System Dynamics Review*, Vol. 32, No. 2, pp. 156–180, 2016.
38. Andrade, J. and J. Duggan, “A Bayesian Approach to Calibrate System Dynamics Models Using Hamiltonian Monte Carlo”, *System Dynamics Review*, Vol. 37, No. 4, pp. 283–309, 2021.
39. Yang, X.-S., *Nature-Inspired Metaheuristic Algorithms*, Luniver Press, Cambridge, 2010.
40. Parra, J. F., P. Jaramillo and S. Arango-Aramburo, “Metaheuristic Optimization Methods for Calibration of System Dynamics Models”, *Journal of Simulation*, Vol. 12, No. 2, pp. 190–209, 2018.
41. Talbi, E.-G., *Metaheuristics: From Design to Implementation*, John Wiley & Sons, Hoboken, New Jersey, 2009.
42. Storn, R. and K. Price, “Differential Evolution—A Simple and Efficient Heuristic for Global Optimization Over Continuous Spaces”, *Journal of Global Optimization*, Vol. 11, No. 4, pp. 341–359, 1997.
43. Back, T., *Evolutionary Algorithms in Theory and Practice: Evolution Strategies, Evolutionary Programming, Genetic Algorithms*, Oxford University Press, New York, 1996.
44. Golberg, D. E., *Genetic Algorithms in Search, Optimization, and Machine Learning*, Addison-Wesley, Reading, 1989.
45. Kirkpatrick, S., C. D. Gelatt Jr and M. P. Vecchi, “Optimization by Simulated Annealing”, *Science*, Vol. 220, No. 4598, pp. 671–680, 1983.
46. Kennedy, J. and R. Eberhart, “Particle Swarm Optimization”, *Proceedings of the*

- International Conference on Neural Networks*, Vol. 4, pp. 1942–1948, Institute of Electrical and Electronics Engineers, Perth, Australia, 1995.
47. Clerc, M. and J. Kennedy, “The Particle Swarm-Explosion, Stability, and Convergence in a Multidimensional Complex Space”, *IEEE Transactions on Evolutionary Computation*, Vol. 6, No. 1, pp. 58–73, 2002.
 48. Raue, A., C. Kreutz, T. Maiwald, J. Bachmann, M. Schilling, U. Klingmüller and J. Timmer, “Structural and Practical Identifiability Analysis of Partially Observed Dynamical Models by Exploiting the Profile Likelihood”, *Bioinformatics*, Vol. 25, No. 15, pp. 1923–1929, 2009.
 49. Hawkins, D. M., “The Problem of Overfitting”, *Journal of Chemical Information and Computer Sciences*, Vol. 44, No. 1, pp. 1–12, 2004.
 50. Bajaj, I., A. Arora and M. F. Hasan, “Black-Box Optimization: Methods and Applications”, P. M. Pardalos, V. Rasskazova and M. N. Vrahatis (Editors), *Black Box Optimization, Machine Learning, and No-Free Lunch Theorems*, pp. 35–65, Springer, Cham, 2021.
 51. Hansen, N., “Benchmarking a BI-Population CMA-ES on the BBOB Function Testbed”, *Proceedings of the 11th Annual Conference Companion on Genetic and Evolutionary Computation Conference: Late Breaking Papers*, pp. 2389–2396, Association for Computing Machinery, New York, USA, 2009.
 52. Bergstra, J., D. Yamins and D. Cox, “Making a Science of Model Search: Hyperparameter Optimization in Hundreds of Dimensions for Vision Architectures”, *Proceedings of the 30th International Conference on Machine Learning*, Vol. 28, pp. 115–123, Proceedings of Machine Learning Research, Atlanta, Georgia, USA, 2013.
 53. Bergstra, J., R. Bardenet, Y. Bengio and B. Kégl, “Algorithms for Hyper-Parameter Optimization”, *Advances in Neural Information Processing Systems*,

Vol. 24, pp. 2546–2554, Curran Associates, Inc., Granada, Spain, 2011.

54. Snoek, J., H. Larochelle and R. P. Adams, “Practical Bayesian Optimization of Machine Learning Algorithms”, *Advances in Neural Information Processing Systems*, Vol. 25, pp. 2951–2959, Curran Associates, Inc., Lake Tahoe, Nevada, USA, 2012.
55. Golovin, D., B. Solnik, S. Moitra, G. Kochanski, J. Karro and D. Sculley, “Google Vizier: A Service for Black-Box Optimization”, *Proceedings of the 23rd Association for Computing Machinery SIGKDD International Conference on Knowledge Discovery and Data Mining*, pp. 1487–1495, Association for Computing Machinery, Halifax, Nova Scotia, Canada, 2017.
56. Kandasamy, K., G. Dasarathy, J. Oliva, J. Schneider and B. Póczos, “Multi-Fidelity Gaussian Process Bandit Optimisation”, *Journal of Artificial Intelligence Research*, Vol. 66, pp. 151–196, 2019.
57. Park, S., J. Na, M. Kim and J. M. Lee, “Multi-Objective Bayesian Optimization of Chemical Reactor Design Using Computational Fluid Dynamics”, *Computers & Chemical Engineering*, Vol. 119, No. 1, pp. 25–37, 2018.
58. Liu, J., N. Ploskas and N. V. Sahinidis, “Tuning BARON Using Derivative-Free Optimization Algorithms”, *Journal of Global Optimization*, Vol. 74, No. 3, pp. 611–637, 2019.
59. Duggan, J., “An Exploratory Evaluation into the Effect of Data Availability and Indicator Coverage on Parameter Estimates Using Hamiltonian Monte Carlo”, *Proceedings of the 42nd International Conference of the System Dynamics Society*, pp. P1114–P1120, System Dynamics Society, Bergen, Norway, 2024.
60. Wakeland, W. and J. Homer, “Addressing Parameter Uncertainty in a Health Policy Simulation Model Using Monte Carlo Sensitivity Methods”, *Systems*, Vol. 10, No. 6, p. 225, 2022.

61. Rahmandad, H., A. Akhavan and M. S. Jalali, “Incorporating Deep Learning Into System Dynamics: Amortized Bayesian Inference for Scalable Likelihood-Free Parameter Estimation”, *System Dynamics Review*, Vol. 41, No. 1, p. e1798, 2025.
62. Angun, E. and J. P. C. Kleijnen, “Black-box Simulation-optimization with Quantile Constraints: An Inventory Case Study”, *Proceedings of the Winter Simulation Conference*, pp. 3506–3517, Institute of Electrical and Electronics Engineers, Orlando, Florida, USA, 2024.
63. Bansal, S., R. Calandra, T. Xiao, S. Levine and C. J. Tomlin, “Goal-Driven Dynamics Learning Via Bayesian Optimization”, *Proceedings of the Conference on Decision and Control*, pp. 1–7, Institute of Electrical and Electronics Engineers, Melbourne, Australia, 2017.
64. Pirrotta, A., “Non-Linear Systems Under Parametric White Noise Input: Digital Simulation and Response”, *International Journal of Non-Linear Mechanics*, Vol. 40, No. 8, pp. 1088–1101, 2005.
65. Stoyanov, M., M. Gunzburger and J. Burkardt, “Pink Noise, $1/f$ α Noise, and Their Effect on Solutions of Differential Equations”, *International Journal for Uncertainty Quantification*, Vol. 1, No. 3, 2011.
66. Voss, R. F. and J. Clarke, “ $1/f$ Noise in Music and Speech”, *Nature*, Vol. 258, No. 5533, pp. 317–318, 1975.
67. Valenti, D., G. Augello and B. Spagnolo, “Dynamics of a FitzHugh-Nagumo System Subjected to Autocorrelated Noise”, *The European Physical Journal B*, Vol. 65, No. 3, pp. 443–451, 2008.
68. Frazier, P. I., “Bayesian Optimization”, *INFORMS Tutorials in Operations Research*, pp. 255–278, Institute for Operations Research and the Management Sciences, Baltimore, 2018.

69. Hethcote, H. W., “The Mathematics of Infectious Diseases”, *SIAM Review*, Vol. 42, No. 4, pp. 599–653, 2000.
70. Brauer, F., “Mathematical Epidemiology: Past, Present, and Future”, *Infectious Disease Modelling*, Vol. 2, No. 2, pp. 113–127, 2017.
71. Ventana Systems, Inc., “SMOOTH Function in Vensim Documentation”, https://vensim.com/documentation/fn_smooth.html?q=smoothi, accessed on January 8, 2025.
72. Rasmussen, C. E. and C. K. I. Williams, *Gaussian Processes for Machine Learning*, MIT Press, Cambridge, 2006.
73. Hansen, N., “The CMA Evolution Strategy: A Comparing Review”, J. Lozano, P. Larrañaga, I. Inza and E. Bengoetxea (Editors), *Towards a New Evolutionary Computation*, pp. 75–102, Springer, Berlin, 2006.
74. Hyndman, R. J. and A. B. Koehler, “Another Look at Measures of Forecast Accuracy”, *International Journal of Forecasting*, Vol. 22, No. 4, pp. 679–688, 2006.
75. Halim, A. H., I. Ismail and S. Das, “Performance Assessment of the Metaheuristic Optimization Algorithms: An Exhaustive Review”, *Artificial Intelligence Review*, Vol. 54, No. 3, pp. 2323–2409, 2021.
76. He, J. and G. Lin, “Average Convergence Rate of Evolutionary Algorithms”, *IEEE Transactions on Evolutionary Computation*, Vol. 20, No. 2, pp. 316–321, 2015.
77. Kumar, K., “Data-Driven Modeling and Parameter Estimation of Nonlinear Systems”, *The European Physical Journal B*, Vol. 96, No. 7, p. 107, 2023.
78. Fan, J., Y. Liao, J. Zou, K. Liao and S. Fan, “A Neural Network with Strong Interpretability for Solving Optimization Problems”, *Applied Soft Computing*, Vol. 170, p. 112688, 2025.

79. Brauer, F. and C. Castillo-Chavez, *Mathematical Models in Population Biology and Epidemiology*, Vol. 40 of *Texts in Applied Mathematics*, Springer, New York, 2nd edn., 2012.
80. Mills, C. E., J. M. Robins and M. Lipsitch, “Transmissibility of 1918 Pandemic Influenza”, *Nature*, Vol. 432, No. 7019, pp. 904–906, 2004.
81. Rahman, A., M. A. Kuddus, R. H. Ip and M. Bewong, “A Review of COVID-19 Modelling Strategies in Three Countries to Develop a Research Framework for Regional Areas”, *Viruses*, Vol. 13, No. 11, p. 2185, 2021.
82. Rainisch, G., E. A. Undurraga and G. Chowell, “A Dynamic Modeling Tool for Estimating Healthcare Demand from the COVID19 Epidemic and Evaluating Population-Wide Interventions”, *International Journal of Infectious Diseases*, Vol. 96, No. 1, pp. 376–383, 2020.
83. Guan, W.-j., Z.-y. Ni, Y. Hu, W.-h. Liang, C.-q. Ou, J.-x. He, L. Liu, H. Shan, C.-l. Lei, D. S. Hui, B. Du, L.-j. Li, G. Zeng, K.-Y. Yuen, R.-c. Chen, C.-l. Tang, T. Wang, P.-y. Chen, J. Xiang, S.-y. Li, J.-l. Wang, Z.-j. Liang, Y.-x. Peng, L. Wei, Y. Liu, Y.-h. Hu, P. Peng, J.-m. Wang, J.-y. Liu, Z. Chen, G. Li, Z.-j. Zheng, S.-q. Qiu, J. Luo, C.-j. Ye, S.-y. Zhu, N.-s. Zhong and China Medical Treatment Expert Group for Covid-19, “Clinical Characteristics of Coronavirus Disease 2019 in China”, *The New England Journal of Medicine*, Vol. 382, No. 18, pp. 1708–1720, 2020.
84. Li, Q., X. Guan, P. Wu, X. Wang, L. Zhou, Y. Tong, R. Ren, K. S. Leung, E. H. Lau, J. Y. Wong, X. Xing, N. Xiang, Y. Wu, C. Li, Q. Chen, D. Li, T. Liu, J. Zhao, M. Liu, W. Tu, C. Chen, L. Jin, R. Yang, Q. Wang, S. Zhou, R. Wang, H. Liu, Y. Luo, Y. Liu, G. Shao, H. Li, Z. Tao, Y. Yang, Z. Deng, B. Liu, Z. Ma, Y. Zhang, G. Shi, T. T. Lam, J. T. Wu, G. F. Gao, B. J. Cowling, B. Yang, G. M. Leung and Z. Feng, “Early Transmission Dynamics in Wuhan, China, of Novel Coronavirus-Infected Pneumonia”, *The New England Journal of Medicine*, Vol. 382, No. 13, pp. 1199–1207, 2020.

85. Lauer, S. A., K. H. Grantz, Q. Bi, F. K. Jones, Q. Zheng, H. R. Meredith, A. S. Azman, N. G. Reich and J. Lessler, “The Incubation Period of Coronavirus Disease 2019 (COVID-19) from Publicly Reported Confirmed Cases: Estimation and Application”, *Annals of Internal Medicine*, Vol. 172, No. 9, pp. 577–582, 2020.
86. Optuna Project Contributors, “Optuna: A Hyperparameter Optimization Framework”, <https://optuna.org/>, version 4.0.0, accessed on October 15, 2024.
87. Towill, D. R., “Industrial Dynamics Modelling of Supply Chains”, *International Journal of Physical Distribution and Logistics Management*, Vol. 26, No. 2, pp. 23–42, 1996.
88. Sücüllü, C. and G. Yücel, “Behavior Analysis and Testing Software (BATS)”, *Proceedings of the 32nd International Conference of the System Dynamics Society*, pp. 20–24, System Dynamics Society, Delft, Netherlands, 2014.
89. Edali, M., M. Baydoğan and G. Yücel, “Classification of Generic System Dynamics Model Outputs via Supervised Time Series Pattern Discovery”, *Turkish Journal of Electrical Engineering and Computer Sciences*, Vol. 27, No. 2, pp. 832–846, 2019.

APPENDIX A: SEIR MODEL EQUATIONS

"C-fraction2"=0.02

Units: 1/Day

"C-fraction1"=0.01

Units: 1/Day

Death Rate= (Infected Population*Infection Fatality Rate)/Time to Decease

Units: Person/Day

Deceased Population= INTEG (Death Rate, 0)

Units: Person

E to S Transmission="I-fraction2"*"S and E Contacts"

Units: Person/Day

Exposed Population= INTEG (Transmission Rate-Onset Rate, 100)

Units: Person

FINAL TIME =100

Units: Day

The final time for the simulation.

I to S Transmission="I-fraction1"*"S and I Contacts"

Units: Person/Day

"I-fraction1"=0.015

Units: 1/Person

"I-fraction2"=0.01

Units: 1/Person

Incubation Time=5

Units: Day

Infected Population= INTEG (Onset Rate-Death Rate-Recovery Rate, 120)

Units: Person

Infection Fatality Rate=0.05

Units: Dmnl

INITIAL TIME=0

Units: Day

The initial time for the simulation.

Onset Rate=Exposed Population/Incubation Time

Units: Person/Day

"Possible Contacts (S and E)"= Exposed Population*Susceptible Population

Units: Person*Person

"Possible Contacts (S and I)"= Infected Population*Susceptible Population

Units: Person*Person

Recovered Population= INTEG (Recovery Rate, 0)

Units: Person

Recovery Rate= (Infected Population*(1-Infection Fatality Rate))/Time to Recover

Units: Person/Day

"S and E Contacts"="C-fracton2"*"Possible Contacts (S and E)"

Units: Person*Person/Day

"S and I Contacts"="C-fraction1"*"Possible Contacts (S and I)"

Units: Person*Person/Day

SAVEPER=TIME STEP

Units: Day [0,?]

The frequency with which output is stored.

Susceptible Population= INTEG (-Transmission Rate, 990)

Units: Person

TIME STEP=0.125

Units: Day [0,?]

The time step for the simulation.

Time to Decease=7

Units: Day

Time to Recover=14

Units: Day

Transmission Rate= (I to S Transmission)+(E to S Transmission)

Units: Person/Day

APPENDIX B: STOCK MANAGEMENT MODEL EQUATIONS

$$\text{Demand}=10$$

Units: Item/Month

$$\text{DesProduction}=(\text{InvGoal}-\text{Inventory})*\text{IAT}+\text{Expected Sales}$$

Units: Item/Month

$$\text{EstDel}=7$$

Units: Month

$$\text{Expected Sales}= \text{INTEG} (\text{Flow2}, 200)$$

Units: Item/Month

$$\text{FINAL TIME}=300$$

Units: Month

The final time for the simulation.

$$\text{Finishing1}=\text{WIP1}/\text{PDel1}$$

Units: Item/Month

$$\text{Finishing2}=\text{WIP2}/\text{PDel2}$$

Units: Item/Month

$$\text{Finishing3}=\text{WIP3}/\text{PDel3}$$

Units: Item/Month

$$\text{Flow1}= (\text{Sales}-\text{Stock1})/\text{EstDel}$$

Units: Item/(Month*Month)

Flow2=(Stock1-Expected Sales)/EstDel

Units: Item/(Month*Month)

IAT=0.3

Units: 1/Month

INITIAL TIME=0

Units: Month

The initial time for the simulation.

Inventory= INTEG (Finishing3-Sales, 200)

Units: Item

InvGoal=650

Units: Item

PCapac=30

Units: Item/Month

PDel1=6

Units: Month

PDel2=5

Units: Month

PDel3=5

Units: Month

Production=MAX(MIN(PCapac, DesProduction), 0)

Units: Item/Month

Sales=MIN(Inventory*SalesT, Demand)

Units: Item/Month

SalesT=0.7

Units: 1/Month

SAVEPER=TIME STEP

Units: Month [0,?]

The frequency with which output is stored.

Stock1= INTEG (Flow1, 200)

Units: Item/Month

TIME STEP=0.125

Units: Month [0,?]

The time step for the simulation.

WIP1= INTEG (Production-Finishing1, 0)

Units: Item

WIP2= INTEG (Finishing1-Finishing2, 40)

Units: Item

WIP3= INTEG (Finishing2-Finishing3, 40)

Units: Item