Automatic Tuning for Factor Graph Based Estimation using Bayesian Optimisation

William Terry [1]

MSc Robotics and AI

Supervisor: Simon Julier & Nisar Ahmed

15 09 2025

Abstract

This dissertation addresses the persistent challenge of manual parameter tuning in factor graph-based estimation systems. While the application of automated tuning methodologies has been explored for Kalman filters, this work proposes to extend this approach to the domain of factor graphs. The performance and statistical consistency of these estimators can be highly dependent on the accurate selection of noise model parameters, a task that is often time-consuming, subjective, and prone to error. This work proposes an automated tuning methodology utilising Bayesian Optimisation (BO), a sample-efficient global optimisation technique. By employing a Gaussian Process to model the relationship between the system's tuning parameters and a chosen performance metric, the BO framework intelligently selects the next set of parameters to evaluate, thereby reducing the number of computationally expensive simulations. The proposed approach is demonstrated to efficiently identify optimal and statistically consistent parameter configurations, outperforming traditional manual tuning methods. The research validates the efficacy of Bayesian Optimisation as a powerful tool for solving complex, non-linear tuning problems, paving the way for more robust and reliable estimation in diverse applications such as robotics, Simultaneous Localisation and Mapping (SLAM), and sensor fusion.

Contents

List of Figures

List of Tables

Abbreviations

BO Bayesian Optimization. 14–16, 18, 28, 32, 39–42, 44–46, 48, 51–53

CNEES Consistent Normalized Estimation Error Squared. 12, 14, 17, 18

CNIS Consistent Normalized Innovation Squared. 12, 14, 17, 18, 22, 26, 28–30, 33–35, 37–42, 44, 45, 48, 51, 53–55, 57

CT Constant Turn rate. 19, 25, 27

CV Constant Velocity. 19, 21, 24, 26

DoF Degrees of Freedom. 10, 12, 45

EI Expected Improvement. 16, 37, 39–42

g2o General Graph Optimisation. 3

GP Gaussian Process. 15–18

GPS Global Positioning System. 3

GTSAM Georgia Tech Smoothing And Mapping. 3

JNEES Joint Normalized Estimation Error Squared. 11

JNIS Joint Normalized Innovation Squared. 11

LCB Lower Confidence Bound. 16, 17, 37, 40, 42

MAP Maximum A Posteriori. 7

MSE Mean Squared Error. 5, 54–57

NEES Normalized Estimation Error Squared. 5, 10, 11, 13, 18, 55

NIS Normalized Innovation Squared. 5, 10, 11, 13, 18, 32, 33, 45, 51, 52, 55

Chapter 1

Introduction

1.1 Background and Motivation

Tracking is a fundamental capability in robotics, forming the backbone of autonomous navigation and interaction with the environment. From household robots that clean and map rooms to exploration robots navigating complex terrains such as caves or disaster zones, accurate tracking ensures robots can localise themselves, plan motion, and interact safely with their surroundings [1, 2].

Despite its importance, reliable tracking remains a challenging problem due to sensor noise, environmental uncertainties, and nonlinear system dynamics. Traditional approaches, such as Kalman Filters [1] or Particle Filters [2], address some of these challenges but have limitations. The Kalman Filter is a linear minimum mean squared error estimator, and while it is Bayes-optimal in the linear Gaussian case, its applicability can be restricted in nonlinear or non-Gaussian settings. Particle Filters, on the other hand, can cope with nonlinearities but often require a large number of particles to maintain accuracy, which can be computationally expensive. Both approaches may still face difficulties in long-term tracking or in high-dimensional, complex environments.

Factor graphs have emerged as a modern and flexible framework for robotic tracking [3, 4, 5]. By representing the system as a graph of variables and factors, they allow a more natural representation of complex conditional dependencies between system variables. This structure enables optimisation methods to better access and combine information from multiple sources for state estimation. As a result, factor graph-based tracking can handle nonlinearities more effectively than traditional filters, integrate multiple sensor modalities, and scale to large problems [6].

While factor graphs provide a powerful framework, their performance can be influenced by choices such as noise models, motion covariances, and solver parameters [7]. Selecting suitable parameters can improve tracking accuracy and consistency, though factor graphs remain capable of producing reasonable results even with default or approximate settings. This observation motivates research into methods for systematically exploring and optimising these parameters in a principled way.

1.2 Problem Statement

In robotic tracking, the accuracy and reliability of a system depend not only on the underlying model but also on how parameters, such as noise covariances and solver settings, are selected [7]. Manual tuning is labor-intensive, requires domain expertise, and may not yield optimal results.

This is particularly relevant for factor graph-based tracking, where high-dimensional, nonlinear parameter spaces make systematic tuning challenging.

Existing automated tuning approaches, such as Bayesian optimisation applied to Kalman Filters [8], provide promising directions. However, they are limited by assumptions of linearity and Gaussian noise, which restricts their applicability in complex, nonlinear robotic tracking problems. Factor graph-based systems offer a more expressive framework, but automated tuning methods for these systems are still an active research area.

The problem this research addresses is therefore: how can we systematically and automatically tune factor graph-based tracking systems in robotics to improve accuracy, consistency, and robustness across diverse environments and sensor setups?

1.3 Manual Tuning Methods and Historical Context

The foundation of parameter tuning in state estimation stems from early work on Kalman filters and their extensions. Simon's book on optimal state estimation [9] and Julier's contributions to unscented Kalman filters [10] established the theoretical basis for consistency evaluation, while Bar-Shalom's seminal work formalized statistical consistency conditions that remain fundamental to modern estimation theory [11].

Early approaches relied heavily on manual parameter adjustment, where engineers would iteratively modify process and measurement noise parameters based on qualitative assessment of filter performance. Grid search methods emerged as a more systematic alternative, involving exhaustive exploration of parameter spaces to find optimal configurations [12]. However, these approaches suffer from computational inefficiency and poor scalability to high-dimensional parameter spaces, motivating the development of more sophisticated automated tuning strategies.

1.4 Project Objectives

The primary objectives of this project are as follows:

1. **Design and implement a factor graph-based tracking system:** Develop a tracking system capable of estimating the state of a robot using sensor inputs such as Global Positioning System (GPS) and odometry. The system will be structured using factor graphs, leveraging open-source libraries such as Georgia Tech Smoothing And Mapping (GTSAM) or General Graph Optimisation (g2o) for efficient optimisation.

2. **Investigate the use of Bayesian optimisation for tuning factor graphs:** Examine whether an approach commonly used for tuning Kalman filters can be effectively applied to the problem of tuning factor graph systems. Evaluate how well the automatically selected noise and solver parameters support accurate and consistent tracking.

3. **Explore refinement strategies for parameter tuning:** Investigate additional methods to improve parameter selection and assess potential gains in consistency compared to the baseline automatically tuned system.

4. **Evaluate system performance across different scenarios:** Test the tracking system's performance under various conditions including different trajectory lengths, initialization strategies, and model assumptions to understand system behavior and identify key performance factors.

1.5 Thesis Outline

This dissertation is structured as follows:

- **Chapter 2 – Background:** Introduces the technical concepts and theoretical foundations relevant to the project, including tracking, factor graphs, noise modelling, and Bayesian optimisation. This chapter provides the necessary context for understanding the methodology and experiments.
- **Chapter 3– 2D Tracking Problem:** This chapter details the design and implementation of the factor graph-based tracking system used for the experiments. It confirms the validity of the system structure with ground truth measurements. The chapter's primary focus is on demonstrating a functional system, providing a clear reference point before the automated tuning process is introduced.
- **Chapter 4 – Experiments:** This is the core of the research. This chapter focuses on applying Bayesian optimisation to automatically tune the noise model parameters of the factor graph. It presents the methodology for the BO framework, including the choice of performance metrics and the experimental setup. The chapter then presents the results of the automated tuning process, demonstrating how BO efficiently identifies optimal and statistically consistent parameter configurations.
- **Chapter 5 – Extension:** This chapter explores further ideas of aiding the tuning process, in order to better the chances of finding optimal parameters. It also talks about the effect of optimally tuned consistency metrics on the accuracy of the trajectory tracking using mean squared error.
- **Chapter 6 – Conclusion and Future Work:** This final chapter summarises the key findings and contributions of the research, highlighting the successful application of Bayesian Optimisation to automatically tune factor graph parameters. It also outlines potential directions for future work, including further optimisation strategies, the exploration of more complex tuning problems, and considerations for real-world deployment.

Chapter 2

Background and Prerequisites

This chapter provides the foundational concepts and mathematical tools necessary to understand factor graph-based state estimation. It begins with an introduction to factor graphs as a probabilistic modeling framework and explains how these models arise naturally from Bayesian inference in robotics and tracking problems. Next the discussion of linear and nonlinear system models, leading to the conditions required for statistical consistency in estimators. Finally, the introduction of key consistency metrics, Normalized Estimation Error Squared (NEES) and Normalized Innovation Squared (NIS), describing how they are used to assess and tune graph-based estimators. These ideas form the basis for the tuning and optimization methods developed in subsequent chapters.

2.1 Factor Graphs as Statistical Models

Factor graphs are a class of *probabilistic graphical models* that represent the factorization of a joint probability distribution into a product of local functions [4, 13]. This structure underpins many state estimation problems in robotics, where the goal is to infer hidden variables (e.g., robot poses, landmark locations) from noisy and partial observations.

Formally, let $X = \{X_1, X_2, \ldots, X_n\}$ be the set of random variables in the estimation problem. The joint distribution $p(X)$ can often be factorized as:

$$p(X) = \prod_{j=1}^{m} \phi_j(S_j), \tag{2.1}$$

where each factor ϕ_j is a non-negative function depending only on a subset $S_j \subseteq X$ of variables [4].

A *factor graph* is a bipartite graph $G = (V, F, E)$ with variable nodes V, factor nodes F, and edges E connecting each factor node ϕ_j to its variables S_j. This explicit encoding of local dependencies reveals the problem's conditional independence structure and enables scalable inference [13].

2.1.1 Bayesian Estimation and Factorization

Many estimation problems in robotics, including object tracking, can be formulated using *Bayesian inference*. Given a sequence of observations $Z = \{z_1, \ldots, z_t\}$ and control inputs or known dynamics $U = \{u_1, \ldots, u_t\}$, the goal is to estimate the posterior distribution over the current state X_t. This can be expressed using Bayes' rule as:

$$p(X_k \mid Z, U) = \frac{p(z_k \mid X_k)\, p(X_k \mid Z_{1:k-1}, U_{1:k})}{p(z_k \mid Z_{1:k-1}, U_{1:k})}, \tag{2.2}$$

where $p(z_k \mid X_k)$ is the likelihood of the observation given the state, and $p(X_k \mid Z_{1:k-1}, U_{1:k})$ is the predictive prior over the current state, typically computed using a motion model or process model.

To make these models explicit two functions are introduced. The motion model $f(\cdot)$, which predicts the next state from the previous state and control inputs, and the measurement model $h(\cdot)$, which predicts the expected observation from a given state. In practice, both models are subject to noise, so they are represented as conditional probability distributions. For example:

$$X_k = f(X_{k-1}, U_k) + \text{noise}, \qquad Z_k = h(X_k) + \text{noise}.$$

Under standard assumptions (Markov property, conditional independence of measurements conditioned on the states), the posterior factorizes naturally into prior, motion, and measurement terms:

$$p(X \mid Z, U) \propto \underbrace{p(X_0)}_{\text{prior}} \prod_{k=1}^{T} \underbrace{p(X_k \mid f(X_{k-1}, U_k))}_{\text{motion factors}} \prod_{k=0}^{T} \underbrace{p(Z_k \mid h(X_k))}_{\text{measurement factors}} . \tag{2.3}$$

This factorization can be directly represented by a factor graph: each conditional probability corresponds to a factor ϕ_j, and the graph's edges reflect the variables each factor depends upon [5, 14].

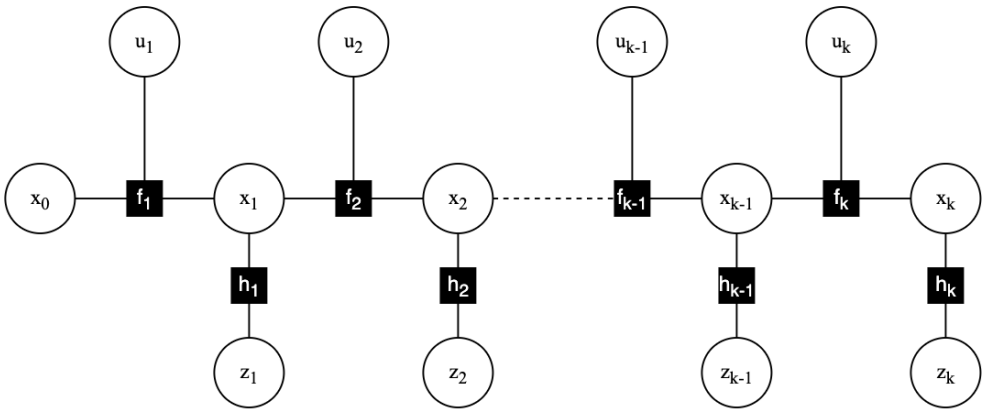


Figure 2.1: Example of a tracking factor graph corresponding to the factorization in Equation 2.3. Variable nodes represent robot poses $\mathbf{x}_{0:K}$, control inputs $\mathbf{u}_{1:k}$, and measurements $\mathbf{z}_{1:K}$. Factor nodes (black squares) correspond to priors, motion models, and measurement constraints, each encoding a conditional probability term in the posterior distribution, represented as a Markov Chain.

2.1.2 Gaussian Noise Models and MAP Estimation

In many state-estimation problems, including tracking, it is common to model both process and measurement perturbations as zero-mean Gaussian noise. This modeling choice is motivated by physical sensor characteristics, the aggregation of many independent disturbances (via the central limit theorem), and the mathematical convenience that Gaussians belong to the exponential family,

enabling tractable local approximations after linearization. Under Gaussian noise assumptions, each factor in the estimation contributes a quadratic penalty on its residual:

$$\phi_j(S_j) \propto \exp\Big(-\tfrac{1}{2}\,\|r_j(S_j)\|^2_{\Sigma_j^{-1}}\Big)\,, \tag{2.4}$$

where $r_j(S_j)$ is the residual or innovation associated with variables S_j, and Σ_j is the corresponding covariance matrix. Here, the notation $\|x\|_P^2$ denotes the squared Mahalanobis norm, defined as $\|x\|_P^2 = x^\top P x$; in this context, $P = \Sigma_j^{-1}$, the inverse covariance matrix of the residual. This connects directly to the factor-specific weighting in the Gaussian likelihood.

Assuming the conditional independences encoded by the factor graph, the posterior distribution is proportional to the product of such Gaussian terms. Therefore, maximizing the posterior (e.g., as in equation 2.3) under Gaussian noise is equivalent to minimizing a sum of squared, covariance-weighted residuals:

$$X^* = \arg\min_X \sum_{j=1}^{m} \|r_j(S_j)\|^2_{\Sigma_j^{-1}}\,. \tag{2.5}$$

After linearization, this optimization reduces to solving a sparse normal equations system with a block-structured Hessian (information) matrix. Standard nonlinear least-squares solvers, like Gauss–Newton[15] or Levenberg–Marquardt[16, 17], exploit this sparsity efficiently. Moreover, the inverse of the information matrix approximates the posterior covariance around the Maximum A Posteriori (MAP) estimate. This equivalence between statistical inference and nonlinear optimization forms the foundation of modern graph-based estimation frameworks, widely used in tracking and related robotics problems [5, 14, 18].

2.2 System Models and Statistical Consistency

To understand how a factor graph operates in a tracking problem, it is essential to define the underlying system models. This section outlines each component of the factor graph, the corresponding system equations, and the conditions required to evaluate the statistical consistency of the resulting estimator.

2.2.1 System Models

2.2.1.1 Linear System

Estimation problems in robotics are often represented using a discrete-time state-space formulation. For clarity, we first consider the linear case, which captures the key interactions between variables:

$$\mathbf{x}_k = \mathbf{F}_k\mathbf{x}_{k-1} + \mathbf{B}_k\mathbf{u}_k + \mathbf{v}_k \tag{2.6}$$

$$\mathbf{z}_k = \mathbf{H}_k\mathbf{x}_k + \mathbf{w}_k \tag{2.7}$$

where:

- $\mathbf{x}_k$: State vector at discrete time step k,
- $\mathbf{u}_k$: Control input applied at time k,
- $\mathbf{z}_k$: Observation or measurement at time k,

- $\mathbf{v}_k$: Process noise (state transition uncertainty),
- $\mathbf{w}_k$: Measurement noise (sensor uncertainty),
- $\mathbf{F}_k$: State transition matrix mapping $\mathbf{x}_{k-1}$ to $\mathbf{x}_k$ in the absence of control and noise,
- $\mathbf{B}_k$: Control input matrix mapping $\mathbf{u}_k$ into the state space,
- $\mathbf{H}_k$: Observation matrix mapping the state into the measurement space.

The process and measurement noise are assumed to be zero-mean Gaussian and mutually independent, with covariances $\mathbf{Q}_k$ and $\mathbf{R}_k$, respectively:

$$\mathbf{v}_k \sim \mathcal{N}(\mathbf{0}, \mathbf{Q}_k), \quad \mathbf{w}_k \sim \mathcal{N}(\mathbf{0}, \mathbf{R}_k).$$

2.2.1.2 Non-linear System

In practice, many robotic systems exhibit nonlinear dynamics and measurement relationships. In such cases, the system can be expressed as:

$$\mathbf{x}_k = f(\mathbf{x}_{k-1}, \mathbf{u}_k) + \mathbf{v}_k \tag{2.8}$$

$$\mathbf{z}_k = h(\mathbf{x}_k) + \mathbf{w}_k \tag{2.9}$$

where:

- $f(\cdot)$: Nonlinear state transition function,
- $h(\cdot)$: Nonlinear measurement function.

The noise assumptions remain the same: $\mathbf{v}_k$ and $\mathbf{w}_k$ are zero-mean Gaussian with covariances $\mathbf{Q}_k$ and $\mathbf{R}_k$, respectively.

2.2.1.3 Covariance Matrix Definitions

Let $\hat{\mathbf{x}}_{k|j} \triangleq \mathbb{E}[\mathbf{x}_k \mid \mathbf{z}_{1:j}]$ denote the conditional mean of the true state $\mathbf{x}_k$ given all measurements up to time j. The corresponding estimation-error covariance is

$$\mathbf{P}_{k|j} \triangleq \mathrm{Cov}\left[\mathbf{x}_k - \hat{\mathbf{x}}_{k|j} \,\middle|\, \mathbf{z}_{1:j}\right].$$

The *state estimation error* is therefore

$$\mathbf{e}_{x,k} \triangleq \mathbf{x}_k - \hat{\mathbf{x}}_{k|k},$$

a random vector of the same dimension as the state.

Similarly, for a measurement model of the form

$$\mathbf{z}_k = h(\mathbf{x}_k) + \mathbf{w}_k, \qquad \mathbf{w}_k \sim \mathcal{N}(\mathbf{0}, \mathbf{R}_k),$$

the predicted measurement is

$$\hat{\mathbf{z}}_{k|k-1} \triangleq h(\hat{\mathbf{x}}_{k|k-1}),$$

and the *measurement residual* (innovation) is

$$\mathbf{e}_{z,k} \triangleq \mathbf{z}_k - \hat{\mathbf{z}}_{k|k-1}.$$

When $h(\cdot)$ is nonlinear, it is common to approximate it locally by its Jacobian $\mathbf{H}_k$ evaluated at $\hat{\mathbf{x}}_{k|k-1}$. Under this approximation, the covariance of the innovation is

$$\mathbf{S}_{k|k-1} \triangleq \mathbf{H}_k \mathbf{P}_{k|k-1} \mathbf{H}_k^\top + \mathbf{R}_k.$$

These quantities, $\mathbf{P}_{k|j}$, $\mathbf{e}_{x,k}$, $\mathbf{e}_{z,k}$, and $\mathbf{S}_{k|k-1}$,are defined here in a general Bayesian estimation context. Although they arise naturally in recursive filters such as the Kalman filter, we introduce them primarily as notational tools to formalize the statistical consistency conditions in the next section.

2.2.2 Statistical Consistency Conditions

Once the system models are defined, statistical consistency is evalutated using three standard conditions [11]. Using the notation established above, with $\mathbf{e}_{x,k} \triangleq \mathbf{x}_k - \hat{\mathbf{x}}_{k|k}$, $\mathbf{e}_{z,k} \triangleq \mathbf{z}_k - \hat{\mathbf{z}}_{k|k-1}$, $\mathbf{P}_{k|k}$ the estimation-error covariance, and $\mathbf{S}_{k|k-1}$ the innovation covariance:

C.1 Unbiased state estimation.

$$\mathbb{E}[\mathbf{e}_{x,k}] = \mathbf{0}, \quad \forall\, k. \tag{2.10}$$

C.2 Correct error covariance (efficiency).

$$\mathbb{E}\big[\mathbf{e}_{x,k}\mathbf{e}_{x,k}^\top\big] = \mathbf{P}_{k|k}, \quad \forall\, k. \tag{2.11}$$

C.3 White Gaussian innovations.

$$\mathbf{e}_{z,k} \sim \mathcal{N}\big(\mathbf{0},\, \mathbf{S}_{k|k-1}\big)\,, \qquad \mathbb{E}[\mathbf{e}_{z,k}] = \mathbf{0}, \qquad \mathbb{E}\big[\mathbf{e}_{z,k}\mathbf{e}_{z,j}^\top\big] = \delta_{kj} \cdot \mathbf{S}_{k|k-1}, \;\; \forall\, k, j, \tag{2.12}$$

where δ_{kj} denotes the Kronecker delta, defined by $\delta_{kj} = 1$ if $k = j$ and $\delta_{kj} = 0$ otherwise; it compactly encodes that cross-covariances vanish for $k \neq j$.

2.2.3 Motivation for Consistency Tuning

In state estimation problems such as target tracking, statistical consistency refers to the agreement between the uncertainty predicted by the estimator and the actual estimation error observed in practice [11, 19]. A statistically consistent estimator provides uncertainty information that can be relied upon when making predictions, assessing performance, or adapting model parameters.

The importance of consistency becomes clear when considering the role of uncertainty in a tracking filter. If the estimator is overconfident, it will place too much trust in its own predictions and may fail to respond adequately to new measurements, allowing estimation errors to persist [11, 9]. Conversely, if the estimator is underconfident, it will overreact to measurement noise, resulting in unstable or erratic state estimates [20, 21]. Both scenarios can lead to degraded tracking performance and reduced reliability.

For systems that operate over extended time horizons, even small inconsistencies can accumulate, causing the estimated error statistics to diverge from reality [11]. This can impair downstream decision-making processes such as sensor fusion, track association, or maneuver detection [22, 23]. Ensuring statistical consistency therefore forms a critical foundation for building robust and trustworthy tracking systems.

2.3 Consistency Tuning

As established in Section 2.1.1 and formalized in the statistical consistency conditions of Section 2.2.2, a reliable estimator ideally (C.1) produces unbiased state estimates, (C.2) predicts covariances that match the true estimation error statistics, and (C.3) yields whitened, Gaussian innovations. In practice, however, these conditions may not hold exactly due to model mismatch, unmodelled dynamics, or incorrect noise parameterization.

Consistency tuning refers to the process of adjusting process and measurement noise models to better satisfy these conditions. This is particularly important in graph-based estimators, where noise parameters influence both the optimizer's convergence behaviour and the statistical trustworthiness of the resulting solution. The tuning process relies on metrics that transform multi-dimensional errors and innovations into scalar quantities with well-defined statistical distributions under the Gaussian assumptions of Section 2.1.1. Two such metrics, widely adopted are the *NEES* and the *NIS*[24].

2.3.1 Performance Metrics: NEES and NIS

The Normalized Estimation Error Squared (NEES) evaluates whether Conditions C.1 and C.2 hold by measuring the degree to which the actual estimation error is consistent with the predicted covariance:

$$\text{NEES}_k \triangleq \mathbf{e}_{x,k}^\top \mathbf{P}_{k|k}^{-1} \mathbf{e}_{x,k}, \tag{2.13}$$

where $\mathbf{e}_{x,k} = \mathbf{x}_k - \hat{\mathbf{x}}_{k|k}$ and $\mathbf{P}_{k|k}$ are defined in Section 2.2.2. For a consistent estimator, NEES_k is distributed as $\chi^2_{n_x}$ with $n_x = \dim(\mathbf{x}_k)$ Degrees of Freedom (DoF). This makes NEES a direct test of the estimator's *unbiasedness* and *covariance correctness* in the state space.

The Normalized Innovation Squared (NIS) is the corresponding metric for innovations and directly evaluates Condition C.3:

$$\text{NIS}_k \triangleq \mathbf{e}_{z,k}^\top \mathbf{S}_{k|k-1}^{-1} \mathbf{e}_{z,k}, \tag{2.14}$$

where $\mathbf{e}_{z,k}$ and $\mathbf{S}_{k|k-1}$ are the innovation and innovation covariance, respectively, also defined in Section 2.2.2. A statistically consistent estimator will yield $\text{NIS}_k \sim \chi^2_{n_z}$, where $n_z = \dim(\mathbf{z}_k)$. Since NIS can be computed without access to the true state, it is particularly useful for *online monitoring* and adaptive tuning in deployed systems.

In graph-based backends, NEES is typically evaluated in simulation or Monte Carlo studies—where ground truth is available. Similarly, NIS can also be evaluated in Monte Carlo studies, but can also be continuously monitored during real-world operation to detect deviations from the assumed noise models.

2.3.2 Theoretical Basis for the χ^2 Properties

The χ^2 properties of NEES and NIS follow from the quadratic form of Gaussian random variables (see also Section 2.2.2). If $\mathbf{e} \sim \mathcal{N}(\mathbf{0}, \mathbf{\Sigma})$, then the scalar

$$q = \mathbf{e}^\top \mathbf{\Sigma}^{-1} \mathbf{e}$$

follows a χ^2_d distribution with $d = \dim(\mathbf{e})$ DoF. This is shown via spectral decomposition $\mathbf{\Sigma} = \mathbf{U}\mathbf{\Lambda}\mathbf{U}^\top$ and the whitening transformation $\mathbf{y} = \mathbf{\Sigma}^{-1/2}\mathbf{e}$[25], yielding $q = \sum_{i=1}^{d} y_i^2$ with $y_i \sim \mathcal{N}(0, 1)$.

Applying this to the metrics above:

$$\text{NEES}_k \sim \chi^2_{n_x} \quad \text{if Conditions C.1 and C.2 hold,}$$

$$\text{NIS}_k \sim \chi^2_{n_z} \quad \text{if Condition C.3 holds.}$$

These results are exact for linear Gaussian systems and hold approximately for nonlinear systems under the local-linearization assumptions discussed in Section 2.2.1.2.

The rigorous statistical properties of NEES and NIS enable formal *hypothesis testing* for estimator consistency, where the NIS statistic will be applied in Chapter 3 through offline Monte Carlo simulations for tuning.

2.4 Graph Tuning

Graph tuning in the context of factor graphs for state estimation, such as in target tracking, involves optimizing the noise parameters (e.g., $\mathbf{Q}_k, \mathbf{R}_k$ in the system models) to predict their values when their true values are unknown. The procedure typically entails running multiple Monte Carlo simulations under a "truth model" to generate ground-truth trajectories and measurements, computing consistency metrics like NEES and NIS, and iteratively adjusting the parameters until the metrics align with their theoretical distributions. The goal is to estimate the noise parameters that best reflect the underlying system dynamics, thereby enhancing the reliability and performance of the graph-based estimator in real-world deployments, even when the true noise characteristics are not directly available.

2.4.1 Monte Carlo-Based Consistency Metrics

Given N Monte Carlo runs, the *average state error* and *average innovation* at time k are computed as:

$$\bar{\mathbf{e}}_{x,k} = \frac{1}{N} \sum_{i=1}^{N} \mathbf{e}_{x,k}^{(i)}, \tag{2.15}$$

$$\bar{\mathbf{e}}_{z,k} = \frac{1}{N} \sum_{i=1}^{N} \mathbf{e}_{z,k}^{(i)}, \tag{2.16}$$

with corresponding sample variances providing error bars to quantify Monte Carlo uncertainty.

The *time-averaged NEES* and *time-averaged NIS* are:

$$\tilde{\epsilon}_x = \frac{1}{T} \sum_{k=1}^{T} \bar{\epsilon}_{x,k}, \tag{2.17}$$

$$\tilde{\epsilon}_z = \frac{1}{T} \sum_{k=1}^{T} \bar{\epsilon}_{z,k}, \tag{2.18}$$

where T is the number of time steps.

2.4.2 Joint NEES/NIS Metrics

To mitigate "exploding/vanishing" effects in long horizons or high-dimensional states, the Joint Normalized Estimation Error Squared (JNEES) and Joint Normalized Innovation Squared (JNIS)

are defined as [26]:

$$J_{\text{NEES}} = \left| \log \left(\frac{\tilde{\epsilon}_x}{n_x} \right) \right|, \tag{2.19}$$

$$J_{\text{NIS}} = \left| \log \left(\frac{\tilde{\epsilon}_z}{n_z} \right) \right|, \tag{2.20}$$

where n_x and n_z are the respective DoF.

The intuition behind the logarithm is that it normalizes the ratio between the time-averaged metric and the expected χ^2 mean. When the numerator ($\tilde{\epsilon}_x$ or $\tilde{\epsilon}_z$) is close to the denominator (n_x or n_z), the logarithm approaches zero, yielding a low score. Minimizing J_{NEES} and J_{NIS} therefore encourages both accurate scaling and stable behavior across Monte Carlo runs or long trajectories.

2.4.3 Consistent NEES/NIS Metrics

Chen et al. [8] extended the standard NEES and NIS metrics to the Consistent Normalized Estimation Error Squared (CNEES) and Consistent Normalized Innovation Squared (CNIS), which incorporate both the mean and variance of the Monte Carlo samples. The first term in each metric measures the deviation of the time-averaged NEES/NIS from its expected χ^2 mean (as in the standard JNEES/JNIS), while the second term captures the variability of the metric across runs. Including the variance term helps prevent overconfidence in the estimator when the sample spread is large, providing a more robust measure of consistency for tuning purposes.

$$C_{\text{NEES}} = \left| \log \left(\frac{\tilde{\epsilon}_x}{n_x} \right) \right| + \left| \log \left(\frac{\tilde{S}_x}{2n_x} \right) \right|, \tag{2.21}$$

$$C_{\text{NIS}} = \left| \log \left(\frac{\tilde{\epsilon}_z}{n_z} \right) \right| + \left| \log \left(\frac{\tilde{S}_z}{2n_z} \right) \right|, \tag{2.22}$$

where the time-averaged sample variances are defined for the filtering case as:

$$\tilde{S}_x = \frac{1}{T(N-1)} \sum_{k=1}^{T} \sum_{i=1}^{N} \left(\epsilon_{i,x,k} - \bar{\epsilon}_{x,k} \right)^2, \tag{2.23}$$

$$\tilde{S}_z = \frac{1}{T(N-1)} \sum_{k=1}^{T} \sum_{i=1}^{N} \left(\epsilon_{i,z,k} - \bar{\epsilon}_{z,k} \right)^2. \tag{2.24}$$

It is important to note that these formulations are directly applicable to sequential filters, where each time step produces a single state estimate and covariance. In graph-based or smoothing approaches, which optimize over entire trajectories, the direct application of these metrics requires computing full-trajectory statistics. CNEES and CNIS are evaluated using the complete trajectory estimates and the full information matrix obtained from the factor graph optimization. This approach allows the metrics to be used for tuning and assessment while acknowledging that the statistical properties may differ from sequential filtering approaches.

These metrics can be further extended over multiple discretization intervals Δt_n:

$$C_{\text{NEES,total}} = \sum_{n=1}^{m} C_{\text{NEES}}(\Delta t_n), \tag{2.25}$$

$$C_{\text{NIS,total}} = \sum_{n=1}^{m} C_{\text{NIS}}(\Delta t_n). \tag{2.26}$$

Using multiple discretization intervals helps mitigate scale ambiguities that can arise when both process and measurement noise parameters ($\mathbf{Q}$ and $\mathbf{R}$) are adjusted simultaneously [26].

2.4.4 Extension to Graph-Based Tracking

Extending the NEES/NIS framework to factor-graph-based estimators requires care. Unlike sequential filters, graph-based methods optimize over the full trajectory simultaneously, and each factor can involve multiple state variables. The total graph cost for a set of states can be expressed as a sum of factor residuals, analogous to the squared, covariance-weighted residuals used in filtering (see Equation 2.3):

$$f(X) = \sum_{j=1}^{m} r_j(S_j)^\top \Sigma_j^{-1} r_j(S_j), \tag{2.27}$$

where $r_j(S_j)$ is the residual for factor j involving the subset of states S_j, and Σ_j is the associated covariance.

Proposition 3: Cost at the ground truth. If the graph is initialized at the ground-truth trajectory x_* without performing optimization, the resulting cost

$$f_* = f(x_*) \tag{2.28}$$

captures only the contribution of measurement noise. Under Gaussian, independent noise assumptions, this cost follows a χ^2 distribution with degrees of freedom equal to the total dimensionality of all measurement constraints in the graph:

$$f_* \sim \chi^2_{n_*}, \quad n_* = \dim(z), \tag{2.29}$$

where $\dim(z)$ sums over all measurements, including odometry and sensor factors. This represents a baseline cost due solely to measurement noise. Significant deviations of f_* from this distribution indicate that the assumed noise parameters may not match the actual noise.

Proposition 4: Cost at the optimized estimate. After optimizing the graph to obtain the maximum likelihood estimate $x^\dagger$, the cost

$$f^\dagger = f(x^\dagger) \sim \chi^2_{n^\dagger}, \quad n^\dagger = \dim(z) - \dim(x), \tag{2.30}$$

has reduced degrees of freedom because the optimization explains part of the variability through the state estimates. Here $\dim(x)$ is the total number of estimated state dimensions. A properly tuned graph should yield $f^\dagger$ consistent with this reduced χ^2 distribution. Systematically low values indicate overconfidence (underestimated noise), while high values suggest overly conservative noise assumptions.

Discussion. These propositions establish a connection between graph-based estimation and the classical NEES/NIS framework. By comparing the graph costs at the ground-truth trajectory and at the optimized estimate with their expected χ^2 distributions, statistical consistency in graphs can be evaluated. The key difference from the filtering case is that $\dim(z)$ counts all factor constraints rather than individual measurements, with correlations between shared variables accounted for implicitly through the optimization. The ratio between $\dim(z)$ and $\dim(x)$ also plays an important role. In dense graphs, where $\dim(z) \gg \dim(x)$, the reduction in degrees of freedom is negligible, whereas in sparse graphs, where $\dim(z)$ is closer to $\dim(x)$, the optimized cost becomes more sensitive to the assumed noise. This motivates the use of consistency tests that explicitly incorporate graph structure.

Practical role of CNEES/CNIS. Propositions 3 and 4 describe the expected behaviour of graph costs under correct noise assumptions, but they do not provide a direct procedure for parameter tuning. The CNEES and CNIS metrics address this by reducing consistency to scalar statistics that can be compared with their theoretical distributions. This reformulation allows the tuning task to be posed as an optimization problem, in which process and measurement noise parameters are adjusted to minimise the gap between empirical and expected consistency measures. In this way, CNEES and CNIS translate the theoretical conditions for consistency into a practical tool for noise parameter tuning.

2.5 Baysian Optimization for Parameter Tuning

2.5.1 Motivation

In state estimation problems, such as target tracking using factor graphs, the noise parameters, typically the process noise covariance $\mathbf{Q}$ and measurement noise covariance $\mathbf{R}$, play a critical role in determining estimator performance and consistency [11]. Manual tuning of these parameters is often impractical due to the complexity of the underlying models and the sensitivity of the system to small perturbations. Key challenges include:

1. the stochastic nature of consistency objectives like CNEES and CNIS, which are computed via Monte Carlo simulations and therefore contain inherent noise
2. the high computational cost of each evaluation, which involves repeated factor graph optimizations over multiple trajectories
3. the absence of reliable gradient information, as the objective is effectively a noisy black-box function

While the project's work focuses on Bayesian Optimization (BO), several alternative approaches have been investigated for parameter tuning in state estimation. Gradient-based methods, though computationally efficient, prove inadequate for consistency objectives due to their non-differentiable nature and the absence of analytical gradients [12]. Evolutionary algorithms, including genetic algorithms and particle swarm optimization, offer derivative-free alternatives but often require extensive parameter tuning themselves and may converge to suboptimal solutions [27]. For instance, multi-objective genetic algorithms have been used for Kalman filter tuning in battery state estimation [28]. Reinforcement learning approaches have shown promise for adaptive parameter selection in some contexts, though they typically require significant training data and may not generalize

well across different system configurations [29]. Multi-objective optimization methods attempt to balance consistency with other performance metrics, though they introduce additional complexity in objective function design and solution selection [28].

BO provides an effective solution for such scenarios. It is specifically designed for optimizing expensive, black-box functions where evaluations are noisy and costly [30, 31, 32]. By treating the objective as a stochastic black box, BO efficiently explores the parameter space with a minimal number of evaluations, making it well-suited for tuning noise parameters under resource constraints. Recent applications in Kalman filter tuning have demonstrated BO's ability to handle noisy, stochastic objectives derived from consistency metrics, while avoiding local minima that can trap conventional optimizers [26, 12, 8].

2.5.2 Bayesian Optimisation Framework with Gaussian Processes

At the core of BO lies the use of a surrogate model to approximate the unknown objective function. Gaussian Process (GP) are commonly employed for this purpose due to their flexibility in modeling uncertainty and providing probabilistic predictions [33]. Formally, the objective function $f(\mathbf{q})$, where $\mathbf{q}$ represents the noise parameters (e.g., elements of Q and R), is modeled as a draw from a GP prior:

$$f(\mathbf{q}) \sim \mathcal{GP}(\mu(\mathbf{q}), k(\mathbf{q}, \mathbf{q}')),$$

with mean function $\mu(\mathbf{q})$ (often set to zero for simplicity) and covariance kernel

$$k(\mathbf{q}, \mathbf{q}') = \sigma^2 \exp\left(-\frac{\|\mathbf{q} - \mathbf{q}'\|^2}{2\rho^2}\right),$$

where σ^2 is the signal variance and ρ is the lengthscale controlling smoothness.

The GP is initialized with a small set of observations $\mathcal{D}_n = \{(\mathbf{q}_i, y_i)\}_{i=1}^n$, where $y_i = f(\mathbf{q}_i) + \epsilon_i$ and $\epsilon_i \sim \mathcal{N}(0, \sigma_n^2)$ accounts for observation noise, which is particularly relevant for stochastic consistency metrics derived from Monte Carlo simulations. The posterior GP is refined using Bayes' rule. For a new point $\mathbf{q}_*$, the predictive mean and variance are:

$$\mu_n(\mathbf{q}_*) = \mathbf{k}_*^\top (\mathbf{K} + \sigma_n^2 \mathbf{I})^{-1} \mathbf{y},$$

$$\sigma_n^2(\mathbf{q}_*) = k(\mathbf{q}_*, \mathbf{q}_*) - \mathbf{k}_*^\top (\mathbf{K} + \sigma_n^2 \mathbf{I})^{-1} \mathbf{k}_*,$$

where $\mathbf{K}$ is the $n \times n$ kernel matrix with entries $K_{ij} = k(\mathbf{q}_i, \mathbf{q}_j)$, $\mathbf{k}_* = [k(\mathbf{q}_*, \mathbf{q}_1), \ldots, k(\mathbf{q}_*, \mathbf{q}_n)]^\top$, and $\mathbf{y} = [y_1, \ldots, y_n]^\top$.

Figure 2.2 illustrates a 1D Gaussian Process surrogate, showing how the model predicts the mean objective function while representing uncertainty in regions without observations.

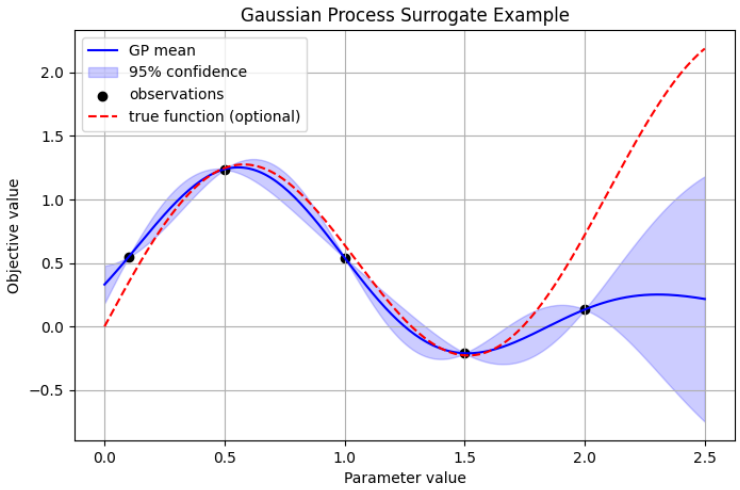


Figure 2.2: Illustration of a 1D Gaussian Process surrogate for Bayesian optimisation. Black circles show observed evaluations of the objective function (e.g., Monte Carlo estimates of a consistency metric). The blue line represents the GP predictive mean, while the shaded region corresponds to the 95% confidence interval ($\mu \pm 2\sigma$). The red dashed line is the true function, included for illustration. The GP explicitly models uncertainty between known evaluations, guiding the choice of the next sample through acquisition functions.

Hyperparameters like σ^2, ρ, and σ_n^2 are typically optimized by maximizing the marginal log-likelihood:

$$\log p(\mathbf{y} \mid \mathbf{q}_{1:n}) = -\frac{1}{2}\mathbf{y}^\top (\mathbf{K} + \sigma_n^2 \mathbf{I})^{-1}\mathbf{y} - \frac{1}{2}\log|\mathbf{K} + \sigma_n^2\mathbf{I}| - \frac{n}{2}\log 2\pi.$$

2.5.3 Acquisition Functions

Acquisition functions are pivotal in BO, as they balance exploration (sampling uncertain regions) and exploitation (focusing on promising areas) to select the next parameter point $\mathbf{q}_{t+1}$ [32]. Assuming minimization of the objective $f(\mathbf{q})$, popular choices include:

Expected Improvement (EI): Measures the expected gain over the current best value

$$f(\mathbf{q}^+) = \min_{i=1,\ldots,t} f(\mathbf{q}_i).$$

Under the posterior GP, the EI has a closed-form expression:

$$\alpha_{\mathrm{EI}}(\mathbf{q}) = \mathbb{E}[\max(0, f(\mathbf{q}^+) - f(\mathbf{q}))] = \sigma_t(\mathbf{q})\left(Z\Phi(Z) + \phi(Z)\right),$$

where $Z = \frac{f(\mathbf{q}^+) - \mu_t(\mathbf{q})}{\sigma_t(\mathbf{q})}$, and Φ and ϕ are the cumulative distribution and probability density functions of the standard normal distribution, respectively. A small $\xi > 0$ can be added to Z as $Z = \frac{f(\mathbf{q}^+) + \xi - \mu_t(\mathbf{q})}{\sigma_t(\mathbf{q})}$ to encourage more exploration [34].

Lower Confidence Bound (LCB): For minimization tasks, the acquisition function is

$$\alpha_{\mathrm{LCB}}(\mathbf{q}) = \mu_t(\mathbf{q}) - \kappa\sigma_t(\mathbf{q}),$$

where $\kappa \geq 0$ controls the exploration–exploitation trade-off: larger values of κ emphasize explo-

ration by sampling uncertain regions, while smaller values favor exploitation of regions with low predicted mean. Theoretical regret bounds exist for LCB with appropriately scheduled κ [35].

Comparison of behaviours. The choice of acquisition function affects the sampling strategy:

- **EI** focuses on regions where the expected improvement over the current best is high. It naturally balances exploration and exploitation but can aggressively exploit promising regions.
- **LCB** explicitly trades off mean prediction and uncertainty via the κ parameter. It can be tuned to favor more exploration in highly uncertain or noisy settings, or more exploitation in well-understood regions.

Testing both acquisition functions allows the optimizer to adapt to different objective landscapes. In the context of consistency tuning, where objective evaluations are noisy and the landscape may be multimodal, comparing EI and LCB can reveal which strategy provides more robust convergence.

2.5.4 Application to Noise Parameter Tuning

In the context of tuning factor graphs for tracking, the parameters of interest include the process noise intensity $\mathbf{Q}$ and measurement noise variance $\mathbf{R}$. The objective is to minimize aggregated consistency metrics, such as CNEES or CNIS, which quantify deviations from statistical consistency across multiple simulated trajectories [11, 8]. The process unfolds as follows

1. Initialize a GP surrogate and select initial candidate parameters $\mathbf{q}_0$ (e.g., via Latin hypercube sampling)
2. Construct Q and R matrices from $\mathbf{q}$
3. Perform factor graph optimization on simulated data
4. Compute the consistency score (e.g., $C_{\text{NEES,total}}$) averaged over trajectories
5. Update the GP with the new observation
6. Use an acquisition function to propose the next $\mathbf{q}_{t+1}$
7. Repeat until convergence or budget exhaustion.

This yields a data-driven calibration of noise statistics, enhancing estimator reliability without exhaustive grid searches [26, 12, 8]. In practice, constraints on parameter positivity (e.g., via log-transforms) and multi-fidelity evaluations can further improve efficiency [31]. Extensions using Student-t processes provide additional robustness to outliers in noisy objectives [8].

2.6 Summary and Transition

This chapter has established the theoretical foundations necessary for understanding automated parameter tuning in factor graph-based state estimation. It began by introducing factor graphs as probabilistic graphical models, highlighting how these structures represent the factorization of joint probability distributions in estimation problems. The discussion then addressed system modeling, covering both linear and nonlinear formulations, and introduced the three fundamental conditions for statistical consistency: unbiased estimation, correct error covariance, and white

Gaussian innovations. Building on this framework, consistency metrics such as NEES and NIS were presented, along with their advanced variants (CNEES/CNIS) that account for both mean and variance considerations. The chapter concluded with an overview of automated parameter tuning strategies, with particular emphasis on BO, which employs GP surrogates to efficiently explore parameter spaces for expensive, black-box consistency objectives. This progression from theoretical principles to methodological considerations provides the foundation for the experimental work and contributions developed in the following chapters.

Chapter 3

2D-Tracking Problem Setup

This chapter describes the tracking problem used to evaluate the factor-graph estimator. Two dynamical systems are considered: a linear Constant Velocity (CV) model and a nonlinear Constant Turn rate (CT) model. For each system the chapter presents the system equations, the process and observation noise models, a short trajectory example for intuition, and the ground-truth consistency tests.

3.1 Common notation and basic assumptions

Throughout the chapter the discrete-time state is written as the column vector $\mathbf{x}_k = [x,\, y,\, v_x,\, v_y]^\top$ with x and y giving the Cartesian position and v_x and v_y giving the Cartesian velocity. Measurements are denoted $\mathbf{z}_k$, control inputs $\mathbf{u}_k$, process noise $\mathbf{v}_k \sim \mathcal{N}(\mathbf{0}, \mathbf{Q}(\Delta t))$ and measurement noise $\mathbf{w}_k \sim \mathcal{N}(\mathbf{0}, \mathbf{R})$. Process covariances $\mathbf{Q}$ depend on the timestep Δt; measurement covariances $\mathbf{R}$ do not, which will be later discussed when formulating both matrix frames. Factor-graph residuals are weighted by these covariances during least-squares optimisation.

Cross-references to Proposition dimensionality are given to the expressions in Chapter 2: (2.29) and (2.30). Concretely, the total residual dimension used in the global χ^2-type checks is obtained by summing process residual dimensions and measurement residual dimensions (this total is $\dim(\mathbf{z})$ below).

3.2 Linear model

3.2.1 System model

The linear baseline uses a constant-velocity model with state

$$\mathbf{x}_k = [x,\, y,\, v_x,\, v_y]^\top. \tag{3.1}$$

The discrete-time linear state update is

$$\mathbf{x}_k = \mathbf{F}_k \mathbf{x}_{k-1} + \mathbf{B}_k \mathbf{u}_k + \mathbf{v}_k, \tag{3.2}$$

with the standard constant-velocity transition and input matrices

$$\mathbf{F} = \begin{bmatrix} 1 & 0 & \Delta t & 0 \\ 0 & 1 & 0 & \Delta t \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}, \qquad \mathbf{B} = \begin{bmatrix} \frac{\Delta t^2}{2} & 0 \\ 0 & \frac{\Delta t^2}{2} \\ \Delta t & 0 \\ 0 & \Delta t \end{bmatrix}.$$

Role of $\mathbf{F}, \mathbf{B}, \mathbf{Q}, \mathbf{R}$. During propagation the predicted state is $\mathbf{F}\mathbf{x}_{k-1} + \mathbf{B}\mathbf{u}_k$ and the process residual used by the binary (process) factor is $\mathbf{x}_k - (\mathbf{F}\mathbf{x}_{k-1} + \mathbf{B}\mathbf{u}_k)$. That residual has covariance $\mathbf{Q}(\Delta t)$ and thus $\mathbf{Q}$ directly determines the process-factor weight in the least-squares problem. Measurement factors are formed from the residual $\mathbf{z}_k - \mathbf{H}\mathbf{x}_k$ and are weighted by $\mathbf{R}$. These two covariances therefore control the trade-off between following the dynamical model and fitting sensor data.

3.2.2 Process model

The discrete-time process covariance used for the linear model is the integrated constant-acceleration form (parametrised by a single isotropic intensity V):

$$\mathbf{Q}_{\mathrm{CV}}(\Delta t; V) = \begin{bmatrix} \frac{\Delta t^3}{3} V_x & 0 & \frac{\Delta t^2}{2} V_x & 0 \\ 0 & \frac{\Delta t^3}{3} V_y & 0 & \frac{\Delta t^2}{2} V_y \\ \frac{\Delta t^2}{2} V_x & 0 & \Delta t\, V_x & 0 \\ 0 & \frac{\Delta t^2}{2} V_y & 0 & \Delta t\, V_y \end{bmatrix}, \tag{3.3}$$

with $V_x = V_y = V$ in the isotropic case used in experiments. Each process factor uses the $\mathbf{Q}$ computed with the local Δt.

3.2.3 Observation model

Position measurements are Cartesian and linear in the state:

$$\mathbf{z}_k = \mathbf{H}\mathbf{x}_k + \mathbf{w}_k, \qquad \mathbf{H} = \begin{bmatrix} \mathbf{I}_{2\times 2} & \mathbf{0}_{2\times 2} \end{bmatrix}, \tag{3.4}$$

and

$$\mathbf{R} = \begin{bmatrix} \sigma_x^2 & 0 \\ 0 & \sigma_y^2 \end{bmatrix}. \tag{3.5}$$

Where once again $\sigma_x^2 = \sigma_y^2 = \sigma^2$. These measurement factors constrain the positional components of $\mathbf{x}_k$ only.

3.2.4 Data generation

To evaluate the estimator performance, Monte Carlo trajectories were generated using the linear system model described above. Each trajectory consists of $T = 100$ time steps with $\Delta t = 1$, starting from initial position $(0, 0)$ with velocity $(1, 1)$. Process noise was sampled from $\mathbf{v}_k \sim \mathcal{N}(\mathbf{0}, \mathbf{Q}_{\mathrm{CV}}(\Delta t; V = 1))$ at each time step, and measurement noise from $\mathbf{w}_k \sim \mathcal{N}(\mathbf{0}, \mathbf{R})$ with $\sigma^2 = 2$. A total of $N = 500$ independent Monte Carlo runs were generated using different random seeds to ensure statistical diversity in the test ensemble.

The data generation process follows the system equations exactly: $\mathbf{x}_k = \mathbf{F}\mathbf{x}_{k-1} + \mathbf{B}\mathbf{u}_k + \mathbf{v}_k$ with $\mathbf{u}_k = \mathbf{0}$ (constant velocity), and measurements $\mathbf{z}_k = \mathbf{H}\mathbf{x}_k + \mathbf{w}_k$. This ensures perfect model consistency between data generation and the factor graph estimator assumptions.

3.2.5 Trajectory example (linear)

Trajectories under the CV model are straight with the addition of process noise on top. A representative simulated trajectory (single Monte Carlo run) is shown in Figure 3.1 to illustrate how process noise $\mathbf{Q}$ produces spread over time and how measurements scatter around the true path according to $\mathbf{R}$.

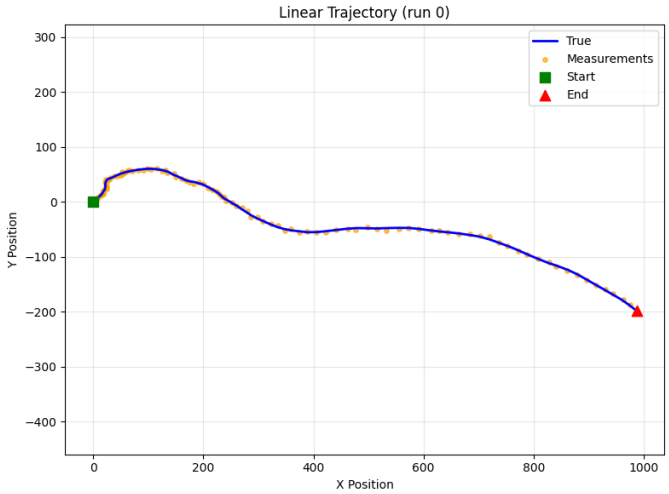


Figure 3.1: Representative CV trajectory with process and measurement noise (single Monte Carlo run). The run uses the initial start position $(0, 0)$ with velocity $(1, 1)$.

3.2.6 Linear ground-truth tests

Baseline parameters set for the ground truth tests are $V = 1$, $\sigma^2 = 2$, $T = 100$, $\Delta t = 1$), and Monte Carlo runs $N = 500$, giving the following $\mathbf{Q}$ and $\mathbf{R}$ covariance matrix:

$$\mathbf{Q} = \begin{bmatrix} \frac{1}{3} & 0 & \frac{1}{2} & 0 \\ 0 & \frac{1}{3} & 0 & \frac{1}{2} \\ \frac{1}{2} & 0 & 1 & 0 \\ 0 & \frac{1}{2} & 0 & 1 \end{bmatrix}, \qquad \mathbf{R} = \begin{bmatrix} 2 & 0 \\ 0 & 2 \end{bmatrix}, \tag{3.6}$$

Proposition 3 (ideal conditions). The total residual degrees of freedom used in the graph-level χ^2 statistic is the sum of process residual dimensions and measurement residual dimensions. Using the notation from Chapter 2:

$$\dim(\mathbf{z}) = n_x(T-1) + n_z T, \tag{3.7}$$

where n_x is the state dimension and n_z the measurement dimension. For $n_x = 4$, $n_z = 2$ (based on the state and observation dimensions), $T = 100$ this gives $\dim(\mathbf{z}) = 596$, and theoretical covariance $2 \cdot 596 = 1192$. The observed NIS values over 500 runs (Figure 3.2) show mean and covariance close

to theory; the computed CNIS was 0.049929, leading to the claim that the system has achieved statistical stability using the ground truth noise covariance.

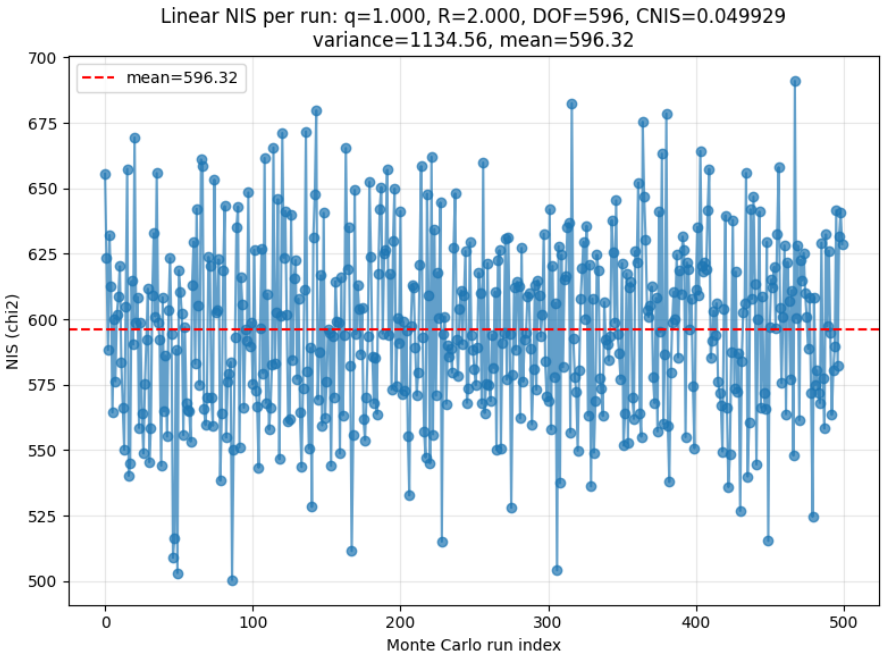


Figure 3.2: NIS values for the linear system under ideal conditions (Proposition 3).

Proposition 4 (non-ideal conditions). When the initial state is treated as unknown the effective degrees of freedom are reduced by $\dim(\mathbf{x})$. The expression is

$$\dim(\mathbf{z}) - \dim(\mathbf{x}) = \big(n_x(T-1) + n_z T\big) - n_x T, \tag{3.8}$$

which for the same numeric values mentioned for proposition 3 yields 196 (theoretical covariance 392). The observed NIS distribution is shown in Figure 3.3; CNIS remained low (0.053066) and results are consistent with finite-sample deviations.

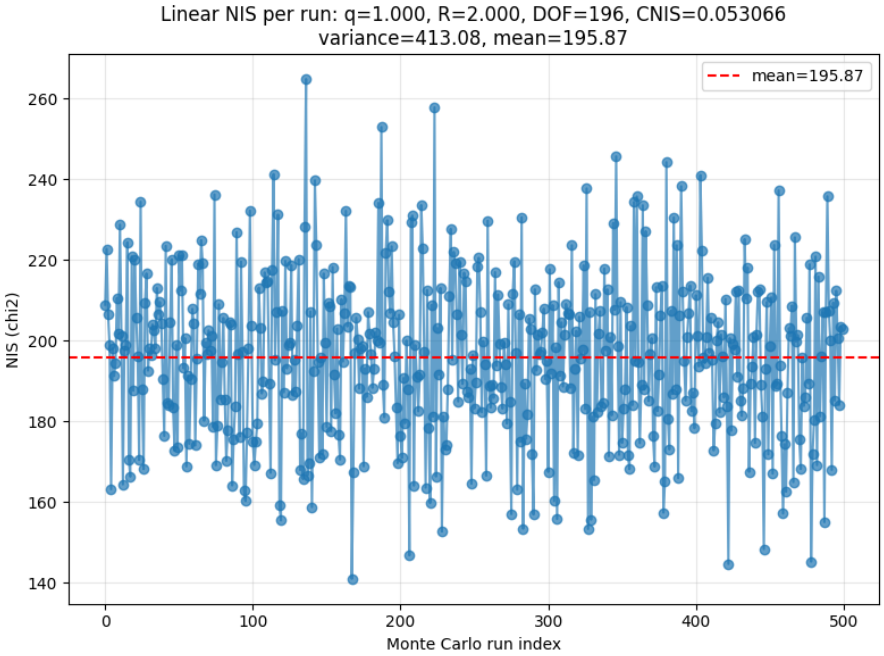


Figure 3.3: NIS values for the linear system under non-ideal conditions (Proposition 4).

Covariance discrepancies and implications (linear). Although mean values of the statistics closely match theory, the empirical covariance occasionally differs from the theoretical covariance by more than would be expected from sampling noise alone. Such discrepancies suggest that one or more Gaussian assumptions (for example, the distribution of measurement or process residuals) might not fully hold in the generated ensembles. This observation does not imply an implementation error: the factor-graph estimator and its weighting by $\mathbf{Q}$ and $\mathbf{R}$ are functioning as intended, and the mean behaviour is consistent with expectation. However, the mismatch in covariance limits the strength of any claim that the system is fully statistically functional in the strictest sense. A more detailed Monte Carlo diagnostic (convergence with Pearson tests) is therefore necessary to determine whether residuals are Gaussian and to guide any modelling changes.

3.3 Nonlinear model

The nonlinear section now describes only the differences relative to the linear model; where no difference is stated the linear model treatment (state vector, measurement model, dimensionality expressions, and the NIS/NEES testing procedure) applies unchanged.

3.3.1 System model differences

The nonlinear constant-turn model uses the same state vector $\mathbf{x}_k = [x, y, v_x, v_y]^\top$ but a nonlinear transition:

$$\mathbf{x}_k = f(\mathbf{x}_{k-1}, \omega) + \mathbf{v}_k, \tag{3.9}$$

with

$$f(\mathbf{x}_k, \omega) = \begin{bmatrix} x_k + \frac{v}{\omega}\big[\sin(\theta_k + \omega\Delta t) - \sin(\theta_k)\big] \\ y_k - \frac{v}{\omega}\big[\cos(\theta_k + \omega\Delta t) - \cos(\theta_k)\big] \\ v\cos(\theta_k + \omega\Delta t) \\ v\sin(\theta_k + \omega\Delta t) \end{bmatrix},$$

where $v = \sqrt{v_x^2 + v_y^2}$ and $\theta_k = \operatorname{atan2}(v_y, v_x)$ and ω is used as the control input $\mathbf{u}_k$. Iterative optimisation (e.g. Gauss–Newton) is required to solve the resulting nonlinear least-squares problem inside the factor graph.

3.3.2 Process covariance for the CT model

To reflect rotational dynamics the CT process covariance is formed by rotating the CV covariance into the turning frame, effectively mapping the covariance of the observation correctly in the world frame.. Define the planar rotation

$$\theta = \omega\frac{\Delta t}{2}, \qquad \mathcal{R}(\theta) = \begin{bmatrix} \cos\theta & -\sin\theta \\ \sin\theta & \cos\theta \end{bmatrix}, \qquad \mathbf{T}(\theta) = \operatorname{diag}\big(\mathcal{R}(\theta),\, \mathcal{R}(\theta)\big).$$

Then

$$\mathbf{Q}_{\mathrm{CT}}(V, \Delta t, \omega) = \mathbf{T}(\theta)\ \mathbf{Q}_{\mathrm{CV}}(V, \Delta t)\ \mathbf{T}(\theta)^{\top}. \tag{3.10}$$

Each nonlinear process factor therefore uses the rotated covariance $\mathbf{Q}_{\mathrm{CT}}$ computed with the local ω and Δt.

3.3.3 Observation model (unchanged)

Measurements remain Cartesian position only and use the same linear measurement model $\mathbf{z}_k = \mathbf{H}\mathbf{x}_k + \mathbf{w}_k$ with $\mathbf{H}$ and $\mathbf{R}$ as in (3.4)–(3.5). This choice keeps the comparison between linear and nonlinear dynamics focused on the propagation rather than sensing.

3.3.4 Data generation (nonlinear)

Nonlinear trajectories were generated using the same baseline parameters as the linear case ($V = 1$, $\sigma^2 = 2$, $T = 100$, $N = 500$ runs) but with the constant-turn dynamics. Each trajectory starts from the same initial conditions and follows the nonlinear state update $\mathbf{x}_k = f(\mathbf{x}_{k-1}, \omega) + \mathbf{v}_k$ with turn rate $\omega = 0.1$ rad/s. Process noise was sampled from $\mathbf{v}_k \sim \mathcal{N}(\mathbf{0}, \mathbf{Q}_{\mathrm{CT}}(V, \Delta t, \omega))$ using the rotated covariance matrix, while measurement noise remained unchanged from the linear case.

The nonlinear data generation ensures consistency between the true dynamics and the factor graph model assumptions, providing a fair baseline for evaluating estimator performance under curved motion. The same random seeds were used across linear and nonlinear experiments to enable direct comparison of consistency metrics.

3.3.5 Trajectory example (nonlinear)

A representative CT trajectory demonstrating curvature and the effect of $\mathbf{Q}_{\mathrm{CT}}$ is shown in Figure 3.4. The figure is intended to give intuition for how turning affects the spread of ground-truth trajectories relative to the CV case.

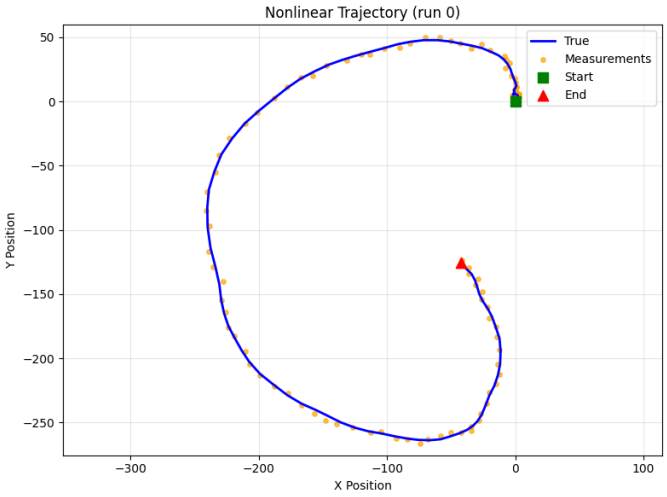


Figure 3.4: Representative CT trajectory with process and measurement noise (single Monte Carlo run).

3.3.6 Nonlinear ground-truth tests

Ground-truth Monte Carlo tests were performed with the same baseline parameters as the linear experiments, Section 3.2.6. Because the measurement model and the dimensionality bookkeeping ($\dim(\mathbf{z})$, $\dim(\mathbf{z}) - \dim(\mathbf{x})$) are unchanged, the same theoretical degrees-of-freedom expressions apply (see (3.7) and (3.8)).

Proposition 3 (ideal conditions). The NIS distribution under ideal conditions is plotted in Figure 3.5 and is close to theory; this is expected since the same measurement model and noise seeds were used to compare methods.

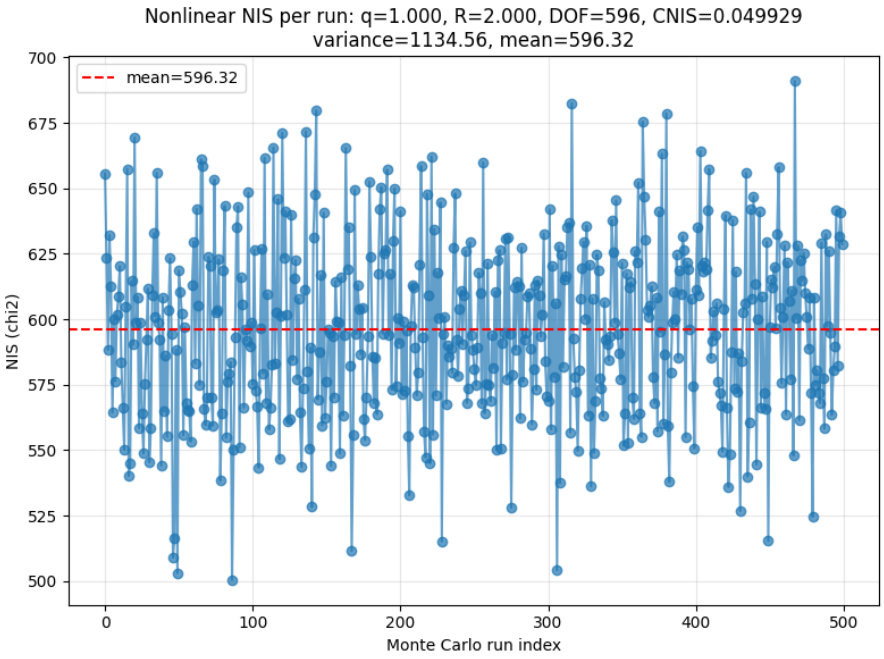


Figure 3.5: NIS values for the nonlinear system under ideal conditions (Proposition 3).

Proposition 4 (non-ideal conditions). Under non-ideal initialisation the nonlinear case shows a slightly larger observed covariance (approximately 413.99 vs theoretical 392), and CNIS=0.055313. These small deviations are consistent with finite-sample effects and the additional optimisation complexity introduced by the nonlinear dynamics (Figure 3.6).

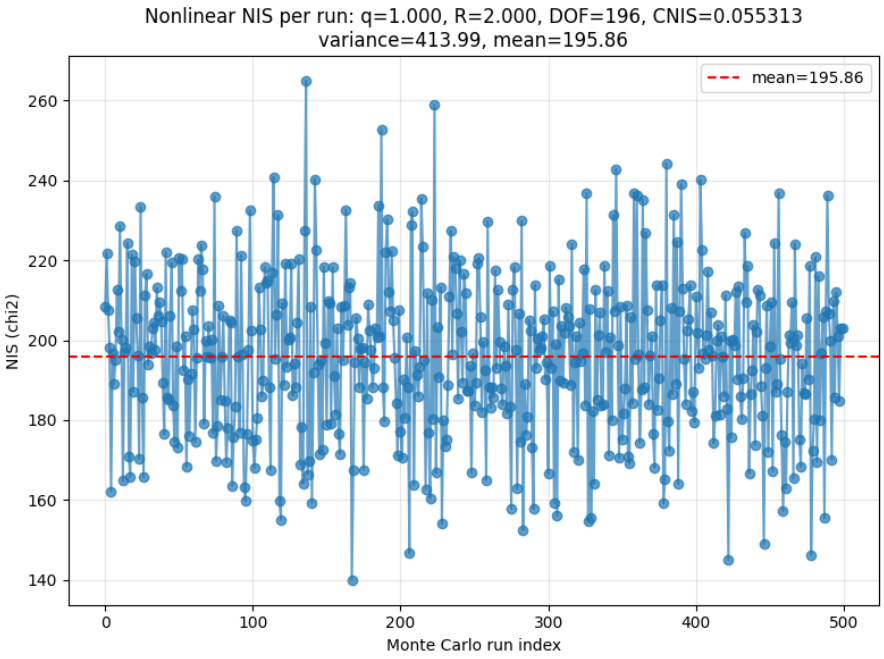


Figure 3.6: NIS values for the nonlinear system under non-ideal conditions (Proposition 4).

Covariance discrepancies and implications (nonlinear). As with the linear experiments, the nonlinear runs show occasional covariance deviations larger than expected from sampling alone. Because the nonlinear propagation and optimisation can amplify non-Gaussian effects, these discrepancies further motivate an explicit examination of the residual distributions. The practical implication is the same: while the estimator is operating as intended (mean statistics are close to theoretical values), the covariance mismatch weakens any claim of full statistical functionality without additional diagnostics. The following chapter therefore contains targeted Monte Carlo analyses to determine whether residuals are non-Gaussian and, if necessary, to recommend model or algorithmic adjustments.

3.4 Summary and discussion

This chapter has presented the 2D tracking problem used to evaluate the factor-graph estimator and the baseline configuration for experiments. The main points are:

- The state is $\mathbf{x} = [x, y, v_x, v_y]^\top$. Both systems use Cartesian position measurements $\mathbf{z}_k$ and a measurement covariance $\mathbf{R} = \sigma^2 \mathbf{I}_2$.
- The linear model is the constant-velocity (CV) formulation with discrete-time update $\mathbf{x}_k = \mathbf{F}\mathbf{x}_{k-1} + \mathbf{B}\mathbf{u}_k + \mathbf{v}_k$. The matrices $\mathbf{F}$ and $\mathbf{B}$ are the standard constant-velocity transition and input matrices, and process uncertainty is modelled by $\mathbf{Q}_{\text{CV}}(\Delta t; V)$.

- The nonlinear model is the constant-turn (CT) formulation. It shares the same state and measurement model; its transition is nonlinear, $\mathbf{x}_k = f(\mathbf{x}_{k-1}, \omega) + \mathbf{v}_k$, and the process covariance is obtained by rotating the CV covariance: $\mathbf{Q}_{\mathrm{CT}} = \mathbf{T}(\theta)\mathbf{Q}_{\mathrm{CV}}\mathbf{T}(\theta)^\top$.

- Process factors in the factor graph use residuals of the form $\mathbf{x}_k - (\mathbf{F}\mathbf{x}_{k-1} + \mathbf{B}\mathbf{u}_k)$ (or the nonlinear analogue) weighted by $\mathbf{Q}$. Measurement factors use residuals $\mathbf{z}_k - \mathbf{H}\mathbf{x}_k$ weighted by $\mathbf{R}$. These covariances control the relative influence of dynamics and observations in the optimisation.

- Ground-truth Monte Carlo tests ($N = 500$ runs) show that mean NIS/NEES values align closely with theoretical expectations for both linear and nonlinear systems. This indicates the estimator and the chosen covariance parameterisations behave as intended under the baseline conditions.

A notable caveat from the ground-truth experiments is that the empirical covariance of the χ^2-type statistics sometimes departs from the theoretical covariance by more than sampling variability would predict. This discrepancy suggests the possibility that one or more Gaussian modelling assumptions (in particular the Gaussianity of process or measurement residuals) may not hold exactly for the generated ensembles. While the estimator itself appears to function correctly (mean statistics are consistent with expectations), the covariance mismatch weakens any claim of full statistical functionality in the strictest sense.

To investigate this, the following chapter contains targeted Monte Carlo diagnostics to assess whether residuals are Gaussian. The experiments that follow (grid search and Bayesian optimisation over V and σ^2) therefore build on the working baseline established here and throughout the rest of the experiments.

Chapter 4

Experiments

This chapter builds on the ground truth validation presented in Chapter 3, where the system was shown to function as intended. The focus here shifts toward exploring parameter tuning and robustness. Starting with a simple blanket search to visualise the CNIS values surrounding the true values of V and σ^2. This is followed by applying BO to more efficiently identify the optimal V and σ values. Finally, stress-testing and refinement experiments, including hyper-parameter variation and adjustments to the factor graph setup, to further evaluate and improve the method.

4.0.1 Correcting Initial Assumptions

During the earlier studies in this section, it became apparent that the results were returning seemingly random V and σ^2 values. Regardless of adjustments made to the BO or grid search experiments, the outcomes varied drastically within each trajectory length. In several cases, the estimates were not only implausible but appeared effectively random. This was further evident in Chapter 3 where the covariance didn't line up with the theoretical covariance when using the ground truth runs. Closer inspection revealed that the number of Monte Carlo runs—although initially assumed sufficient—was likely too low to produce statistically stable results. This motivated the following Section 4.1, which tests whether the original choice of 500 runs was truly adequate.

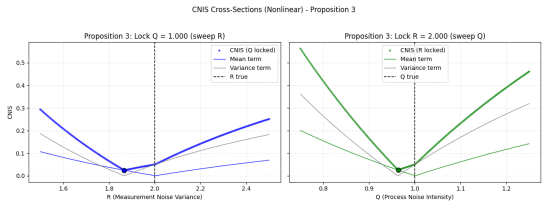
4.1 Monte Carlo Run Convergence Analysis

Monte Carlo (MC) evaluation underpins the consistency testing framework used throughout this dissertation. As with any stochastic method, the reliability of results depends critically on the number of independent trials N. If N is too small, the statistics may not approximate their true distributions, leading to misleading conclusions. This section investigates how N influences the stability of the CNIS measure, and related statistics, in order to establish a practical guideline for simulation experiments.

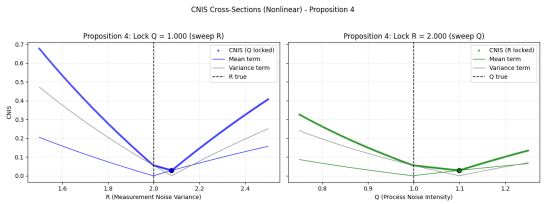
Motivation

In preliminary experiments, unusual behaviour was observed in the CNIS cross-sections for Propositions 3 and 4. Theory predicts that plotting CNIS against V and σ^2 (the scalar terms in the covariance matrix structures) should yield a clear "valley" structure. However, with $N = 500$ runs the cross-sections produced a wide, flat-bottomed 'U' shape (Fig. 4.1). Decomposing CNIS into its two logarithmic components, mean and variance, confirmed that instability in the separate terms

was responsible for this behaviour. Such results indicate that CNIS values from earlier experiments were not representative of their underlying χ^2 distributions. It was therefore hypothesised that increasing N would correct the behaviour, motivating the following convergence study.



Proposition 3, $M = 500$ (bad alignment).



Proposition 4, $M = 500$ (bad alignment).

Figure 4.1: CNIS cross-sections for Proposition 3 and 4 at insufficient $N = 500$ Monte Carlo runs.

4.1.1 Experimental Setup

To systematically study convergence, the number of Monte Carlo runs N was varied from 25 to $30,000$ in increments of 25. For each N, the following quantities were computed and stored:

- The log-mean term of NIS, $\left|\log\left(\frac{\tilde{\epsilon}_z}{n_z}\right)\right|$
- The log-variance term of NIS, $\left|\log\left(\frac{\tilde{S}_z}{2n_z}\right)\right|$
- The CNIS metric from equation 2.22

This approach makes it possible to directly observe how the statistics stabilise at the ground-truth values of V and σ^2 from the data generation component in section 3. While such reference trajectories are not available in real-world deployments, they can be accessed in simulation, enabling validation of convergence behaviour against known distributions. The aim here is not to prescribe a universal N for practice, but rather to determine how many runs are required in simulation for stable, interpretable results.

4.1.2 Data Generation

4.1.3 Results for $T = 100$

Using a trajectory length of $T = 100$, a dataset of $30,000$ Monte Carlo runs was generated. The script iteratively evaluated subsets, starting with the first 25 runs and increasing in steps of 25, each time using the ground-truth values $V = 1.0$ and $\sigma^2 = 2.0$.

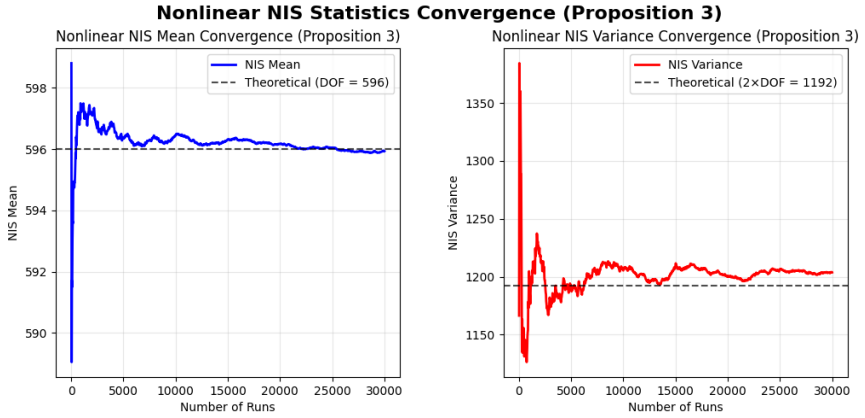


Figure 4.2: Proposition 3: mean/variance behaviour of NIS, $T = 100$.

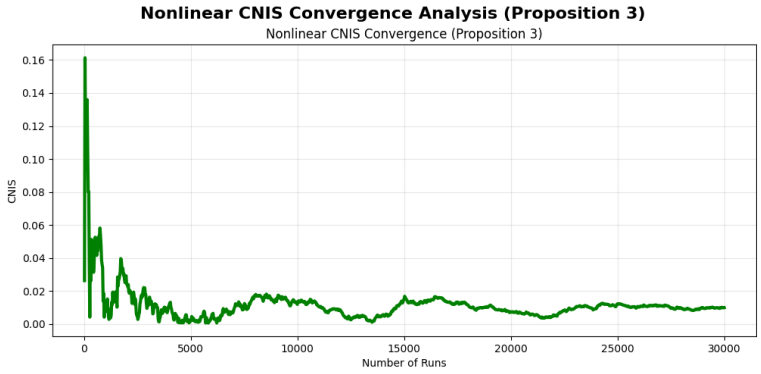


Figure 4.3: Proposition 3: CNIS convergence, $T = 100$.

Figure 4.2 shows the two log terms across iterations. It is clear that the original choice of $N = 500$ was insufficient: with too few runs, the mean and variance values vary widely between number of Monte Carlo runs. Focusing on the CNIS measure, which combines these terms, highlights how strongly under-sampling distorts the results.

Both graphs begin to plateau around $N = 5000$ runs, though they do not stabilise fully until approximately $N = 25,000$. Since increasing N greatly extends runtime, a compromise of $N \geq 5000$ appears sufficient for practical data collection.

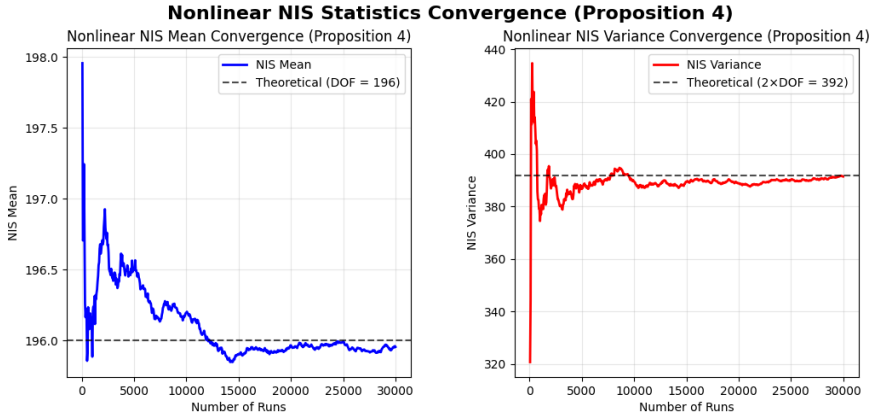


Figure 4.4: Proposition 4: mean/variance behaviour of NIS, $T = 100$.

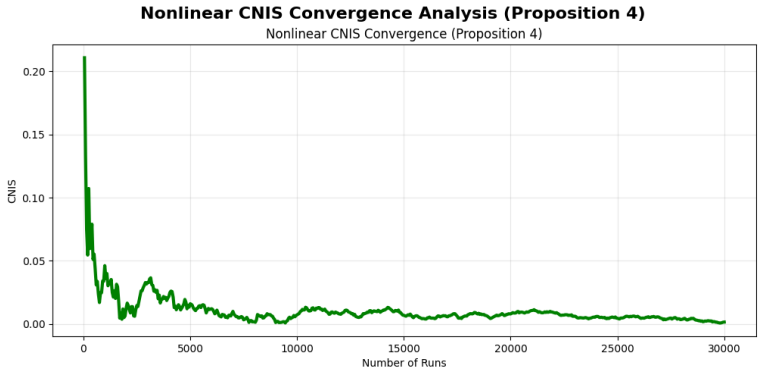
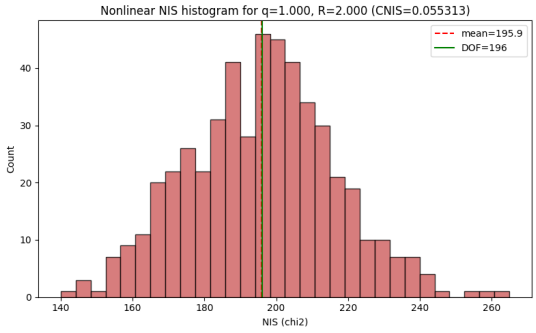


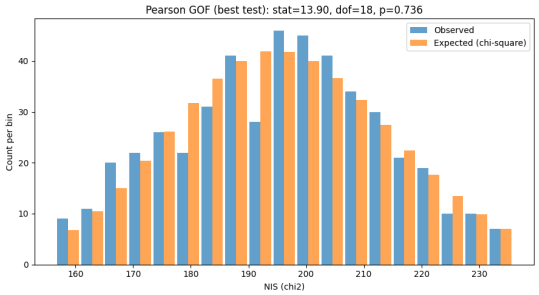
Figure 4.5: Proposition 4: CNIS convergence, $T = 100$.

For Proposition 4 the behaviour is more erratic. Although the relative changes mirror Proposition 3, the smaller degrees of freedom lead to larger relative fluctuations. Here, convergence is delayed until closer to $N \approx 10000$, and even after plateauing, the CNIS curve exhibits noticeable oscillations. Overall, Proposition 4 requires a larger number of runs for stable results.

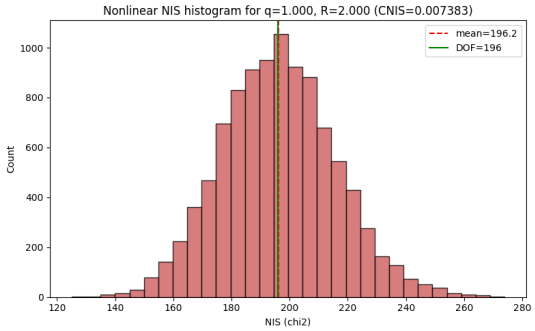
To further confirm these observations, the NIS noise was mapped as a probability density function for $N = 500$ and $N = 10000$.



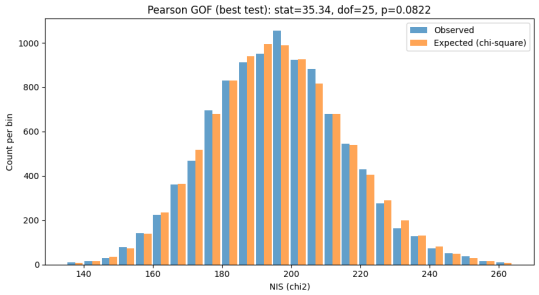
NIS distribution, $M = 500$.



Pearson test results, $M = 500$.



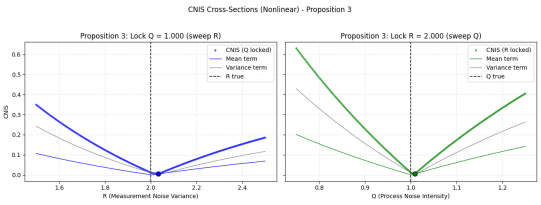
NIS distribution, $M = 10\,000$.



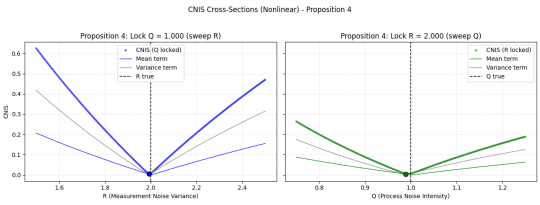
Pearson test results, $M = 10\,000$.

Figure 4.6: Empirical Probability Density Function of NIS and Pearson test overlays for insufficient ($M = 500$) vs sufficient ($M = 10\,000$) Monte Carlo runs.

The NIS density functions in Fig. 4.6 reinforce the findings. With $N = 500$, the distribution deviates significantly from the theoretical χ^2 reference, and the Pearson test rejects consistency, reporting a coefficient of 0.736. By contrast, with $N = 10,000$, the histogram closely matches the Gaussian χ^2 curve and the Pearson test confirms consistency, with a coefficient of 0.0822. This demonstrates the risks of under-sampling.



Proposition 3, $M = 10\,000$ (good alignment).



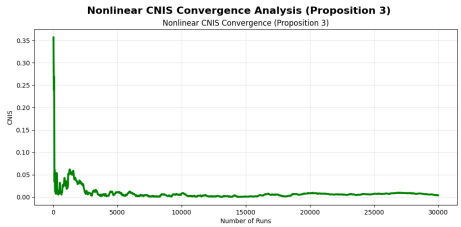
Proposition 4, $M = 10\,000$ (good alignment).

Figure 4.7: CNIS cross-sections for Proposition 3 and 4 at sufficient $N = 10,000$ Monte Carlo runs.

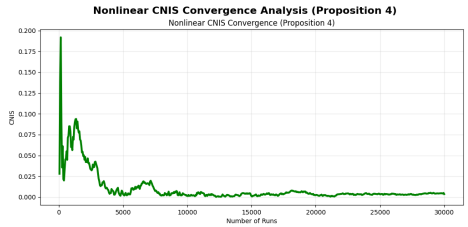
With the increased sample size, the cross-sections now exhibit the expected 'V' shape rather than the flat-bottomed 'U'. This sharper valley structure should improve the performance of later BO experiments by providing a clearer optimisation surface.

Effect of Shorter Trajectories

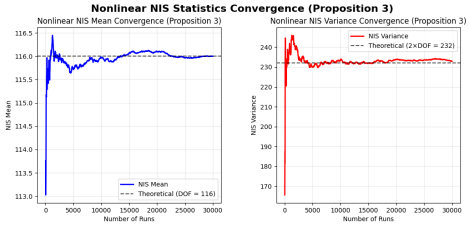
To test whether trajectory length affects the required N, experiments were repeated with shorter trajectories ($T = 20$).



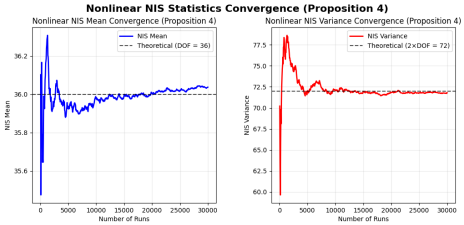
Proposition 3 convergence, $T = 20$.



Proposition 4 convergence, $T = 20$.



Proposition 3 mean/variance behaviour, $T = 20$.



Proposition 4 mean/variance behaviour, $T = 20$.

Figure 4.8: Convergence of CNIS, mean NIS, and variance of NIS for shorter trajectories ($T = 20$).

As shown in Fig. 4.8, convergence occurs somewhat earlier: around $N \approx 3000$ for Proposition 3 and $N \approx 8000$ for Proposition 4. However, the estimates are noisier overall compared with the $T = 100$ case. This is expected—fewer degrees of freedom increase the relative change of the NIS, making the estimates less stable.

Summary

This analysis shows that the original choice of $N = 500$ was inadequate for reliable CNIS evaluation. Stable convergence typically requires on the order of $N = 5000$ for Proposition 3 and $N = 10,000$ for Proposition 4, depending on trajectory length. Although computationally expensive, this is justified in simulation studies where accurate consistency validation is essential. For the remainder of this dissertation, $N = 10,000$ was used for all experiments to ensure consistency across Propositions 3 and 4. These findings provide a solid foundation for the Bayesian optimisation stage, ensuring that parameter tuning is based on trustworthy statistical evidence.

4.2 Experiment 1: Dense Cost Landscape in Q, R

The first experiment is designed to map the entire cost landscape of the CNIS metric as a function of the process noise covariance scaling term, V, and the measurement noise covariance scaling term, σ^2. This "blanket search" provides a detailed visualisation of the problem space, serving as a ground truth against which more sophisticated optimisation techniques can later be assessed. By characterising the topology of this surface, its smoothness, convexity, and location of minima, we can better interpret the behaviour of the tracking system and the performance of subsequent optimisation algorithms.

4.2.1 Setup

The experimental procedure consists of an exhaustive evaluation over a two-dimensional grid of hyper-parameters. The grid spans process noise intensity V and measurement noise variance σ^2, with a dense resolution of 50×50 points, yielding 2,500 parameter pairs in total. This resolution

was chosen to balance computational cost against the structure of the cost surface. The ranges for V and σ^2 were selected broadly enough to contain the expected optima, guided by the ground truth parameters used in data generation. In a practical study, the initial values of the noise covariances should be estimated prior to this step as a starting range. Given the true parameters, the ranges are $V = (0.01, 2.0)$ and $\sigma^2 = (0.01, 4.0)$. These ranges should provide sufficient rough estimates to capture the area around the true tuning paramters.

For each (V, σ^2) pair, the following steps are carried out:

1. A factor graph model is configured with process noise covariance Q and measurement noise covariance R.
2. The model is evaluated against a dataset of 10,000 pre-generated Monte Carlo trajectories, with runs executed in parallel to maintain computational feasibility.
3. For each full set of trajectories, the CNIS value is computed. Both Proposition 3 and Proposition 4 are considered in different plot instances, providing consistency checks under ideal and non-ideal conditions.
4. From the resulting distribution of 2,500 CNIS values, the minimum is identified, giving an approximate estimate of the corresponding V and σ^2.

Throughout this experiment a constant time step of $dt = 1.0$ is used, ensuring that the Q matrix, which depends on the time interval, remains consistent across all factors and runs. This controlled setup provides a clean baseline for analysing how the static noise parameters (V, σ^2) shape the CNIS landscape. The final output is a 2D matrix of CNIS values, visualised as a heatmap.

4.2.2 Results and Discussion

4.2.2.1 Linear Model.

The dense grid search for the linear constant-velocity model reveals a diagonal valley structure in the (V, σ^2) plane.

Proposition 3. The optimum occurs at $(V, \sigma^2) = (0.980102, 2.12292)$ with CNIS $= 0.00615216$, close to the ground truth $(1.0, 2.0)$.

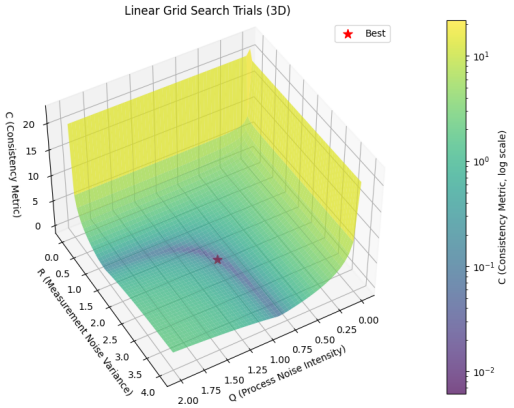


Figure 4.9: Linear model, Proposition 3: 3D cost surface using a log scale key for better viewing of the valley.

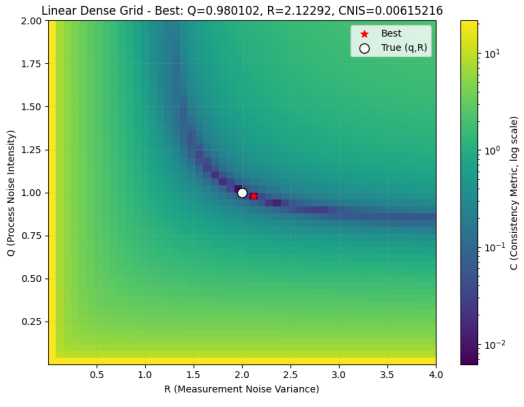


Figure 4.10: Linear model, Proposition 3: 2D heatmap using a log scale key for better viewing of the valley.

The surface forms a smooth curved valley with "tails" roughly aligned with the true V and σ^2 values, converging towards a minimum close to ground truth. In log space, colour scaling exaggerates gradients, but in absolute terms the region around the optimum is relatively flat with even the tails differing only slightly from the centre. Based on the shape, σ^2 is more sensitive to CNIS variation than V. The tuning parameters are not located exactly at the apex of the curve but are skewed towards the V axis. This means small movements along the valley ridge affect σ^2 more strongly than V. The parameters identified by the minimum CNIS should therefore be viewed as rough estimates of the true optimum. The limited resolution of the grid is partly responsible. This ridge requires finer sampling to pinpoint the true values accurately.

Proposition 4. Here the optimum shifts to $(V, \sigma^2) = (1.18408, 1.87808)$ with CNIS $= 0.00639314$.

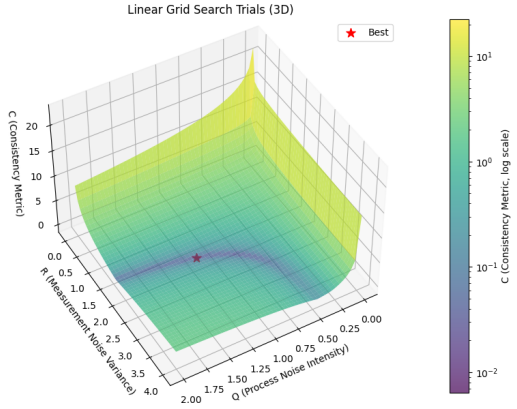


Figure 4.11: Linear model, Proposition 4: 3D cost surface using a log scale key for better viewing of the valley

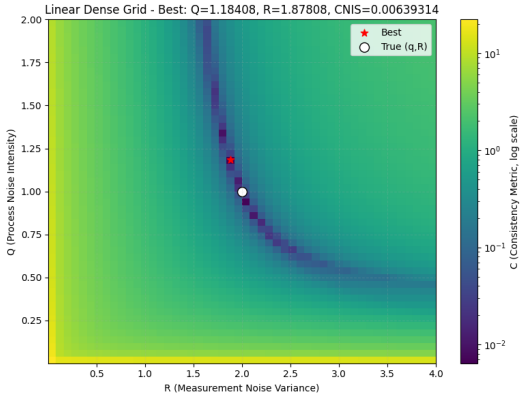


Figure 4.12: Linear model, Proposition 4: 2D heatmap using a log scale key for better viewing of the valley

In the case of proposition 4 the valley is longer and flatter, meaning CNIS varies little along the ridge. This elongation reduces the likelihood of identifying the exact tuning parameters. Unlike Proposition 3, sensitivity is reversed: small movements along the ridge affect V more strongly than σ^2. The flatness again implies that the reported optimum is only approximate, since nearby points produce equally good scores.

4.2.2.2 Nonlinear Model.

The nonlinear constant-turn system ($\omega = 0.1\,\text{rad/s}$) produces qualitatively similar diagonal valleys in (V, σ^2).

Proposition 3. The minimum occurs at $(V, \sigma^2) = (0.980102, 2.12292)$ with CNIS $= 0.00615216$.

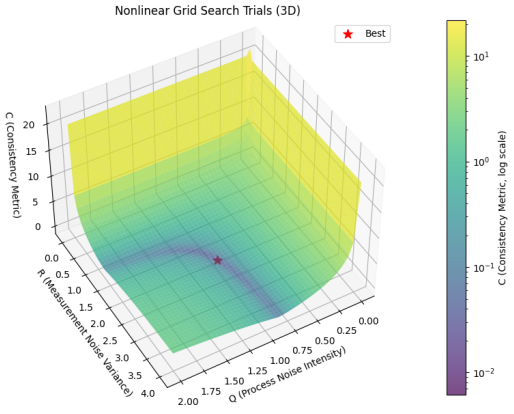


Figure 4.13: Non Linear model, Proposition 3: 3D cost surface using a log scale key for better viewing of the valley

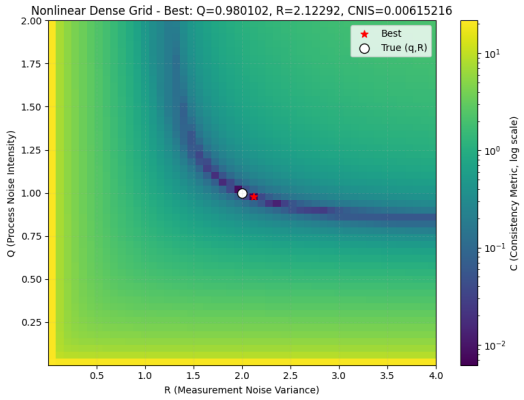


Figure 4.14: Non Linear model, Proposition 3: 2D heatmap using a log scale key for better viewing of the valley

As in the linear case, the valley converges near the true parameters. The flatness of the optimum shows robustness to small mis-specifications in V or σ^2. Notably, the linear and nonlinear models yield identical results for Proposition 3, since the noise injected into each trajectory is the same and the ideal initialisation minimises differences.

Proposition 4. The optimum is $(V, \sigma^2) = (0.939306, 2.04131)$ with CNIS $= 0.683499$.

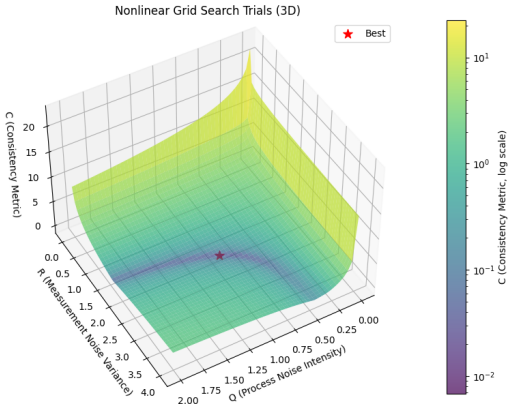


Figure 4.15: Non Linear model, Proposition 4: 3D cost surface using a log scale key for better viewing of the valley

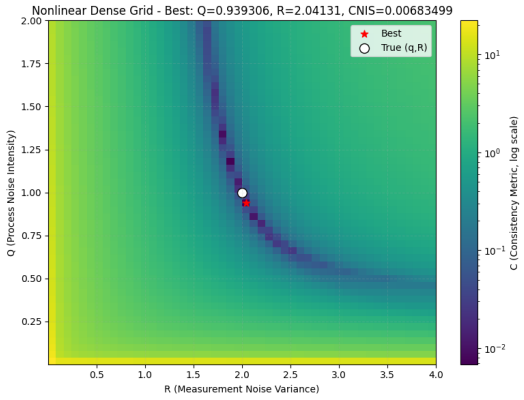


Figure 4.16: Non Linear model, Proposition 4: 2D heatmap using a log scale key for better viewing of the valley

Here the valley is longer and even flatter than in Proposition 3, with many (V, σ^2) pairs producing almost identical CNIS scores. Unlike Proposition 3, the linear and nonlinear results are not identical: although the same noise seeds are used, the nonlinear dynamics introduce small differences during the optimisation step, leading to slightly different optima.

4.2.2.3 General Observations.

Across both models and propositions, the CNIS landscapes consistently form diagonal valleys, reflecting the trade-off between V and σ^2. Increasing V (process noise) can be offset by reducing σ^2 (measurement noise), and vice versa, since both control effective uncertainty in the system.

Along the ridge, the system achieves a similar balance between process noise and measurement noise, producing near-constant CNIS values with only a gradual dip towards the true parameters.

The difference between Propositions 3 and 4 is primarily due to initialisation. In Proposition 3, sensitivity lies more in σ^2, which makes sense given the ideal start from $(0, 0)$: measurement noise dominates since no initial process noise is present. In Proposition 4, with random initialisation, the system relies more heavily on process noise, making V more influential, aligning with the statements made in section 2.4.4.

4.2.2.4 Summary and Motivation for Bayesian Optimisation.

In summary, the grid search successfully reveals the valley structure of the CNIS surface but is computationally expensive and limited in resolution. The flat ridges, especially in Proposition 4, mean that small amounts of numerical noise or grid spacing can shift the reported optimum, even though the underlying surface is genuinely flat. This explains the slight displacements of optima observed across runs.

These findings motivate the use of Bayesian Optimisation in the following section. By adaptively concentrating evaluations in promising regions, Bayesian Optimisation can achieve finer resolution along the ridge without exhaustively sampling the entire space. This both reduces computational cost and improves our ability to distinguish subtle local minima, while also clarifying the identifiability of (V, σ^2).

4.3 Experiment 2: BO parameter search of V, σ^2

Following the previous section, experiment two aims to counter the drawbacks of an exhaustive search by striking a balance between detail and computational cost. Previous Sections tested on both the linear and non-linear setup, however, as it has been proven both work and produce simmilar results, time will only be spent on the non-linear setup for the parameter exploration. Firstly, testing the acquisition functions EI and LCB are the first priority. The goal of this short first half of the experiment is to determine which results leads to a better CNIS result and to assess the general shape of the search.

4.3.1 Setup

- **Search space:** (V, σ^2) optimised in linear space with bounds matched to the grid experiment for comparability.
- **Objective:** CNIS calculated across 10000 Monte Carlo trajectories. Each candidate is mapped to $(\mathbf{Q}, \mathbf{R})$, the factor graph is solved, and CNIS is computed under the selected proposition (either Proposition 3 or Proposition 4, as configured).
- **Acquisition function:** EI and LCB separately tested in order to determine best likely acquisition function. Both functions also have a force jump parameter of 10 so that they would leave local minima if stuck for too long, reducing chance of incorrect minima.
- **Surrogate model:** Gaussian Process with Matérn-5/2 ARD kernel; no output normalization.
- **Evaluation budget:** $n_{\text{init}} = 100$ initial samples and $n_{\text{BO}} = 750$ BO iterations (approximately 850 total evaluations).

- **Data Input:** trajectories consistiting of the base hyper parameters, $\Delta t = 1.0$, set seed to the same as previous sections, $V = 1.0$, $\sigma^2 = 2.0$, turn rate $\omega = 0.1 rad/s$.
- **Outputs:** Report best (V, σ^2, C_{NIS}), and runtime.
- **Repetitions:** Repeated 3 times, tabulating all results and plotting the run with the lowest CNIS value.

Implementation (Pseudocode). A high-level pseudocode of the BO implementation is given below, describing the outputs of the script for the non linear system.

Algorithm 1: Bayesian Optimization for Tuning (V, σ)

Input: Initial parameters (V, σ), number of iterations N
Output: Optimized $(V^\star, \sigma^\star)$
Initialize Gaussian Process surrogate $\mathcal{GP}$;
for $i = 1$ **to** N **do**
    Select (V_i, σ_i) using acquisition function α;
    Evaluate CNIS at (V_i, σ_i) with Monte Carlo sampling;
    Update $\mathcal{GP}$ with $(V_i, \sigma_i, \text{CNIS})$;
end
return best $(V^\star, \sigma^\star)$ found;

4.3.2 Comparison of EI and LCB Acquisition Functions

Both functions were applied to Proposition 3 and Proposition 4 using trajectories of length $T = 100$. Each setup was repeated three times to check for robustness. The following results combine tabulated statistics with visualisations of the optimisation surfaces.

4.3.2.1 Proposition 3: ideal case

Table 4.1 reports the EI results for Proposition 3, while Figure 4.17 shows the corresponding optimisation surface.

Table 4.1: EI results for Proposition 3 ($T = 100$).

Run	V	σ^2	CNIS	Mean NIS	Var NIS	Time [s]
1	1.00617	2.00855	0.0049321	593.160	1192.186	613
2	1.00049	2.03066	0.0051505	593.226	1192.579	608
3	1.01434	1.97659	0.0050835	593.212	1192.471	619
GT	1.00000	2.00000	0.0119351	596.441	1205.419	–

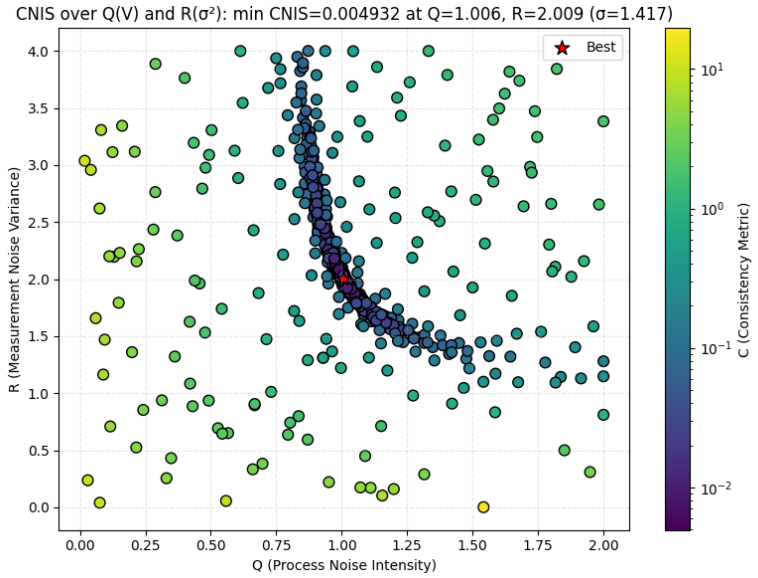


Figure 4.17: The EI optimisation surface using BO to find the best value of CNIS under proposition 3 with a trajectory length of $T = 100$.

The EI search traces the main ridge of the optimisation valley but also steps up the walls in a few places. This gives a good mix of refinement and wider coverage, with CNIS values as low as 0.004932, with V and σ^2 values extremely close to the ground truth. The broader spread comes from the force jump mechanic, which stops the optimiser from sitting in one part of the valley. EI naturally complements this approach because it assigns high acquisition values to regions where uncertainty is large and the mean prediction suggests potential improvement, creating a principled exploration-exploitation trade-off that prevents premature convergence.

Table 4.2 and Figure 4.18 show the results under LCB.

Table 4.2: LCB results for Proposition 3 ($T = 100$).

Run	V	σ^2	CNIS	Mean NIS	Var NIS	Time [s]
1	0.99520	2.05380	0.0051826	593.110	1192.383	621
2	1.03435	1.90924	0.0055071	592.799	1192.145	609
3	0.99687	2.04646	0.0051512	593.142	1192.410	609
GT	1.00000	2.00000	0.0119351	596.441	1205.419	–

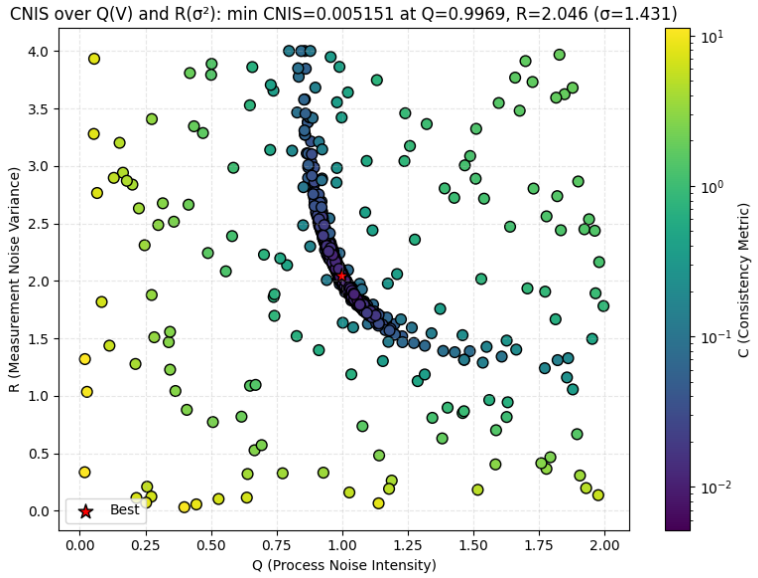


Figure 4.18: The LCB optimisation surface using BO to find the best value of CNIS under proposition 3 with a trajectory length of $T = 100$.

LCB, on the other hand, stays tighter along the ridge and the search path looks shorter. This is because LCB mostly favours points with low predicted mean, so on a flat valley it keeps resampling in the same region. Unless κ is pushed higher, the added uncertainty at the walls is not enough to pull it away. The force jump has less effect here too, since most jump points score worse than the valley floor under LCB. The minimum CNIS of LCB is 0.005151, slightly higher than EI but practically the same. The main difference lies in the recovered parameters. The V value is still close to ground truth, but the σ^2 estimate drifts slightly further than in the EI case. This reflects the sensitivity of the valley ridge, where even small shifts in V can lead to larger variations in σ^2.

Overall, both methods work well in this ideal case, but EI spreads further and gives slightly better coverage of the landscape. Now lets see if proposition 4 shows varied results.

4.3.2.2 Proposition 4: Misaligned case

For Proposition 4, both EI and LCB show degraded behaviour. Tables 4.3 and 4.4, together with Figures 4.19 and 4.20, summarise the results.

Table 4.3: EI results for Proposition 4 ($T = 100$).

Run	V	σ^2	CNIS	Mean NIS	Var NIS	Time [s]
1	1.12991	1.90284	0.0013981	196.237	391.926	10645
2	1.14312	1.89494	0.0009512	196.176	392.022	10668
3	0.82187	2.18042	0.0006435	195.980	391.788	10614
GT	1.00000	2.00000	0.0073827	196.196	389.505	–

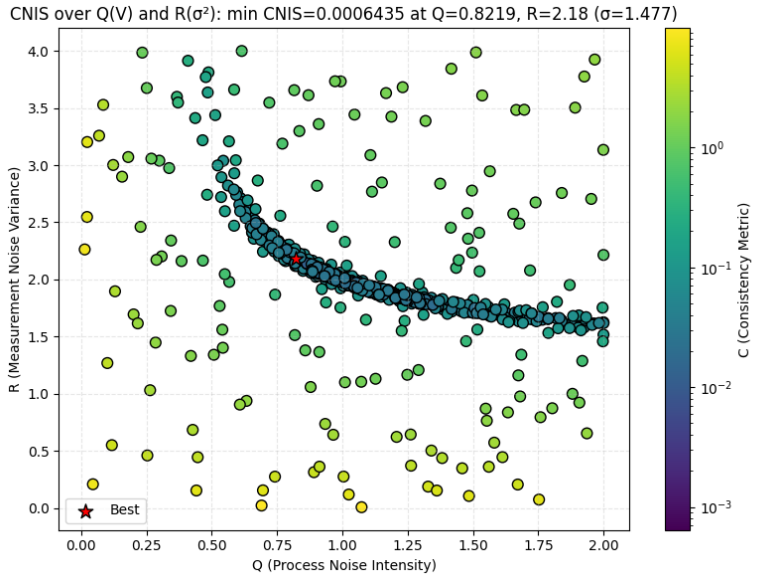


Figure 4.19: The EI optimisation surface using BO to find the best value of CNIS under proposition 4 with a trajectory length of $T = 100$.

Similar to Proposition 3, EI explores the ridge in detail but also climbs the "walls" of the valley in places. The overall search pattern behaves the same, but with the pattern of proposition 4. The minimum CNIS drops much lower here, down to 0.0006435, but the recovered V and σ^2 are noticeably further from the ground truth. The reasons behind this will be discussed later in section 4.3.2.3, but for now the focus is just on the shapes of the surfaces.

Table 4.4: LCB results for Proposition 4 ($T = 100$).

Run	V	σ^2	CNIS	Mean NIS	Var NIS	Time [s]
1	1.16414	1.88283	0.0007651	196.068	392.165	8581
2	0.82742	2.17158	0.0006776	196.116	392.033	8608
3	0.79728	2.21600	0.0017812	195.661	391.980	8594
GT	1.00000	2.00000	0.0073827	196.196	389.505	–

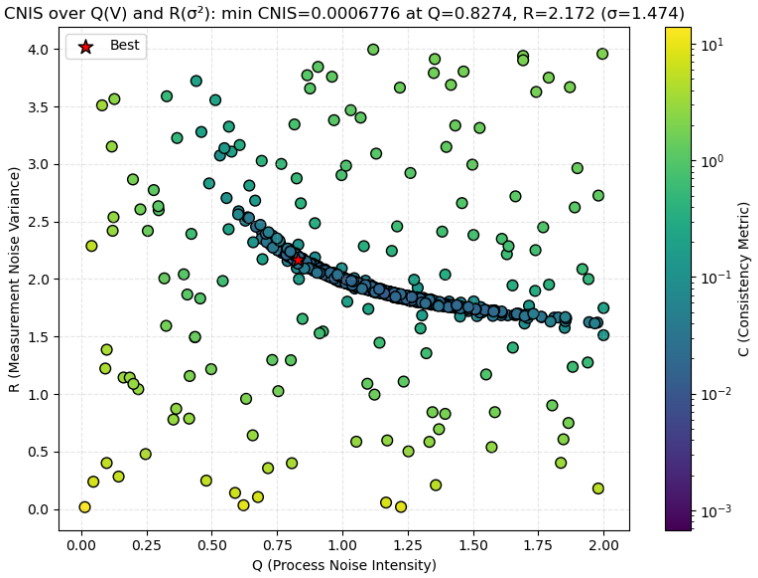


Figure 4.20: The LCB optimisation surface using BO to find the best value of CNIS under proposition 4 with a trajectory length of $T = 100$.

In both EI and LCB, the CNIS values stay very stable across runs, around 10^{-2} to 10^{-3}, even though the recovered V and σ^2 vary a lot. This suggests the mismatch isn't down to the acquisition function itself, but something else driving the gap between the estimated and true parameters. The overall shape of the LCB function is simmilar to that of the proposition 3 run, where the ridge has more definition but it fails to find the best minimum between the two acquisition function.

Overall, both methods capture the general landscape, but EI edges out LCB. It returns slightly lower CNIS values and shows a bit more exploration which is vital when searching a flat valley like structure. For that reason, EI will be used as the default in the experiments to come.

4.3.2.3 Anomalous behaviour at $T = 100$

A striking feature across both acquisition functions is the anomalous behaviour observed for trajectories of length $T = 100$ in Proposition 4. The CNIS results were around an order of magnitude smaller than those obtained at the ground-truth parameters, with "optimal" (V, σ^2) estimates that deviate noticeably from $(1, 2)$. For example, under EI the optimiser located points near $(V, \sigma^2) = (0.82, 2.18)$ and $(1.14, 1.89)$ with CNIS values on the order of 10^{-4}, far below the baseline ground-truth CNIS of 7.4×10^{-3}.

To check whether this effect was just due to one unlucky trajectory, the experiment was repeated with a new random seed. The anomaly persisted, though the deviation was smaller. Table 4.5 shows one such re-run, where the optimiser again found (V, σ^2) values away from $(1, 2)$ that returned CNIS values between 1.5–3.9 $\times 10^{-4}$, still below the baseline of 1.7×10^{-3} for that seed.

Table 4.5: EI results for Proposition 4 with $T = 100$ under a different seed.

Run	V	σ^2	CNIS	Mean(χ^2)	Var(χ^2)	Time [s]
1	1.04119	1.96594	0.0001503	196.011	392.037	7260
2	0.95607	2.03654	0.0001595	196.020	392.022	7419
3	1.04876	1.96081	0.0003867	195.964	391.921	7388
GT	1.00000	2.00000	0.0016688	195.929	391.489	–

The persistence of this effect across seeds shows it is not just random initialisation. Instead, it suggests a structural property of the optimisation surface at $T = 100$. The cost landscape here forms a broad, flat ridge where many (V, σ^2) pairs that appear similar. Small fluctuations from finite-sample noise can then carve out shallow dips just off the ridge, which look like true minima. Bayesian optimisation latches onto these dips because they consistently report lower CNIS than the ground-truth point, even though they are not genuinely better solutions.

In summary:

- The anomaly is reproducible across seeds, not just a one-off effect.
- The optimiser finds off-truth (V, σ^2) with deceptively low CNIS values due to ridge flatness and surface noise.
- All recovered parameters remain near the ridge, confirming that the optimiser is tracking the valley but being pulled toward spurious dips.

This behaviour raises doubts about relying on $T = 100$ alone: the CNIS metric here can give artificially low scores at incorrect parameter values. A fuller trajectory-length study helps to put this anomaly in context and shows how varying T changes the stability of the optimisation surface.

4.4 Experiment 3: Trajectory-length Study

Building on the EI–LCB comparison at $T = 100$, Experiment 3 investigates how trajectory length T influences the optimisation surface and parameter recovery, in the hopes of improving overall results. The study focuses on the robustness of the CNIS objective, the mean and variance of NIS and recovered parameters (V, σ^2) for trajectory lengths $T \in \{20, 50, 100, 200\}$ under Proposition 3 and Proposition 4.

4.4.1 Setup

Motivation and objective. The objective is to characterise how trajectory length modifies the optimisation landscape (surface curvature and ridge flatness), the susceptibility to spurious noise-induced dips, and the identifiability of the process/measurement parameters. The working hypothesis is that some trajectory lengths produce broader, shallower minima, with a less defined point to the ridge, which reduce identifiability and increase sensitivity to Monte Carlo noise. It is also hypothesised that the two propositions will respond differently because they imply different effective degrees of freedom.

Design and controls. The experimental configuration is aligned with Experiment 2 to permit direct comparison:

- **System cases:** Proposition 3 (near-ideal) and Proposition 4 (misaligned).
- **Trajectory lengths:** $T \in \{20, 50, 100, 200\}$. Shorter trajectories are formed as prefixes of the $T = 200$ dataset (this correlation is noted as an experimental caveat).
- **Objective:** CNIS computed via the same Monte Carlo budget used in Experiment 2.
- **Bayesian optimisation:** Expected Improvement (EI) is used as the acquisition function; GP surrogate with Matérn-5/2 ARD kernel; BO budget and initialisation match Experiment 2.
- **Repetitions:** Three independent BO repetitions (with the same distinct seed) per (Proposition, T) pair. The best (lowest CNIS) run is reported along with the NIS mean and standard deviation across repetitions where informative.
- **Controls:** Factor graph structure, solver tolerances, Δt, and parallel execution settings are held constant.
- **Outputs:** recovered (V, σ^2), CNIS, mean/variance of NIS, runtime.

The important thing to note is that using prefixes of the whole trajectory helps with real world implementation, reducing the amount of data required to collect, giving a more realistic way of validating results.

Reproducibility notes. All seeds and runtime environment details are logged in the HDF5 outputs. Since shorter trajectories are prefixes of the $T = 200$ run, results at different T are not statistically independent. Still, the comparison is useful as it shows how trajectory length affects stability and parameter recovery.

4.4.1.1 Results

Proposition 3: Near-ideal case Table 4.6 shows the outcomes for Proposition 3 across the four tested trajectory lengths.

Table 4.6: EI results for Proposition 3 across trajectory lengths.

T	Run	V	σ^2	CNIS	Mean NIS	Var NIS
200	1	1.01189	1.99489	0.007260	1187.914	2393.140
	2	1.00943	2.00598	0.007047	1187.612	2392.021
	3	1.01294	1.99145	0.007149	1187.791	2392.626
	GT	1.00000	2.00000	0.014759	1196.248	2427.062
100	1	1.00617	2.00855	0.004932	593.160	1192.186
	2	1.00049	2.03066	0.005150	593.226	1192.579
	3	1.01434	1.97659	0.005083	593.212	1192.471
	GT	1.00000	2.00000	0.011935	596.441	1205.419
50	1	1.01179	2.00490	0.007787	293.717	591.974
	2	1.01483	1.99333	0.007791	293.717	591.972
	3	1.03302	1.92848	0.008278	293.690	592.263
	GT	1.00000	2.00000	0.017988	296.247	602.242
20	1	1.00121	2.01596	0.003965	115.551	232.019
	2	1.00040	2.01832	0.004130	115.566	232.088
	3	1.00399	2.00532	0.003906	115.551	231.993
	GT	1.00000	2.00000	0.007388	115.959	233.638

The three runs for each T are very consistent. Comparing BO CNIS results to ground truth CNIS gives ratios of about:

$$\mathrm{GT/CNIS}_{min} \approx \begin{cases} 1.89 & (T = 20), \\ 2.31 & (T = 50), \\ 2.42 & (T = 100), \\ 2.09 & (T = 200). \end{cases}$$

So BO finds parameters that cut CNIS by approximately a factor of two across all T. Errors in V and σ^2 are small (MAPEs $\approx$ 0.2–2.0%). Interestingly, $T = 20$ gives slightly better CNIS than longer trajectories, probably because short runs don't accumulate rare anomalies. Still, all CNIS values are in the same 10^{-3} range. This differs from the expectation that longer T increases the accuracy as the absolute difference in NIS mean and covariance stay the same but the relative difference decreases with the larger DoF. However the extra length also introduces more opportunities for errors, which seems to balance the CNIS results out between varied lengths of trajectory.

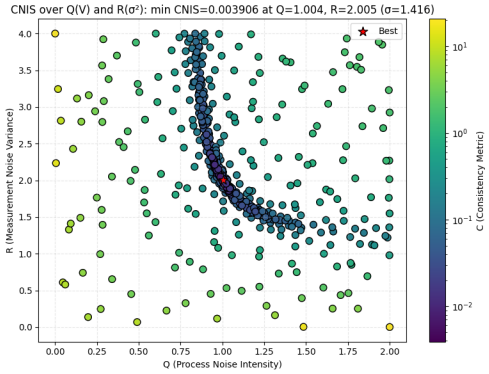


Figure 4.21: Proposition 3, $T = 20$

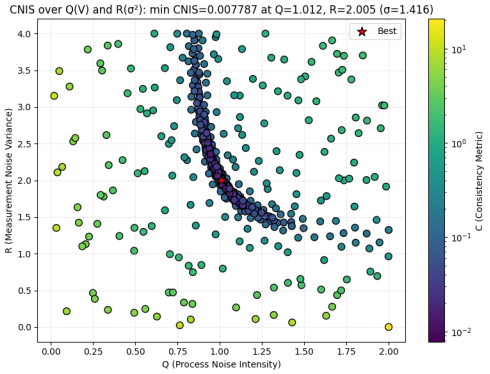


Figure 4.22: Proposition 3, $T = 50$

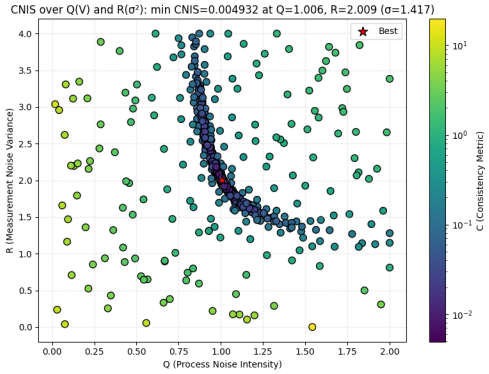


Figure 4.23: Proposition 3, $T = 100$

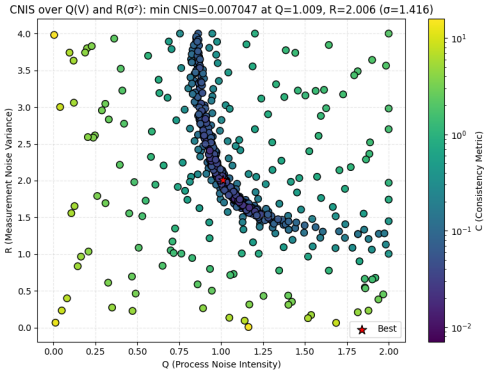


Figure 4.24: Proposition 3, $T = 200$

Figure 4.25: A summary of all the different trajectory legnth BO studies under proposition 3, ranging from $T = 20, 50, 100, 200$.

The surface plots (Fig. 4.25) also look very similar across T, suggesting trajectory length has little effect on the overall outcome, at least up to $T = 200$ for Prop 3.

Proposition 4: Misaligned case Table 4.7 gives the numbers for Prop 4.

Table 4.7: EI results for Proposition 4 across trajectory lengths.

T	Run	V	σ^2	CNIS	Mean NIS	Var NIS
200	1	1.00578	2.00108	0.002367	395.255	792.383
	2	1.00179	2.00484	0.002241	395.196	792.165
	3	1.00610	2.00153	0.002160	395.155	791.982
	GT	1.00000	2.00000	0.004866	396.079	795.705
100	1	1.12991	1.90284	0.001398	196.237	391.926
	2	1.14312	1.89494	0.000951	196.176	392.022
	3	0.82187	2.18042	0.000643	195.980	391.788
	GT	1.00000	2.00000	0.007383	196.196	389.505
50	1	0.99273	2.04302	0.012351	94.832	192.020
	2	0.97240	2.06087	0.012492	94.815	191.986
	3	1.02242	2.01922	0.012451	94.818	191.989
	GT	1.00000	2.00000	0.027024	96.076	197.103
20	1	1.03015	1.97876	0.001379	35.965	72.030
	2	1.04868	1.96592	0.001466	35.963	72.032
	3	1.00795	1.99548	0.001218	35.957	71.998
	GT	1.00000	2.00000	0.001717	35.975	72.074

Prop 4 is more sensitive to T. The GT/BO CNIS ratios are:

$$\text{GT/CNIS}_{min} \approx \begin{cases} 1.41 & (T = 20), \\ 2.19 & (T = 50), \\ 11.64 & (T = 100) \text{ (outlier)}, \\ 2.25 & (T = 200). \end{cases}$$

The $T = 100$ case is the clear outlier. BO drives CNIS very low but the recovered parameters are way off (MAPE(V) $\approx$ 15.0%, MAPE(σ^2) $\approx$ 6.4%). By contrast, $T = 200$ gives both low CNIS and very accurate parameters (MAPEs below 0.5%). So the $T = 100$ result is most likely a noise-induced dip that BO locked onto. This is a reminder that very small CNIS values aren't always trustworthy.

For the other lengths ($T = 20, 50, 200$), the ratios to GT are similar, but the absolute CNIS values differ more than in Prop 3. This makes sense: Prop 4 adds an optimisation step, which seems to amplify differences between trajectories. In other words, the extra flexibility in the system makes results more variable depending on T.

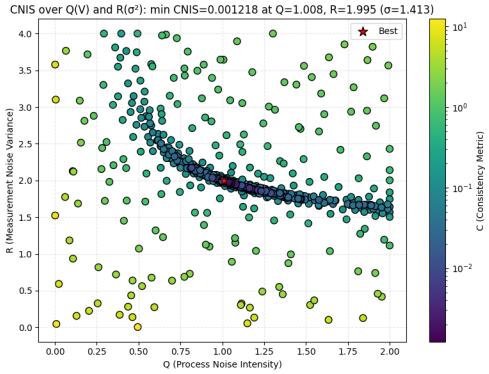


Figure 4.26: Proposition 4, $T = 20$

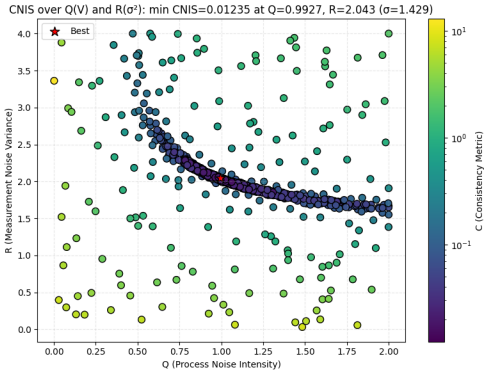


Figure 4.27: Proposition 4, $T = 50$

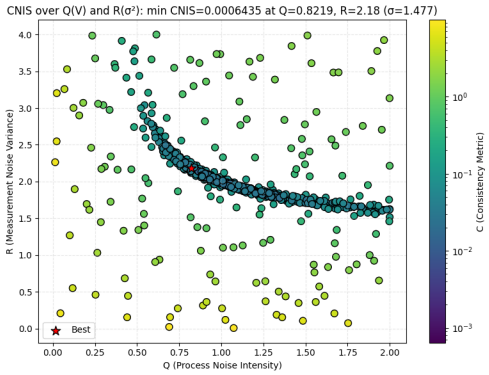


Figure 4.28: Proposition 4, $T = 100$

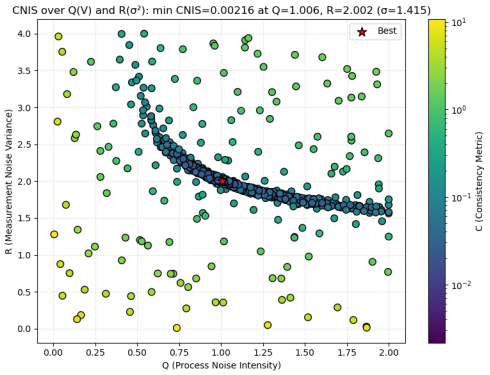


Figure 4.29: Proposition 4, $T = 200$

Figure 4.30: A summary of all the different trajectory legnth BO studies under proposition 4, ranging from $T = 20, 50, 100, 200$.

The surfaces in Fig. 4.30 all look similar in shape, so the $T = 100$ anomaly seems down to unlucky alignment in that specific prefix, not a fundamental difference in the landscape.

Cross-case synthesis and recommendations Two main takeaways:

1. **Prop 3 is stable:** Near-ideal init gives accurate, consistent recovery across all T (MAPEs small, GT/BO CNIS ratios ~ 2). Larger T averages out noise better.
2. **Prop 4 is sensitive:** Misaligned init increases sensitivity to the specific sequence and seed. The $T = 100$ anomaly shows BO can exploit a dip that doesn't generalise.

Practical points:

- Always repeat BO on multiple *independent* trajectories lengths (prefixes and independent runs) and combine results (e.g. median) to avoid overfitting to quirks.
- Mixing trajectory lengths is a cheap validation check when data is scarce.
- Treat concerningly low CNIS values with caution: check parameter errors if ground truth is known, or repeat runs if it isn't.

Overall parameter recovery Across all runs, the best (V, σ^2) estimates are consistently close to ground truth. For Proposition 3, recovery is very stable: all trajectory lengths cluster near $(1.01, 2.00)$, with the closest of all trajectories being $T = 20$ with run 3, and run 2 from $T = 200$ followed closely behind. For Proposition 4, the most reliable recovery occurs at $T = 200$, with both parameters within $< 0.5\%$ of the true values. In contrast, $T = 100$ produces misleading estimates despite an apparently good CNIS, showing how strongly trajectory length interacts with sequence-specific quirks. When runtime allows, $T = 200$ is the safest choice, though $T = 20$ or 50 still give reasonable estimates in well-initialised setups, but results should always be confirmed across multiple trajectories.

Runtime trade-offs Trajectory length also drives runtime. Shorter runs ($T = 20, 50$) finish quickly, which is convenient for rapid checks or when data is limited. Longer runs ($T = 200$) take more time but provide the most reliable results, especially for Proposition 4. The trade-off is clear: shorter trajectories are efficient but more sensitive to noise, whereas longer ones average out stochastic fluctuations at the cost of computational effort.

Design insight The appropriate trajectory length depends on the system and the setup. For the tested 2D tracking setup, shorter trajectories are generally fine for proposition 3, since parameter recovery is robust. For misaligned initialisation (Prop 4), longer trajectories are strongly recommended to avoid BO locking onto spurious dips. In practice, combining several shorter prefixes with one longer run offers a good compromise between efficiency and reliability.

Limitations and next steps Since shorter trajectories here are just prefixes of $T = 200$, some T (notably $T = 100$) may be biased by sequence-specific quirks. The next step is testing variations of Δt within a run, to see if time spacing itself shapes the optimisation surface.

Chapter 5

Extension: Improvement and Practical Performance

5.1 Extension 1: Variable Time steps

5.1.1 Setup

The previous experiments held the sampling interval Δt fixed throughout the trajectory, so that the process noise covariance $Q(\Delta t)$ in Eq. 3.3 and 3.10 was homogeneous across all steps. In this experiment, the aim is to deliberately vary Δt within the same trajectory, alternating between $\Delta t \in \{1, 5, 10\}$ under two mixture ratios: an even schedule (33:33:33) and a skewed schedule (60:30:10) dominated by small time steps. The trajectory length is fixed at $T = 50$, like the previous test, however $T = 100$ will also be tested to explore the effects on outlier data. Aside from this change in step scheduling, the Bayesian optimisation procedure, objectives, and evaluation metrics follow the same setup as before, ensuring results are directly comparable. It is important to note that the length of the trajectory refers directly to the number of measurements taken, not the length of the states. Figure 5.1.1 shows the number of measurements misaligning with the states once Δt is increased

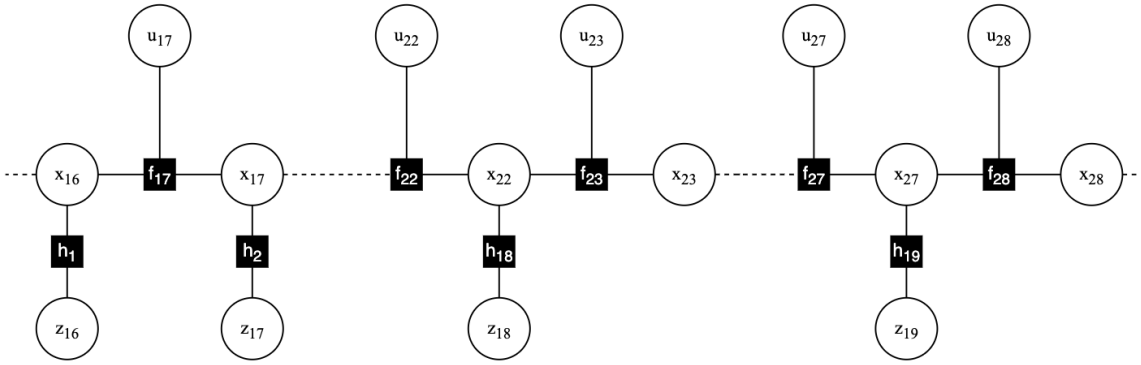


Figure 5.1: Example Markov Chain of the changing structure for the factor graph. This case uses the 33:33:33 ratio alongside $T = 50$, showing how measurements change

The motivation is to investigate how heterogeneous step sizes reshape the optimisation land-

scape and affect identifiability of the process and measurement parameters. Since $Q(\Delta t)$ grows polynomially with Δt(cubic in position variance, quadratic in cross-terms, and linear in velocity variance)longer steps magnify the effect of process noise, while shorter steps impose tighter constraints. When these are combined, some portions of the factor graph become strongly informative while others are diffuse, producing an optimisation surface that is no longer a broad, flat valley but instead contracts and steepens around certain directions in (V, σ^2). Put differently, large Δt segments act as leverage points, exaggerating the penalty for mis-specified V, whereas small Δt segments stabilise the trajectory and reduce Monte Carlo variance in CNIS/NIS. The two selected ratios of the Δt schedules should show different trade-offs: the even ratio balances these influences, while the skewed ratio concentrates on local stability with occasional large steps to enforce global consistency.

This part of the experiment is motivated by general control-theory considerations around irregular sampling. In practice, unevenly spaced measurements can make it harder to observe the full state, which in turn can complicate parameter estimation. On the other hand, having more frequent observations tends to reduce the variance of estimated parameters, simply because there's more information to work with [36, 9]. Thinking about it in the context of our mixed-Δt trajectories, this gives us a way to explore how the balance of short and long intervals might affect how well V and σ^2 can be recovered within the Bayesian optimisation setup. From this perspective, we would expect that an even mix of Δt values might maintain the ridge-like structures we saw in earlier experiments, while a skewed mix could sharpen the minima and make parameter recovery slightly more precise, though perhaps more sensitive to noise.

All runs are made similarly to that in section 4.3. The only variation in terms of the BO and grid search methods are the changing Δt structure. The same Δt schedule is applied consistently across Proposition 3 (near-ideal initialisation) and Proposition 4 (misaligned initialisation) cases, so that differences in outcome can be attributed to the interaction between initialisation and Δt heterogeneity.

5.1.2 Results under Varying ΔT Ratios

Having established the baseline trajectory-length results, the next step is to test the effect of mixing different ΔT values within a single trajectory. A trajectory length of $T = 50$ is adopted as the main case, since it strikes a balance between computational efficiency and the ability to refine results. In the previous section, $T = 50$ was shown to be the least well-performing of the otherwise "good" configurations, making it a natural candidate for improvement. Two mixing schedules are considered: an even 33:33:33 ratio of short/medium/long ΔT, and a skewed 60:30:10 ratio. These are compared against the constant-ΔT baseline. The focus is again on CNIS, NIS statistics, and parameter recovery. The representation of the shapes shall be grid search plots as they do a better job of showing the overall shape. BO experiments will be run alongside these but only tabulated data will be shown.

5.1.3 Proposition 3: Invariance under ΔT

For Proposition 3, the introduction of mixed ΔT values produces no meaningful change compared with the constant-ΔT baseline. Figure 5.2 shows the grid search heat maps for both the 33:33:33 and 60:30:10 schedules, alongside the constant-ΔT case. All surfaces remain virtually identical, confirming that the optimisation landscape is unaffected. This invariance arises because the model and filter are perfectly matched: the discretisation terms $F(\Delta T)$ and $Q(\Delta T)$ are both derived

from the same continuous-time dynamics. As a result, the innovations remain properly whitened, and the NIS expectation is governed only by measurement dimension.

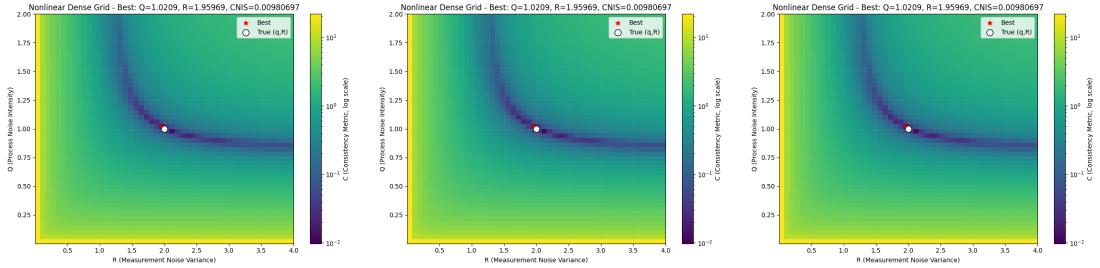


Figure 5.2: Grid search landscapes for Proposition 3, $T = 50$: constant ΔT, 33:33:33, and 60:30:10.

Focussing on the true results now there are two things to take note of. Firstly they are very similar to that of the constant Δt method in table 4.6, with only minor changes for both the schedule ratios due to the randomness of the BO algorithm.

Table 5.1: Proposition 3, 33:33:33 ratio, $T = 50$.

Run	V	σ^2	C	NIS Mean	NIS Var	Orig q	Orig R
1	0.99425	2.07605	0.008416	293.71	592.39	1	2
2	1.01522	1.99035	0.007955	293.79	592.28	1	2
3	1.00186	2.04361	0.008054	293.75	592.24	1	2
GT	1	2	0.01799	296.25	602.24	1	2

Table 5.2: Proposition 3, 60:30:10 ratio, $T = 50$.

Run	V	σ^2	C	NIS Mean	NIS Var	Orig q	Orig R
1	1.01312	1.99998	0.007877	293.71	591.94	1	2
2	1.00045	2.05019	0.008000	293.71	592.14	1	2
3	0.99837	2.06005	0.008014	293.65	591.97	1	2
GT	1	2	0.01799	296.25	602.24	1	2

5.1.4 Proposition 4: Amplification and Shifts

In contrast, Proposition 4 shows much stronger changes when ΔT is mixed. At $T = 50$, the optimisation surface becomes noticeably sharper and more amplified compared with the constant-ΔT case.

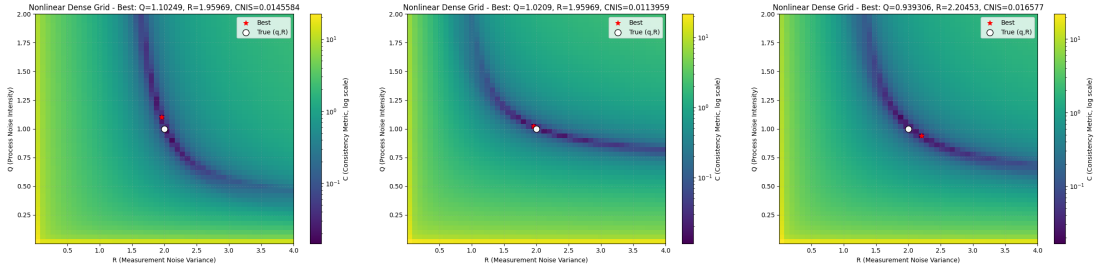


Figure 5.3: Grid search landscapes for Proposition 4, $T = 50$: constant ΔT, 33:33:33, and 60:30:10.

The shapes of the graphs vary drastically with variability in Δt. The clearest change appears in the even ratio 33:33:33, which shifts the surface to look more like the Proposition 3 case: sensitivity in V is reduced, while shifts in σ^2 dominate the structure. For the 60:30:10 split, the graph no longer clearly favours either parameter. Instead, it creates a more balanced trade-off between V and σ^2, with both showing similar sensitivities. On top of this, the overall valley bottom shrinks, reflecting the underlying shift in shape.

These large shifts lead to very different experimental outcomes. Tables 5.3–5.4 show the numerical results.

Table 5.3: Proposition 4, 33:33:33 ratio, $T = 50$.

Run	V	σ^2	C	NIS Mean	NIS Var	Orig q	Orig R
1	1.00339	2.04962	0.009367	95.21	192.21	1	2
2	1.00824	2.03226	0.008899	95.27	192.03	1	2
3	1.02224	1.96724	0.009271	95.18	192.13	1	2
GT	1	2	0.02225	96.15	196.02	1	2

Table 5.4: Proposition 4, 60:30:10 ratio, $T = 50$.

Run	V	σ^2	C	NIS Mean	NIS Var	Orig q	Orig R
1	0.97595	2.11590	0.01408	94.66	192.00	1	2
2	0.97212	2.12588	0.01454	94.64	191.95	1	2
3	0.98947	2.08361	0.01412	94.67	192.04	1	2
GT	1	2	0.02969	96.04	197.70	1	2

The 33:33:33 ratio gives parameter estimates closest to the ground truth, while the 60:30:10 ratio drifts further away. This suggests that skewed interval distributions reduce stability in recovery. A plausible explanation is the mismatch between model and filter discretisations, which biases the innovations. With moderate variability in ΔT, the ridge sharpens and recovery improves, but once the distribution becomes skewed, the surface distorts.

Interestingly, for the more balanced 33:33:33 case, the overall CNIS is actually lower than in the constant-ΔT setup. Table 4.7 shows that using the ground truth V and σ^2, the CNIS is 0.027024, whereas the varied-Δt ground truth results gave 0.02225. By contrast, the skewed 60:30:10 ratio performed worse, with a value of 0.02969. The BO-recovered parameters also show smaller CNIS values under the even ratio. In effect, the balanced variability brings V closer to its true value while still favouring σ^2, consistent with the shifted ridge structure described earlier. Since the even ratio clearly improves recovery while the skewed version distorts it, the 33:33:33 case will be carried forward for further testing at $T = 100$ to see if it helps mitigate the outlier behaviour observed there.

5.1.5 Outlier Case: $T = 100$ in Proposition 4

The $T = 100$ case in Proposition 4 was previously highlighted as an outlier. Figure 5.4 shows that, similar to the $T = 50$ case, introducing variability in ΔT shifts the surface to resemble Proposition 3 more closely.

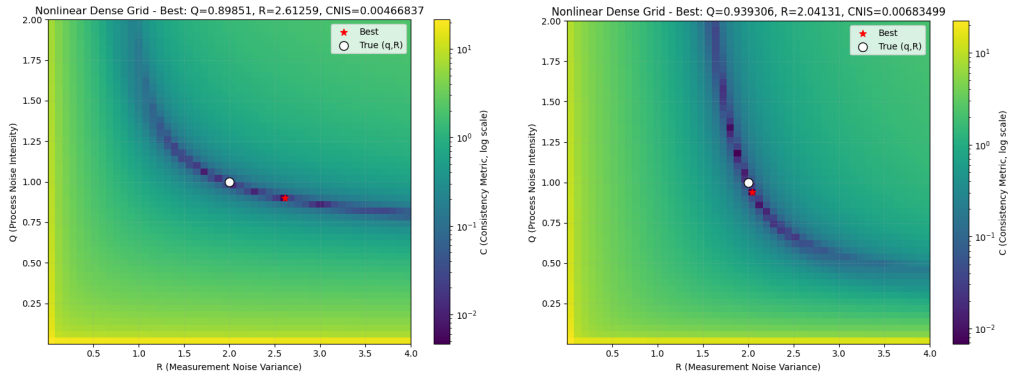


Figure 5.4: Grid search landscapes for Proposition 4, $T = 100$: 33:33:33 vs. constant ΔT.

On the grid search, the best parameter lies far from the true parameter, already pointing to the poor behaviour that follows.

Table 5.5: Proposition 4, 33:33:33 ratio, $T = 100$.

Run	V	σ^2	C	NIS Mean	NIS Var	Orig q	Orig R
1	1.12305	1.55032	0.001408	196.11	392.33	1	2
2	0.89377	2.63507	0.001251	196.23	391.96	1	2
3	1.11730	1.56849	0.002110	196.03	391.23	1	2
GT	1	2	0.02200	195.98	383.50	1	2

Here, the introduction of mixed ΔT amplifies the poor behaviour already present. CNIS values increase across the board compared with constant ΔT, including at the ground truth, and the surface shifts further away from the expected minimum. Outliers are magnified, leading to unstable (V, σ^2) estimates (Table 5.5).

This essentially confirms that variability in ΔT can worsen performance when results are already poor, making it harder to detect or correct issues. The fact that $T = 100$ was identified as an outlier in Section 4.4 only compounds the problem, leading to highly unstable convergence of V and σ^2 away from the true values.

5.1.6 Summary

To summarise, introducing variable ΔT had little effect on Proposition 3, but led to amplified and shifted behaviour in Proposition 4. The even 33:33:33 split generally improved recovery, whereas skewed ratios tended to worsen stability. Outlier results could not be corrected, and were even made worse. These findings show that while balanced variability can sharpen the optimisation surface, it can also magnify poor behaviour when conditions are already unfavourable.

In the following section, CNIS-based measures are replaced with MSE, to examine whether improving consistency through tuning actually impacts error in this factor graph estimation problem, similar to Kalman Filters.

5.2 Extension 2: Mean Squared Error Observations

Up to this point, evaluation has focussed on consistency-based measures such as NIS, NEES, and their normalised variants. These have been central to the Bayesian optimisation experiments [8], i.e. that improving consistency should indirectly improve accuracy. While this has been widely discussed in the Kalman filter literature as using correctly tuned noise parameters is vital, it is not obvious that the same link holds in the factor graph setting used here. This section therefore shifts focus to accuracy directly, by measuring mean squared error (MSE) of estimated trajectories.

5.2.1 Setup

The setup mirrors the earlier grid search experiments, but with MSE replacing CNIS as the evaluation metric. A blanket search was performed across (V, σ^2) pairs, with average MSE computed over 10,000 Monte Carlo runs. To make the results interpretable, three plots were produced at $T = 50$:

- **2D heatmap** showing raw MSE values.
- **Thresholded heatmap** collapsing all MSE > 1 to a single colour, highlighting the low-error region.
- **3D surface** showing the MSE landscape over the (V, σ^2) grid.

In addition to these plots, a selection of parameter settings from earlier sections were re-tested to assess whether tuning had an impact on MSE. The resulting table compares trajectory lengths $T \in \{20, 50, 100, 200\}$ across ground truth parameters, BayesOpt-tuned parameters (from both propositions), and deliberately poor noise settings.

5.2.2 Results and Discussion

Figures 5.5, 5.6, and 5.7 show the $T = 50$ results. The overall MSE surface is flat: almost all reasonable (V, σ^2) values achieve errors well below 1. The thresholded plot emphasises that the majority of the parameter space lies in this low-error regime, with only extreme noise misspecifications producing degraded performance.

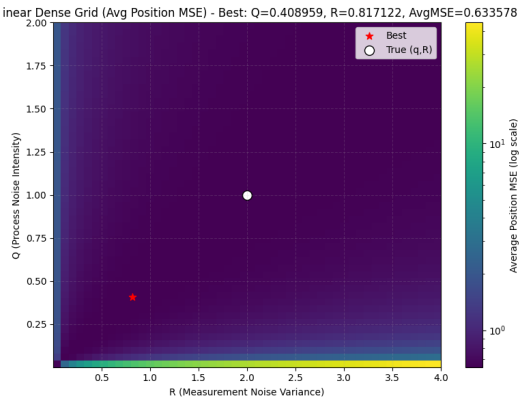


Figure 5.5: MSE heatmap, $T = 50$.

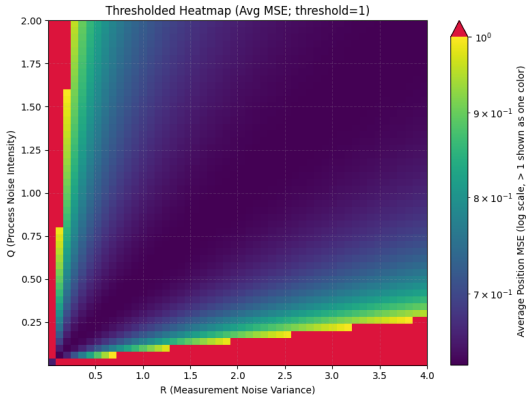


Figure 5.6: MSE heatmap, $T = 50$. Regions with MSE > 1 are collapsed to a single colour.

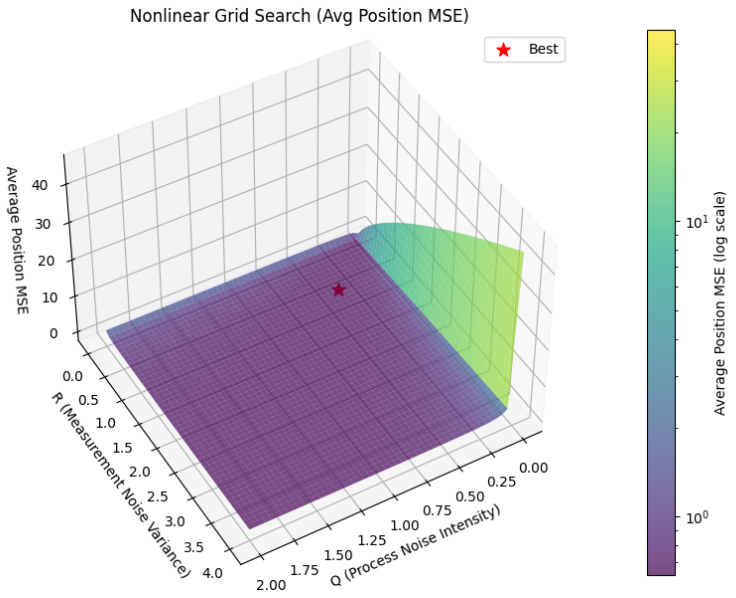


Figure 5.7: 3D MSE surface over (V, σ^2), $T = 50$.

Table 5.6: MSE results for the nonlinear model across different trajectory lengths and parameter settings.

T	GT	Best Prop 3	Best Prop 4	Large Measurement	Large Process
200	0.60447	0.604468	0.604468	9.29314	1.89557
100	0.614767	0.614767	0.619497	9.33039	1.90011
50	0.633578	0.633582	0.633626	9.39309	1.90294
20	0.688271	0.688272	0.688283	8.91858	1.90551

Table 5.7: MSE results for the linear model across different trajectory lengths and parameter settings.

T	GT	Best Prop 3	Best Prop 4	Large Measurement	Large Process
200	0.603196	0.603194	0.603194	9.48971	1.89546
100	0.613518	0.613518	0.618237	9.50112	1.90000
50	0.632351	0.632355	0.632401	9.48916	1.90283
20	0.687119	0.687120	0.687131	1.90540	1.90540

The tabulated averages confirm the graphical trends. For both the nonlinear (Table 5.6) and linear (Table 5.7) models, the ground truth and BayesOpt-tuned parameters produce almost identical results across all T. Longer trajectories (e.g. $T = 200$) yield slightly lower MSE, consistent with reduced estimator variance when more data are available.

The impact of deliberately poor parameter settings is immediately visible. Large process noise consistently produces moderate degradation, with MSE values clustering around ~ 1.9. In contrast, large measurement noise causes severe errors (~ 9), overwhelming the estimator and confirming its greater sensitivity to mis-specified measurement uncertainty. This asymmetry is natural given the true parameter setting $(V, \sigma^2) = (1, 2)$: underestimating measurement precision penalises accuracy more strongly than modest distortions in process modelling.

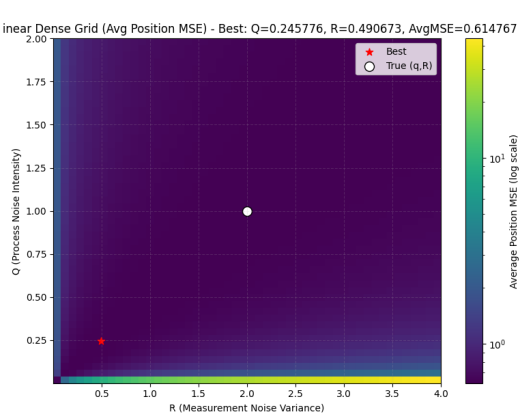


Figure 5.8: MSE heatmap, $T = 100$.

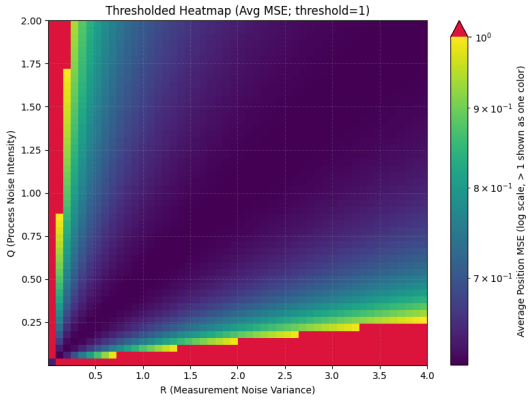


Figure 5.9: MSE heatmap, $T = 100$. Regions with MSE > 1 are collapsed to a single colour.

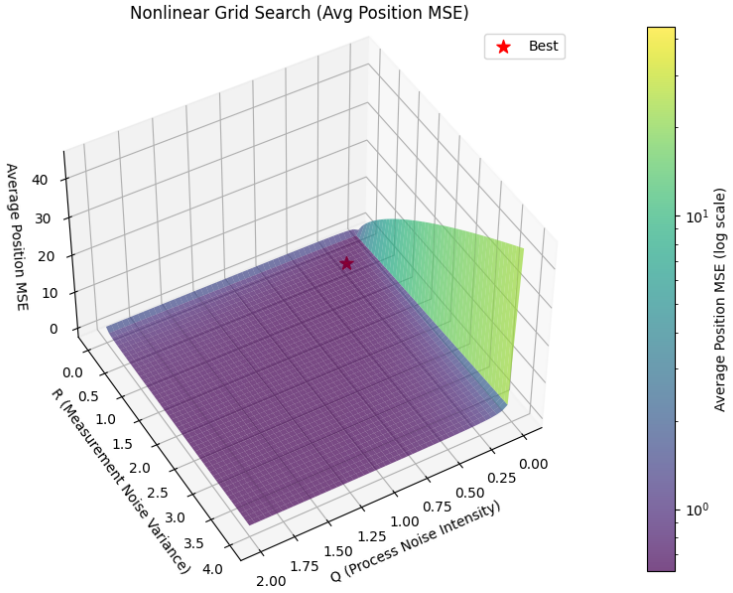


Figure 5.10: 3D MSE surface over (V, σ^2), $T = 100$.

Perhaps most striking is the behaviour of the $T = 100$ case. Despite being identified as an outlier in the consistency-based experiments (Section 4.4), its MSE is practically indistinguishable from the other trajectory lengths. This demonstrates that even when consistency with the noise parameters appear unstable, overall trajectory accuracy remains robust. In other words, consistency measures may flag problematic convergence behaviour, but the estimator can still achieve low MSE.

A final comparison between linear and nonlinear cases shows only marginal differences. The linear model tends to produce slightly lower MSE, which can be attributed to simpler dynamics and less variability across trajectories, making estimation easier. The nonlinear constant-turn model introduces more structural uncertainty, which inflates MSE slightly, but the differences remain small overall.

Taken together, these results suggest that while Bayesian tuning is effective for improving consistency, accuracy in terms of MSE is already well preserved across most of the parameter space. Tuning matters far less for MSE than for CNIS in GPS based factor graph estimation, though extreme mis-specification can still cause severe degradation.

Chapter 6

Conclusions and Future Work

6.1 Conclusion

This dissertation has presented a comprehensive analysis of automatic tuning for factor graph-based estimation using Bayesian Optimization (BO). The work demonstrates a proof-of-concept for automatically tuning the noise parameters of a factor graph-based tracking system, providing a significant advancement over manual tuning and brute-force grid searches.

A central contribution of this study is the validation of the intricate "diagonal valley" structure of the tuning landscape and the critical role of Monte Carlo convergence in obtaining a stable objective function. It was shown that a sufficient number of Monte Carlo runs is essential for reliable evaluation of the *Consistent Normalized Innovation Squared* (CNIS) metric. Specifically, while preliminary experiments with $N = 500$ runs produced a flat and noisy landscape, stable CNIS estimates for Proposition 3 required approximately $N = 5,000$ runs, and Proposition 4 required at least $N = 10,000$ runs. Probability density function analyses and Pearson tests confirmed the convergence of the NIS distribution to the expected χ^2 distribution at these higher sample counts, highlighting the statistical necessity for large simulation budgets.

The dense grid search across both linear and nonlinear models revealed a smooth, diagonal "valley" in the CNIS cost surface, reflecting a fundamental trade-off between process noise intensity (V) and measurement noise variance (σ^2). This broad and flat valley indicates that multiple (V, σ^2) combinations can achieve statistical consistency, explaining why Bayesian Optimization can converge to low CNIS values that do not necessarily correspond to the ground truth parameters. This susceptibility to noise-induced strange minima behaviour, particularly evident in the trajectory-length study at $T = 100$, illustrating a structural property of the optimization problem rather than an algorithmic failure.

Despite successful CNIS minimization, meaning $\mathbf{Q}$ and $\mathbf{R}$ covariance matrix were adequately found, the analysis revealed a near-zero correlation between CNIS and the estimator's *Mean Squared Error* (MSE) for the chosen system model. Dense grid search validation showed that a wide range of parameter values, including ground truth and BO-tuned parameters, achieved similarly low MSE. This indicates that the factor graph-based estimator is inherently robust and that precise CNIS optimization is not strictly necessary to improve tracking accuracy, only "good enough" guesses are required, making sure that large inbalances between V and σ^2 aren't picked.

This distinction underlines an important subtlety. Statistical consistency tests ensure that the assumed noise models align with the observed innovation statistics, meaning the system "correctly" identifies its noise parameters. While this is valuable for diagnosis and theoretical soundness, it does

not necessarily translate into better tracking performance in practice. For the GPS tracking factor graph studied here, estimator robustness meant that even imperfectly tuned parameters produced similar MSE outcomes, limiting the practical impact of fine tuning consistency optimization.

Finally, the study explored the impact of variable timesteps. For Proposition 3 (ideal initialization with no optimisation), heterogeneous time intervals had minimal effect, based on the fact the setup of the $\mathbf{R}$ parameter was not dependent on the time step. For Proposition 4 (misaligned initialization + optimisation), variable timesteps amplified and shifted the CNIS landscape, demonstrating the fragility of the tuning process under non-ideal conditions.

In summary, the dissertation establishes that while Bayesian Optimization is a powerful method for automatically tuning factor graph parameters, reliance on a single consistency metric such as CNIS may not suffice for practical accuracy. The results highlight the need for multi-objective approaches and more robust, computationally efficient frameworks for real-world deployment.

6.2 Future Work

Based on the results of this project, several directions for future work can be identified. These aim to improve both the performance of the tuning framework and its relevance to practical estimation problems.

6.2.1 Multi-Objective Optimization

The difference observed between CNIS and MSE suggests that a single objective is not always enough. A useful extension would be a multi-objective Bayesian Optimization (MOBO) approach, which considers both consistency and accuracy at the same time. Instead of giving one "best" set of parameters, this would provide a set of trade-offs, allowing the choice of parameters to depend on the priorities of a specific application.

6.2.2 Efficiency and Robustness

The framework in its current form is computationally expensive and at times sensitive to strange minima behaviour. Two ways forward are:

- **Multi-Fidelity Bayesian Optimization (MFBO):** Using cheaper, lower-fidelity evaluations (e.g., shorter runs or fewer Monte Carlo samples) to guide the search before confirming candidates with more accurate evaluations. This could cut down overall cost.
- **Noise-Aware Acquisition Functions:** Developing acquisition functions that explicitly handle noise and flat cost surfaces, helping avoid poor convergence.

6.2.3 Alternative Measurement Models

Another line of investigation would be to test the tuning method with different measurement types. For example, replacing position measurements with range-bearing measurements changes both the Jacobians and the structure of the noise covariance matrices. This would test how well the approach generalises. Key questions include:

- How does the CNIS surface change when using range-bearing measurements?
- Does Bayesian Optimization remain effective at finding good parameter regions in this case?

- How does the relationship between CNIS and MSE change with these measurements?

Testing such cases would show whether the method is flexible enough for a wider range of estimation problems, including robotics and navigation tasks that rely on non-linear sensors.

6.2.4 Application to Real-World Problems

Finally, there are several ways this work could be pushed towards real applications:

- **Non-Gaussian Noise:** Real sensors often produce outliers or heavy-tailed noise, so relaxing the Gaussian assumption could improve robustness, but comes with its own set of problems as the theory described in chapter 2 relies on Gaussian distributed assumptions.
- **Adaptive Tuning:** Moving from an offline method to an online one could allow parameters to adapt as conditions change. Reinforcement learning may offer one route, though this brings new challenges for safety and stability.
- **Multi-Sensor Fusion and SLAM:** Extending the framework to multi-sensor or SLAM problems would test scalability, since these involve larger parameter spaces. Multi-fidelity ideas could again help to manage complexity.

Overall, these future directions would help to move the method from a proof of concept towards a more general and practical tool for automatic tuning in factor graph-based estimation.

Appendix A

An Appendix About Stuff

This appendix lists all the source code files used in this dissertation for the Bayesian optimization experiments and factor graph tracking implementation.

A.1 C++ Source Files

- `BO_Tracking_Test_linear.cpp` - Bayesian optimization implementation for linear tracking systems
- `BO_Tracking_Test_Nonlinear.cpp` - Bayesian optimization implementation for nonlinear tracking systems
- `collect_nis_linear.cpp` - NIS data collection utility for linear systems
- `collect_nis_nonlinear.cpp` - NIS data collection utility for nonlinear systems
- `convergence_test_linear.cpp` - Convergence analysis for linear tracking
- `convergence_test_nonlinear.cpp` - Convergence analysis for nonlinear tracking
- `CrossSection_Tracking_Nonlinear.cpp` - Cross-section analysis for nonlinear systems
- `Evaluate_MSE_Points_Linear.cpp` - MSE evaluation at specific parameter points for linear systems
- `Evaluate_MSE_Points_Nonlinear.cpp` - MSE evaluation at specific parameter points for nonlinear systems
- `fg_class_tracking.cpp` - Main factor graph tracking class implementation
- `fg_class_tracking.h` - Header file for factor graph tracking class
- `GridSearch_Tracking_linear.cpp` - Grid search optimization for linear systems
- `GridSearch_Tracking_Nonlinear.cpp` - Grid search optimization for nonlinear systems
- `GridSearch_Tracking_Linear_MSE.cpp` - Grid search minimising average position MSE for linear systems
- `GridSearch_Tracking_Nonlinear_MSE.cpp` - Grid search minimising average position MSE for nonlinear systems

- `tracking_gen_data_linear.cpp` - Linear trajectory data generation
- `tracking_gen_data_nonlinear.cpp` - Nonlinear trajectory data generation
- `2D_h5_loader.h` - HDF5 loaders for states and measurements

A.2 Configuration Files

- `scenario_linear.yaml` - Configuration parameters for linear tracking experiments
- `scenario_nonlinear.yaml` - Configuration parameters for nonlinear tracking experiments

All source code files are available in the project repository on github: https://github.com/WillTerry01/UCL-Dissertation.

Bibliography

[1] R. E. Kalman, "A new approach to linear filtering and prediction problems," *ASME Journal of Basic Engineering*, vol. 82, no. 1, pp. 35–45, 1960.

[2] N. J. Gordon, D. J. Salmond, and A. F. Smith, "Novel approach to nonlinear/non-gaussian bayesian state estimation," in *IEE proceedings F (radar and signal processing)*, vol. 140, pp. 107–113, IET, 1993.

[3] F. Dellaert, "Factor graphs and gtsam: A hands-on introduction," *Georgia Institute of Technology, Tech. Rep*, vol. 2, no. 4, 2012.

[4] F. R. Kschischang, B. J. Frey, and H.-A. Loeliger, "Factor graphs and the sum-product algorithm," *IEEE Transactions on information theory*, vol. 47, no. 2, pp. 498–519, 2002.

[5] F. Dellaert, M. Kaess, *et al.*, "Factor graphs for robot perception," *Foundations and Trends® in Robotics*, vol. 6, no. 1-2, pp. 1–139, 2017.

[6] R. Kümmerle, G. Grisetti, H. Strasdat, K. Konolige, and W. Burgard, "g 2 o: A general framework for graph optimization," in *2011 IEEE international conference on robotics and automation*, pp. 3607–3613, IEEE, 2011.

[7] Z. A. Ahmed and S. M. Raafat, "An extensive analysis and fine-tuning of gmapping's initialization parameters.," *International Journal of Intelligent Engineering & Systems*, vol. 16, no. 3, 2023.

[8] Z. Chen, H. Biggie, N. Ahmed, S. Julier, and C. Heckman, "Kalman filter auto-tuning with consistent and robust bayesian optimization," *IEEE Transactions on Aerospace and Electronic Systems*, vol. 60, no. 2, pp. 2236–2250, 2024.

[9] D. Simon, *Optimal state estimation: Kalman, H infinity, and nonlinear approaches*. John Wiley & Sons, 2006.

[10] S. J. Julier and J. K. Uhlmann, "New extension of the kalman filter to nonlinear systems," in *Signal processing, sensor fusion, and target recognition VI*, vol. 3068, pp. 182–193, Spie, 1997.

[11] Y. Bar-Shalom, X. R. Li, and T. Kirubarajan, *Estimation with applications to tracking and navigation: theory algorithms and software*. John Wiley & Sons, 2001.

[12] Z. Chen, N. Ahmed, S. Julier, and C. Heckman, "Kalman filter tuning with bayesian optimization," *arXiv preprint arXiv:1912.08601*, 2019.

[13] D. Koller and N. Friedman, *Probabilistic graphical models: principles and techniques*. MIT press, 2009.

[14] M. Kaess, H. Johannsson, R. Roberts, V. Ila, J. J. Leonard, and F. Dellaert, "isam2: Incremental smoothing and mapping using the bayes tree," *The International Journal of Robotics Research*, vol. 31, no. 2, pp. 216–235, 2012.

[15] J. Nocedal and S. J. Wright, *Numerical optimization*. Springer, 2006.

[16] D. W. Marquardt, "An algorithm for least-squares estimation of nonlinear parameters," *Journal of the society for Industrial and Applied Mathematics*, vol. 11, no. 2, pp. 431–441, 1963.

[17] K. Levenberg, "A method for the solution of certain non-linear problems in least squares," *Quarterly of applied mathematics*, vol. 2, no. 2, pp. 164–168, 1944.

[18] G. Grisetti, R. Kümmerle, C. Stachniss, and W. Burgard, "A tutorial on graph-based slam," *IEEE Intelligent Transportation Systems Magazine*, vol. 2, no. 4, pp. 31–43, 2011.

[19] P. S. Maybeck, *Stochastic models, estimation, and control*, vol. 3. Academic press, 1982.

[20] B. D. Anderson and J. B. Moore, *Optimal filtering*. Courier Corporation, 2005.

[21] R. G. Brown and P. Y. Hwang, "Introduction to random signals and applied kalman filtering: with matlab exercises and solutions," *Introduction to random signals and applied Kalman filtering: with MATLAB exercises and solutions*, 1997.

[22] S. S. Blackman and R. Popoli, "Design and analysis of modern tracking systems," *(No Title)*, 1999.

[23] X. R. Li and V. P. Jilkov, "Survey of maneuvering target tracking. part i. dynamic models," *IEEE Transactions on aerospace and electronic systems*, vol. 39, no. 4, pp. 1333–1364, 2003.

[24] R. K. Mehra and J. Peschon, "An innovations approach to fault detection and diagnosis in dynamic systems," *Automatica*, vol. 7, no. 5, pp. 637–640, 1971.

[25] Y. Bar-Shalom and X.-R. Li, "Estimation and tracking- principles, techniques, and software," *Norwood, MA: Artech House, Inc, 1993.*, 1993.

[26] Z. Chen, C. Heckman, S. Julier, and N. Ahmed, "Weak in the nees?: Auto-tuning kalman filters with bayesian optimization," in *2018 21st International Conference on Information Fusion (FUSION)*, pp. 1072–1079, IEEE, 2018.

[27] M. Theiler, D. Schneider, and C. Endisch, "Kalman filter tuning using multi-objective genetic algorithm for state and parameter estimation of lithium-ion cells," *Batteries*, vol. 8, no. 9, p. 104, 2022.

[28] P. Mößle, T. Tietze, and M. A. Danzer, "Kalman filter tuning for state estimation of lithium-ion batteries by multi-objective optimization via hyperspace exploration," *Energy Technology*, vol. 11, no. 12, p. 2300796, 2023.

[29] X. Zhang, Z. Song, and Y. Wang, "Reinforcement learning-based kalman filter for adaptive brain control in brain-machine interface," in *2021 43rd Annual International Conference of the IEEE Engineering in Medicine & Biology Society (EMBC)*, pp. 6619–6622, IEEE, 2021.

[30] E. Brochu, V. M. Cora, and N. de Freitas, "A tutorial on bayesian optimization of expensive cost functions," 2009.

[31] B. Shahriari, 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, 2015.

[32] P. I. Frazier, "A tutorial on bayesian optimization," *arXiv preprint arXiv:1807.02811*, 2018.

[33] C. K. Williams and C. E. Rasmussen, *Gaussian processes for machine learning*, vol. 2. MIT press Cambridge, MA, 2006.

[34] D. R. Jones, M. Schonlau, and W. J. Welch, "Efficient global optimization of expensive black-box functions," *Journal of Global optimization*, vol. 13, no. 4, pp. 455–492, 1998.

[35] N. Srinivas, A. Krause, S. M. Kakade, and M. Seeger, "Gaussian process optimization in the bandit setting: No regret and experimental design," *arXiv preprint arXiv:0912.3995*, 2009.

[36] A. Simpkins, "System identification: Theory for the user, (ljung, l.; 1999)[on the shelf]," *IEEE Robotics & Automation Magazine*, vol. 19, no. 2, pp. 95–96, 2012.