Weak in the NEES?: Auto-tuning Kalman Filters with Bayesian Optimization

Zhaozhong Chen, Christoffer Heckman
Department of Computer Science
University of Colorado Boulder
430 UCB
Boulder, CO 80309
email: Christoffer.Heckman@colorado.edu

Simon Julier
Department of Computer Science
University College London
66–72 Gower Street,
London WC1E 6BT, UK
email: s.julier@ucl.ac.uk

Nisar Ahmed
Smead Aerospace Engineering Sciences
University of Colorado Boulder
429 UCB
Boulder, CO 80309
email: Nisar.Ahmed@colorado.edu

***Abstract*—Kalman filters are routinely used for many data fusion applications including navigation, tracking, and simultaneous localization and mapping problems. However, significant time and effort is frequently required to tune various Kalman filter model parameters, e.g. process noise covariance, pre-whitening filter models for non-white noise, etc. Conventional optimization techniques for tuning can get stuck in poor local minima and can be expensive to implement with real sensor data. To address these issues, a new "black box" Bayesian optimization strategy is developed for automatically tuning Kalman filters. In this approach, performance is characterized by one of two stochastic objective functions: normalized estimation error squared (NEES) when ground truth state models are available, or the normalized innovation error squared (NIS) when only sensor data is available. By intelligently sampling the parameter space to both learn and exploit a nonparametric Gaussian process surrogate function for the NEES/NIS costs, Bayesian optimization can efficiently identify multiple local minima and provide uncertainty quantification on its results.**

I. Introduction

Although research in the past few years has introduced many new estimation algorithms, the Kalman filter still remains one of the most widely used algorithms in the world today. Its popularity can largely be attributed to its efficiency, simplicity and robustness.

A major challenge in developing a Kalman filter is that it must be *tuned*. Given a real world application and a system design, the process and observation covariance matrices must be set to given an acceptable level of performance. This performance is often defined in terms of the mean squared error of the estimate. The general idea behind tuning is to search over the space of filter parameters and assess performance. If one has access to ground truth (either from an extra measurement system or simulation), the performance can be assessed statistically based on the normalized estimation error squared (NEES). However, often only observation sequences are available and thus the normalized innovation squared (NIS) must be used instead. Tuning then becomes a problem of balancing the behaviour of the filter performance metric over time. One approach is to do this manually, i.e. simply explore over all available degrees of freedom until good results obtained. However, this can often be a long and difficult process which requires studying the interaction of many different filter parameters.

Given the difficulty of manual tuning, methods for automating filter tuning are of great practical interest. These methods typically pose tuning as an optimisation problem: given a measure of performance, such as NIS and NEES, iterate through points in parameter space to find the one which provides the best results. These can give very good results. However, a key issue is that the optimisation problem is often highly non-covex. As a result, gradient-based optimization algorithms suffer from the possibility that they could fall into a local minima.

In this paper, we consider the problem of how to develop Kalman filter tuning algorithm using Bayesian Optimization. Our idea is to recast optimization as a Bayesian search problem in which the next iteration of the optimizer seeks a point which maximizes the probability of improving an overall measure of the state estimator performance. As such, Bayesian optimization offers a potentially principled way to handle the local minima problem. For this initial investigation, we restrict ourselves to linear systems. However, the underlying principles apply to nonlinear systems as well and are amenable to extension covering these cases.

This paper is structured as follows. Section II introduces the filter tuning problem. Section III provides an overview of Bayesian optimization using Gaussian process models for optimizing stochastic black box cost functions, and then describes its novel application to Kalman filter tuning using cost functions based on χ^2 consistency test statistics. Section IV presents numerical examples showing the application of Bayesian optimization auto-tuning to a linear system. Conclusions and ongoing/ future work are given in Section V.

II. Preliminaries

A. System Description

Consider the problem of estimating the state and quantifying the uncertainty in that estimate in discrete time. Let the state of the system at time step k be $\mathbf{x}_k$; our goal is to develop an algorithm that can result in the state estimate. Let $\hat{\mathbf{x}}_{i|j}$ be the

estimate of $\mathbf{x}_i$ using all observations up to time step j, and the covariance of this estimate be $\mathbf{P}_{i|j}$:

$$\hat{\mathbf{x}}_{i|j} = \mathrm{E}\left[\mathbf{x}_i|\mathbf{z}_{1:j}\right] \tag{1}$$

$$\mathbf{P}_{i|j} = \mathrm{E}\left[\left(\mathbf{x}_i - \hat{\mathbf{x}}_{i|j}\right)|\mathbf{z}_{1:j}\left(\mathbf{x}_i - \hat{\mathbf{x}}_{i|j}\right)|\mathbf{z}_{1:j}{}^{\top}\right]. \tag{2}$$

The system is described by a process model and an observation model. The process model that describes how the system evolves from state $k-1$ to k is:

$$\mathbf{x}_k = \mathbf{F}_k\mathbf{x}_{k-1} + \mathbf{B}_k\mathbf{u}_k + \mathbf{v}_k, \tag{3}$$

where $\mathbf{u}_k$ is the control input and $\mathbf{v}_k$ is the process noise, which is assumed to be zero mean and independent with covariance $\mathbf{Q}_k$. The observation model is

$$\mathbf{z}_k = \mathbf{H}_k\mathbf{x}_k + \mathbf{w}_k, \tag{4}$$

where $\mathbf{w}_k$ is the observation noise. This is assumed to be zero-mean and independent with a covariance $\mathbf{R}_k$.

As is well-known, a Kalman filter may be applied to this problem in order to find the optimal estimate [1]; this filter follows a two stage process of prediction followed by update. The predicted state is given by

$$\hat{\mathbf{x}}_{k|k-1} = \mathbf{F}_k\hat{\mathbf{x}}_{k-1|k-1} + \mathbf{B}_k\mathbf{u}_k \tag{5}$$

$$\mathbf{P}_{k|k-1} = \mathbf{F}_k\mathbf{P}_{k-1|k-1}\mathbf{F}_k^{\top} + \mathbf{Q}_k \tag{6}$$

while the update is given by

$$\hat{\mathbf{x}}_{k|k} = \hat{\mathbf{x}}_{k|k} + \mathbf{K}_k\mathbf{e}_{\mathbf{z},k}, \tag{7}$$

$$\mathbf{P}_{k|k} = \mathbf{P}_{k|k-1} - \mathbf{K}_k\mathbf{S}_{k|k-1}\mathbf{K}_k^{\top}, \tag{8}$$

$$\mathbf{S}_{k|k-1} = \mathbf{H}_k\mathbf{P}_{k|k-1}\mathbf{H}_k^{\top} + \mathbf{R}_k \tag{9}$$

$$\mathbf{K}_k = \mathbf{P}_{k|k-1}\mathbf{H}_k^{\top}\mathbf{S}_{k|k-1}^{-1} \tag{10}$$

where $\mathbf{e}_{\mathbf{z},k} = \hat{\mathbf{z}}_{k|k-1} - \mathbf{z}_k$ is the so-called *innovation vector*.

A dynamical state estimator is statistically consistent if the following conditions are met [2]:

1) the state estimation errors are unbiased,

$$\mathrm{E}\left[\mathbf{e}_{\mathbf{x},k}\right] = \mathbf{0},\ \forall k \tag{11}$$

2) the estimator is efficient,

$$\mathrm{E}\left[\mathbf{e}_{\mathbf{x},k}\mathbf{e}_{\mathbf{x},k}^T\right] = \mathbf{P}_{k|k},\ \forall k \tag{12}$$

3) the innovations form a white Gaussian sequence, such that for all times k and j,

$$\mathbf{e}_{\mathbf{z},k} \sim \mathcal{N}(\mathbf{0}, \mathbf{S}_{k|k-1}) \tag{13}$$

$$\mathrm{E}\left[\mathbf{e}_{\mathbf{z},k}\right] = \mathbf{0}, \tag{14}$$

$$\mathrm{E}\left[\mathbf{e}_{\mathbf{z},k}\mathbf{e}_{\mathbf{z},j}^T\right] = \delta_{jk}\cdot\mathbf{S}_{k|k-1} \tag{15}$$

Intuitively, a filter is statistically consistent if it correctly describes the actual state error statistics for any set of (simulated) ground truth state sequences, as well as correctly describes the actual measurement residual errors for any set of measurement data logs. When the full structure of the system state ($\mathbf{F}_k$, $\mathbf{H}_k$, $\mathbf{Q}_k$ and $\mathbf{R}_k$) is known, the Kalman filter equations automatically guarantee statistical consistency. However, in many situations the model is not known precisely, and so the filter must be tuned.

B. Filter Tuning

Filter tuning is the process of selecting parameters to optimize performance. Consistency ensures two desirable properties in a Kalman filter: (i) the filter is 'aware' of how wrong it could actually be; and (ii) the filter blends the right amount of information from its process model and measurements to recursively correct its state estimate.

Given values for $\mathbf{F}_k$ and $\mathbf{H}_k$, tuning involves choosing $\mathbf{Q}_k$ and $\mathbf{R}_k$. If the model is matched ($\mathbf{F}_k$ and $\mathbf{H}_k$ are the same as the true system), the statistical consistency can be achieved. However, in general the model can be mismatched. In this case, we seek to satisfy the weaker condition of *covariance consistency*,

$$\hat{\mathbf{x}}_{i|j} \approx \mathrm{E}\left[\mathbf{x}_i|\mathbf{z}_{1:j}\right] \tag{16}$$

$$\mathbf{P}_{i|j} \geq \mathrm{E}\left[\left(\mathbf{x}_i - \hat{\mathbf{x}}_{i|j}\right)|\mathbf{z}_{1:j}\left(\mathbf{x}_i - \hat{\mathbf{x}}_{i|j}\right)|\mathbf{z}_{1:j}{}^{\top}\right]. \tag{17}$$

where $\approx$ is application specific and $\mathbf{A} \geq \mathbf{B}$ means that $\mathbf{A} - \mathbf{B}$ is positive semidefinite. In other words, the estimate should be approximately unbiased, and the estimator should not over estimate its level of confidence. At the same time, the estimated covariance should not be very large.

These conditions can be assessed by examining the normalized scalar magnitudes of the random variables $\mathbf{e}_{\mathbf{x},k}$ and $\mathbf{e}_{\mathbf{z},k}$,

$$\epsilon_{\mathbf{x},k} = \mathbf{e}_{\mathbf{x},k}^T\mathbf{P}_{k|k}^{-1}\mathbf{e}_{\mathbf{x},k} \tag{18}$$

$$\epsilon_{\mathbf{z},k} = \mathbf{e}_{\mathbf{z},k}^T\mathbf{S}_{k|k-1}^{-1}\mathbf{e}_{\mathbf{z},k}, \tag{19}$$

which define the *normalized estimation error squared (NEES)* and *normalized innovation error squared (NIS)*, respectively. If the dynamical consistency conditions are met, then it is easy to show that $\epsilon_{\mathbf{x},k}$ and $\epsilon_{\mathbf{z},k}$ should be χ^2 random variables with $n_{\mathbf{x}}$ and $n_{\mathbf{z}}$ degrees of freedom, respectively [2]. Therefore, χ^2 hypothesis tests can be performed on calculated values for $\epsilon_{\mathbf{x},k}$ (when ground truth data is available) and $\epsilon_{\mathbf{z},k}$ to see if the consistency conditions hold at each time k.

In practice, NEES χ^2 tests are conducted using multiple offline Monte Carlo 'truth model' simulations to obtain ground truth $\mathbf{x}_k$ values. The truth model simulator represents a high-fidelity model of the 'actual' system dynamics and sensor observations, which may contain non-linearities and other non-ideal characteristics that must be compensated for via Kalman filter tuning. NIS χ^2 can be conducted offline using multiple Monte Carlo simulations (e.g. in parallel with NEES tests), but can also be conducted online with real sensor data logs.

Offline truth model tests are conducted as follows [1]: suppose N independent instances of the true state are randomly initialized according to $\hat{\mathbf{x}}_{0|0}$ and $\mathbf{P}_{0|0}$ (the initial state of the filter), and then propagated through the true stochastic dynamics (3) and measurement model (4) for T time steps, yielding sample ground truth sequences $\mathbf{x}_1^i, \mathbf{x}_2^i, \ldots, \mathbf{x}_T^i$ and measurement sequences $\mathbf{z}_1^i, \mathbf{z}_2^i, \ldots, \mathbf{z}_T^i$ for $i = 1, \ldots, N$. If the resulting measurement sequences are then fed to a Kalman filter with tuning parameters $(\mathbf{Q}_k, \mathbf{R}_k)$, the resulting NEES

[1] Online NIS tests with real sensor data are similar, but exploit ergodicity of measurement innovation sequences

and NIS statistics for each simulation run i at each time k can be averaged across problem instances to give the test statistics

$$\bar{\epsilon}_{\mathbf{x},k} = \frac{1}{N}\sum_{i=1}^{N} \epsilon^i_{\mathbf{x},k} \tag{20}$$

$$\bar{\epsilon}_{\mathbf{z},k} = \frac{1}{N}\sum_{i=1}^{N} \epsilon^i_{\mathbf{z},k}. \tag{21}$$

Then, given some desired Type I error rate α, the NEES and NIS χ^2 tests provide lower and upper tail bounds $[l_{\mathbf{x}}(\alpha, N), u_{\mathbf{x}}(\alpha, N)]$ and $[l_{\mathbf{z}}(\alpha, N), u_{\mathbf{z}}(\alpha, N)]$, such that the Kalman filter tuning is declared to be consistent if, with probability $100(1-\alpha)$ at each time k,

$$\bar{\epsilon}_{\mathbf{x},k} \in [l_{\mathbf{x}}(\alpha, N), u_{\mathbf{x}}(\alpha, N)] \quad \text{and} \quad \bar{\epsilon}_{\mathbf{z},k} \in [l_{\mathbf{z}}(\alpha, N), u_{\mathbf{z}}(\alpha, N)].$$

Otherwise, the filter is declared to be inconsistent. Specifically, if $\bar{\epsilon}_{\mathbf{x},k} < l_{\mathbf{x}}(\alpha, N)$ or $\bar{\epsilon}_{\mathbf{z},k} < l_{\mathbf{z}}(\alpha, N)$, then the filter tuning is 'pessimistic' ('underconfident'), since the filter-estimated state error/innovation covariances are too large relative to the true values. On the other hand, if $\bar{\epsilon}_{\mathbf{x},k} > u_{\mathbf{x}}(\alpha, N)$ or $\bar{\epsilon}_{\mathbf{z},k} > u_{\mathbf{z}}(\alpha, N)$, then the filter tuning is 'optimistic' ('overconfident'), since the filter-estimated state error/innovation covariance are too small relative to the true values.

The χ^2 consistency tests provide a very principled basis for validating Kalman filter performance in domain-agnostic way, and also provide a well-established means for guiding the tuning of noise parameters $\mathbf{Q}_k$ and $\mathbf{R}_k$ in practical applications. Tuning via the χ^2 tests is most often done manually, and thus requires repeated 'guessing and checking' over multiple Monte Carlo simulation runs. However, this quickly becomes cumbersome and non-trivial for systems with several tunable noise terms. Heuristics for manual filter tuning have been developed in the linear-quadratic optimal control literature [3], e.g. to coarsely tune diagonals of $\mathbf{Q}_k$ first, before fine-tuning the elements of $\mathbf{Q}_k$ further. Such heuristics are useful for bounding the shape and magnitude of $\mathbf{Q}_k$ in linear-Gaussian problems, but are of little help for tuning 'fudge factor' process noise parameters that are used to cope with model errors from state truncation, approximations of non-linearities, poorly modeled dynamics, etc.

Manual tuning is especially challenging if truth model simulations are computationally expensive to run or the filter involves many parameters which can interact with one another in subtle and surprising ways. This not only also makes it difficult to explore the parameter space to properly calibrate heuristics, but also makes it difficult to achieve a large enough N to properly assess inherently noisy NEES/NIS test statistics. Furthermore, because the NEES and NIS are outputs of a stochastic non-differentiable 'black box' simulation function, the filter tuning process cannot be simply automated via conventional convex optimization methods (e.g. line search, gradient descent, etc.). Given this, we need to use alternative optimization techniques which are robust to stochastic variations in the cost function and can explore nonlinear spaces.

III. Bayesian Optimization for Filter Auto-tuning

A common approach to solving nonlinear optimization problems is to use gradient descent. However, the risk with these approaches is that they can fall into local minima. This issue is exacerbated for filter tuning problems where objective functions are governed by noisy dynamical systems. With a finite number of samples, stochastic variation introduces many small local minima and maxima which can trap gradient descent methods. One principled way to handle parameter tuning problems in such cases is to use *Bayesian Optimization* which poses optimization as a Bayesian search problem. The objective function is unknown and is treated as a random variable. A prior is placed over it. As the algorithm proceeds, each iteration takes samples from the objective function which are used to refine the distribution. The next sample point is selected to maximize the probability of improving the current best estimate.

First we will describe Bayesian optimization for dealing with generic 'black box' stochastic objective functions. We then describe its novel application for simulation-based Kalman filter auto-tuning.

A. Bayesian Optimization Theory

Consider the minimization of some objective function $y : \mathcal{Q} \to \mathbb{R}$, where $\mathcal{Q} \in \mathbb{R}^d$ is the search or solution space, and the element $\mathbf{q}^* \in \mathcal{Q}$ is the minimizer, such that $y(\mathcal{Q}^*) \leq y(\mathcal{Q}),\ \forall \mathbf{q} \in \mathcal{Q}$. For simplicity, we assume the solution space is bounded for global optimization, where $\mathcal{Q}(i) \in [\mathbf{q}(i)_l, \mathbf{q}(i)_u]$ for lower bound $\mathbf{q}(i)_l$ and upper bound $\mathbf{q}(i)_u$ for element i of $\mathbf{q}$. When the mapping from $\mathbf{q}$ to y is not known explicitly, the optimization typically requires the evaluation of a 'black box' function. In our application, y is the result of evaluating the performance of a Kalman filter in design configuration $\mathbf{q}$ on a set of synthetic/real sensor data logs generated by a 'true' underlying dynamical system. The black box evaluations of y can therefore be expensive, slow, and produce noisy results for the same input $\mathbf{q}$.

The goal of Bayesian optimization is to find the minimizer of a noisy objective function y that is costly to evaluate at any given design point $\mathbf{q}$, while also learning about the mapping from $\mathbf{q}$ to y at the same time via Bayesian inference. An initial prior belief $p(y)$ over possible y functions is updated by subsequent observations (evidence) E consisting of sample y evaluations for different sampled $\mathbf{q}$ values. Mathematically, this leads to an application of Bayes' rule: $p(y|E) \propto p(E|y)p(y)$, where $p(E|y)$ is the observation likelihood and $p(y|E)$ is the posterior of y given E. Hence, evidence E gives information about the actual shape of y, allowing the posterior belief about the assumed shape of y to be recursively updated. As long as both $p(y)$ and $p(y|E)$ are consistent with the true nature of y, then the law of large numbers ensures that the posterior $p(y|E)$ converges with high probability to the true y, in the limit of infinite observations E covering $\mathcal{Q}$.

Bayesian optimization uses black box point evaluations of y to efficiently find $\mathbf{q}^*$. This is accomplished by maintaining beliefs about how y behaves over all $\mathbf{q}$ in the form of a

"surrogate model" $\mathcal{S}$, which statistically approximates y and is easier to evaluate (e.g. since y might be an expensive high fidelity simulation). During optimization, $\mathcal{S}$ is used to determine where the next design point sample evaluation of y should occur, in order to update beliefs over y and thus simultaneously improve $\mathcal{S}$ while finding the (expected) minimum of y as quickly as possible. The key idea is that, as more observations are sampled at different $\mathbf{q}$ locations, the $\mathbf{q}$ samples themselves eventually converge to the expected minimizer $\mathbf{q}^*$ of y. Since $\mathcal{S}$ contains statistical information about the level of uncertainty in y (i.e. related to $p(y|E)$, the posterior belief), Bayesian optimization effectively leverages probabilistic 'explore-exploit' behavior to learn an approximate model of y while also minimizing it. We next describe the two main components of the Bayesian optimization process: (1) the surrogate model $\mathcal{S}$, which encodes statistical beliefs about y in light of previous observations and a prior belief; and (2) the acquisition function $a(\mathbf{q})$, which is used to intelligently guide the search for $\mathbf{q}^*$ via $\mathcal{S}$.

1) The Surrogate Model: $\mathcal{S}$ must approximate y in areas where it has not yet been evaluated, and must also provide a predicted value and corresponding uncertainty to quantify the possibility that the optimum is located at some location $\mathbf{q}$. Gaussian Processes (GPs) [4] are the most common family of surrogate models used in Bayesian optimization; the acronym GPBO here refers to Bayesian optimization using a GP surrogate model $\mathcal{S}$. A GP describes a distribution over functions; it is more formally defined as a collection of random variables, any finite number of which have a joint Gaussian distribution [5], [4],

$$f(\mathbf{q}) \sim \mathcal{GP}(m(\mathbf{q}), k(\mathbf{q}, \mathbf{q}')) \tag{22}$$

$$m(\mathbf{q}) = \mathbb{E}[f(\mathbf{q})] \tag{23}$$

$$k(\mathbf{q}, \mathbf{q}') = \mathbb{E}[(f(\mathbf{q}) - m(\mathbf{q}))(f(\mathbf{q}') - m(\mathbf{q}'))] \tag{24}$$

where the process is completely specified by its mean function $m(\mathbf{q})$ (equation 23), and its covariance function $k(\mathbf{q}, \mathbf{q}')$ (equation 24). In theory $m(\mathbf{q})$ could be any function; as is common practice, this work assumes m is zero for simplicity. The covariance (or kernel) function is a mapping $k : (\mathbf{q}, \mathbf{q}') \rightarrow \mathbb{R}$; this must be specified a priori, and is usually based on some knowledge of y's smoothness properties.

A valid kernel must be positive semi-definite (PSD), i.e. it must produce a Gram matrix K, with individual elements $[K_{i,j}]$ given by $k(\mathbf{q}_i, \mathbf{q}_j)$, that is PSD given a set of training data $Q = \{\mathbf{q}_1, \ldots, \mathbf{q}_n\}$. Let $K(Q, Q)$ be the Gram matrix defined by kernel function k,

$$K(Q,Q) = \begin{bmatrix} k(\mathbf{q}_1,\mathbf{q}_1) & k(\mathbf{q}_1,\mathbf{q}_2) & \cdots & k(\mathbf{q}_1,\mathbf{q}_n) \\ k(\mathbf{q}_2,\mathbf{q}_1) & k(\mathbf{q}_2,\mathbf{q}_2) & \cdots & k(\mathbf{q}_2,\mathbf{q}_n) \\ \vdots & \vdots & \vdots & \\ k(\mathbf{q}_n,\mathbf{q}_1) & k(\mathbf{q}_n,\mathbf{q}_2) & \cdots & k(\mathbf{q}_n,\mathbf{q}_n) \end{bmatrix}. \tag{25}$$

Given n training observations, the elements of the covariance matrix $K(Q, Q) \in \mathbb{R}^{n \times n}$ are the covariances $k(\mathbf{q}_i, \mathbf{q}_j)$ between $\mathbf{q}_i$ and $\mathbf{q}_j$ for all pairs of training data. The joint distribution of n training outputs $\mathbf{f}(Q) \in \mathbb{R}^{n \times 1}$ and p test outputs $\mathbf{f}_*(Q_*) \in \mathbb{R}^{p \times 1}$ for inputs $Q_* = \{\mathbf{q}_{*1}, \ldots, \mathbf{q}_{*p}\}$ is

$$\begin{bmatrix} \mathbf{f} \\ \mathbf{f}_* \end{bmatrix} \sim \mathcal{N}\left(\mathbf{0}, \begin{bmatrix} K(Q,Q) & K(Q,Q_*) \\ K(Q_*,Q) & K(Q_*,Q_*) \end{bmatrix}\right), \tag{26}$$

$$K(Q_*,Q) = \begin{bmatrix} k(\mathbf{q}_{*1},\mathbf{q}_1) & k(\mathbf{q}_{*1},\mathbf{q}_2) & \cdots & k(\mathbf{q}_{*1},\mathbf{q}_n) \\ k(\mathbf{q}_{*2},\mathbf{q}_1) & k(\mathbf{q}_{*2},\mathbf{q}_2) & \cdots & k(\mathbf{q}_{*2},\mathbf{q}_n) \\ \vdots & \vdots & \ddots & \vdots \\ k(\mathbf{q}_{*p},\mathbf{q}_1) & k(\mathbf{q}_{*p},\mathbf{q}_2) & \cdots & k(\mathbf{q}_{*p},\mathbf{q}_n) \end{bmatrix} \tag{27}$$

Given Q and $\mathbf{f}$, $\mathbf{f}_*$ can be predicted at new 'test locations' $\mathcal{Q}_*$, using the conditional GP mean and covariance relations

$$\mathbf{f}_*|Q_*, Q, \mathbf{f} \sim \mathcal{N}(\mu(Q_*), \sigma^2(Q_*)) \tag{28}$$

$$\mu(Q_*) = K(Q_*,Q)K(Q,Q)^{-1}\mathbf{f} \tag{29}$$

$$\sigma^2(Q_*) = K(Q_*,Q_*) - K(Q_*,Q)K(Q,Q)^{-1}K(Q,Q_*) \tag{30}$$

Here, $K(Q, Q_*) \in \mathbb{R}^{p \times n}$, so that $\mu(Q_*) \in \mathbb{R}^{p \times 1}$ and $\sigma^2(Q_*) \in \mathbb{R}^{p \times p}$. Eq. (28) gives the expression of the conditional distribution of $\mathbf{f}_*$ given test points Q_*, and training data Q and $\mathbf{f}$. The mean and variance of this predictive distribution are found via Eqs. (29) and (30). In the context of Bayesian optimization, the GP surrogate model provides statistical information (i.e. mean and variance from 29 and 30) of how the underlying objective function y behaves for all possible values Q_* that have not yet been sampled.

The Mateŕn kernel is one of the most popular choices for the kernel function k in GPBO,

$$k_{\nu=3/2}(\boldsymbol{x}_{b,i}, \boldsymbol{x}_{b,j}) = \sigma_0 \left(1 + \frac{\sqrt{3}r_{ij}}{\ell}\right) \exp\left(-\frac{\sqrt{3}r_{ij}}{\ell}\right), \tag{31}$$

$$r_{ij} = \sqrt{(\boldsymbol{x}_{b,i} - \boldsymbol{x}_{b,j})^T(\boldsymbol{x}_{b,i} - \boldsymbol{x}_{b,j})}, \tag{32}$$

with hyperparameters σ_0 and ℓ, which are the kernel amplitude and length-scale, respectively. This kernel is guaranteed to be k times differentiable when $k \leq \nu$ (where ν is nearly always taken to be half integer to simplify the kernel expression). As is standard in GP regression, an additive observation noise variance σ_n^2 is also assumed for each training datum $f(\mathbf{q}_i)$, where $\mathbf{q}_i \in Q$

$$f(\mathbf{q}_i) = y(\mathbf{q}_i) + \epsilon_i, \tag{33}$$

$$\epsilon_i \sim \mathcal{N}(0, \sigma_n^2), \tag{34}$$

Hence, the full set of hyperparameters $\Theta = \left\{\sigma_n^2, \sigma_0, \ell\right\}$ governs the GP covariance function in Eq. (24).

Since the best Θ setting is not known a priori, it must be learned and updated during GPBO. Point estimation strategies based on maximum likelihood estimation and maximum a posteriori estimation are the most widely used in the GPBO literature for supervised learning of Θ [6]. Fast gradient-based convex optimization techniques are most commonly used to minimize the negative log likelihood, since the required derivatives can be obtained analytically. However, since the GP

likelihood is generally non-convex, numerical optimization can converge to many different local optima for Θ. Furthermore, the best local optimum may be undesirable for learning with sparse data early on in the GPBO process, since the associated Θ values typically overfit the training data [7], [4]. This behavior is especially important to consider when trying to minimize the number of simulations for GPBO [8].

2) The Acquisition Function: The acquisition function is defined as the mapping $a : (\mathbf{q}, \mathcal{S}) \rightarrow \mathbb{R}$, abbreviated as

$$a(\mathbf{q}) \triangleq a(\mathbf{q}, \mathcal{GP}(m(\mathbf{q}), k(\mathbf{q}, \mathbf{q}'))) \tag{35}$$

which assumes the inclusion of the GP surrogate model as an argument. GPBO selects $\hat{\mathbf{q}} = \text{argmax}_{\mathcal{Q}}\, a(\mathbf{q})$ as the next location in $\mathcal{Q}$ to be evaluated in the search process. Ideally, $a(\mathbf{q})$ should enable exploration and modeling of y by sampling new locations $\mathbf{q}$ that will improve the accuracy of $\mathcal{S}$. At the same time, $a(\mathbf{q})$ must exploit $\mathcal{S}$ to reach the expected minimum of y as quickly as possible. Therefore, $a(\mathbf{q})$ should not lead to greedy or myopic behavior, or get stuck in poor local minima. There are many ways to define $a(\mathbf{q})$ to balance these needs, but the best choice is heavily application dependent [6], [9], [10]. Some popular methods include Expected Improvement (EI) and the Upper Confidence Bound. We focus only on EI here, since it does not require extra hyperparameters.

EI selects the next sample point to maximize the statistically expected improvement in the optimum when when the current best minimizer is $\mathbf{q}^+$. The EI function is defined by [11]

$$a(\mathbf{q}) = \begin{cases} (\mu(\mathbf{q}) - \mathbf{f}(\mathbf{q}^+))\Phi(Z) + \sigma(\mathbf{q})\phi(Z) & , \quad \sigma(\mathbf{q}) > 0 \\ 0 & , \quad \sigma(\mathbf{q}) = 0 \end{cases}$$
$$Z = \frac{\mu\,(\mathbf{q}) - \mathbf{f}(\mathbf{q}^+)}{\sigma(\mathbf{q})},$$

where $\mu(\mathbf{q})$ is the mean predicted value of the GP at $\mathbf{q}$ and $\sigma(\mathbf{q})$ is the predicted standard deviation at $\mathbf{q}$, $\mathbf{f}(\mathbf{q}^+)$ is the best observed value of the objective function, and $\Phi(Z)$ and $\phi(Z)$ are the PDF and CDF of the standard normal distribution Z.

For any definition of $a(\mathbf{q})$, another optimization routine must be used to identify the maximum of $a(\mathbf{q})$ via point-based evaluation on $\mathcal{S}$. The most popular method for doing this in GPBO is the DIviding RECTangles (DIRECT) algorithm [11], which is a fast global non-convex optimization method that uses the Lipschitz continuity properties of $\mathcal{S}$ to bound function values in local rectangles and search accordingly for the best local maximum of $a(\mathbf{q})$. Note that the use of a non-convex optimization technique like DIRECT makes sense here, since they key idea behind Bayesian optimization is that evaluation of $a(\mathbf{q})$ at multiple test points $\mathbf{q}$ will be cheaper and faster than evaluating y at those points directly. In this work, we use the classical approach of selecting a single new design point $\mathbf{q}$ on each iteration of GPBO, although variations to sample multiple design points at once or repeatedly on each iteration are also possible [8].

B. Stochastic Costs for Consistency-based Filter Auto-tuning

We now consider how $y(\mathbf{q})$ can be defined via NEES and NIS consistency test statistics for Kalman filter tuning. As such, let $\mathcal{Q}$ be some space of configurable Kalman filter parameters (e.g. the set of all parameters defining some positive definite symmetric process noise covariance $\mathbf{Q}_k$) and let $\mathbf{q} \in \mathcal{Q}$ be a design point.

Consider first the case of tuning based on assessment of NEES statistics obtained via Monte Carlo ground truth simulation models. If N Monte Carlo simulations are performed for T time steps at any given design point $\mathbf{q}$, starting from the initial conditions $\hat{\mathbf{x}}_{0|0}$ and $\mathbf{P}_{0|0}$, then the average NEES statistic $\bar{\epsilon}_{\mathbf{x},k}$ can be computed via (20) for each time $k = 1, ..., T$. To summarize how 'well-behaved' $\bar{\epsilon}_{\mathbf{x},k}$ is across all time steps, we can leverage the fact that the expected value of $\bar{\epsilon}_{\mathbf{x},k}$ for a consistent Kalman filter ought to be $n_{\mathbf{x}}$, i.e. the degrees of freedom of the χ^2 NEES random variable (which is the same as the number of states). We can therefore use the following scalar function $y(\mathbf{q})$ to assess how much $\bar{\epsilon}_{\mathbf{x},k}$ deviates from this ideal expected value across all time steps k in N Monte Carlo truth model simulations evaluated at $\mathbf{q}$,

$$y(\mathbf{q}) = J_{NEES}(\mathbf{q}) = \sqrt{\left[\log\left(\frac{\sum_{k=1}^{T} \bar{\epsilon}_{\mathbf{x},k}}{n_{\mathbf{x}}}\right)\right]^2} \tag{36}$$

By similar reasoning, we can also define

$$y(\mathbf{q}) = J_{NIS}(\mathbf{q}) = \sqrt{\left[\log\left(\frac{\sum_{k=1}^{T} \bar{\epsilon}_{\mathbf{z},k}}{n_{\mathbf{z}}}\right)\right]^2}. \tag{37}$$

where $\bar{\epsilon}_{\mathbf{z},k}$ could either represent NIS outcomes obtained from truth model simulation or from a set of real data logs.

Many other possible cost functions could also be used to summarize the behavior of the NEES/NIS statistics relative to $n_{\mathbf{x}}$. For instance, instead of the mean over T steps, $y(\mathbf{q})$ could be defined in terms of the min/max or median of $\bar{\epsilon}_{\mathbf{x},k}$ or $\bar{\epsilon}_{\mathbf{z},k}$ vs. $n_{\mathbf{x}}$ over T steps. Or, $y(\mathbf{q})$ could also be based on counting the number of times $\bar{\epsilon}_{\mathbf{x},k}$ or $\bar{\epsilon}_{\mathbf{z},k}$ exceed the χ^2 hypothesis test bounds $[l(\alpha, N), u(\alpha, N)]$ for some given α. While such alternative cost definitions could be useful for different applications (say, depending on the filter parameters being tuned), we focus on J_{NEES} and J_{NIS} here for simplicity.

Algorithm 1 summarizes the GPBO procedure for Kalman filter tuning. The termination criteria could be based on iteration thresholds, tolerances on changes to the optimum $\mathbf{q}$ and/or y between iterations, or other methods. An attractive feature of GPBO is that eqs. 23-24 naturally provide uncertainty quantification on the shape of the objective function at both sampled and unsampled locations. This allows GPBO to cope with multiple local minima in the parameter space $\mathcal{Q}$. However, in practice, the GPBO's performance depends on the selection and parameterization of the surrogate model kernel, as well as the number and placement of initial training observations (i.e. seed points) to bootstrap the search process.

Algorithm 1 GPBO for Kalman Filter tuning

1: Initialize GP with seed data $\{\mathbf{q}_s, y_s\}_{s=1}^{N_{seed}}$ and hyperparameters Θ
2: **while** termination criteria not met **do**
3: $\quad \mathbf{q}_j = \text{argmax}_{\mathcal{Q}}\, a(\mathbf{q})$
4: $\quad$ Evaluate $y(\mathbf{q}_j)$, e.g. using $J_{NEES}(\mathbf{q})$ or $J_{NIS}(\mathbf{q})$.
5: $\quad$ Add $y(\mathbf{q}_j)$ to $\mathbf{f}(Q)$, $\mathbf{q}_j$ to Q, and update Θ
6: **end while**
7: **return** $\mathbf{q}^* = \arg\min_{\mathbf{q}_j \in Q} \mathbf{f}(\mathbf{q}_j)$

IV. Numerical Application Examples

For ease of presenting the proof of concept and discussion in this initial investigation, we restrict ourselves to an application case study involving a simple linear time-invariant system. However, the underlying principles apply to more complex linear and nonlinear systems as well.

Consider a robot that moves along a 1D track and receives position measurements every $\Delta t = 0.1s$. Suppose the position and velocity state $\mathbf{x} = [\xi, \dot{\xi}]^T$ are governed by the linear time invariant kinematics model

$$\dot{\mathbf{x}}_t = \mathbf{A}\mathbf{x}_t + \mathbf{G}\mathbf{u}_t + \mathbf{\Gamma}\mathbf{v}_t$$
$$\mathbf{z}_t = \mathbf{H}\mathbf{x}_t + \mathbf{w}_t,$$

where

$$\mathbf{A} = \begin{bmatrix} 0 & 1 \\ 0 & 0 \end{bmatrix}, \quad \mathbf{G} = \begin{bmatrix} 0 \\ 1 \end{bmatrix}, \quad \mathbf{H} = \begin{bmatrix} 1 & 0 \end{bmatrix}, \quad \mathbf{\Gamma} = \begin{bmatrix} 0 \\ 1 \end{bmatrix},$$

and the inputs to the system consist of a control acceleration $\mathbf{u}_t$, additive white Gaussian noise acceleration process $\mathbf{v}_t$ with intensity $\mathbf{V}$, and additive white Gaussian position measurement noise process $\mathbf{w}_t$ with continuous time intensity $\mathbf{W}$. The control input $\mathbf{u}_t = 2\cos(0.75t)$ causes the robot to move with a low frequency oscillation. Applying a zero-order hold discretization to this system, we obtain discrete time position and velocity state $\mathbf{x}_k = [\xi_k, \dot{\xi}_k]^T$ and the linear time-invariant parameters for eqs. (3)-(4)

$$\mathbf{F} = \begin{bmatrix} 1 & \Delta t \\ 0 & 1 \end{bmatrix}, \quad \mathbf{B} = \begin{bmatrix} 0.5\Delta t^2 \\ \Delta t \end{bmatrix}, \quad \mathbf{H} = \begin{bmatrix} 1 & 0 \end{bmatrix},$$

where the inputs to the system now consist of a discretized zero-order hold control acceleration $\mathbf{u}_k = 2\cos(0.075k)$, additive white process noise vector $\mathbf{v}_k \in \mathbb{R}^2$ with discrete time covariance $\mathbf{Q}_k \in \mathbb{R}^{2\times 2}$, and additive white measurement noise $\mathbf{w}_k$ with discrete time covariance $\mathbf{R}_k$. Note that, given $\mathbf{V}$ and $\mathbf{W}$, the corresponding discrete time noise covariances are

$$\mathbf{R} = \frac{\mathbf{W}}{\Delta T}, \tag{38}$$

$$\mathbf{Q} = \int_0^{\Delta t} \int_0^{\Delta t} e^{\mathbf{A}\Delta t} \mathbf{\Gamma}\mathbf{V}\mathbf{\Gamma}^T e^{\mathbf{A}^T \Delta t} \delta(\tau_1 - \tau_2) d\tau_1 d\tau_2 \tag{39}$$

where the matrix expression for $\mathbf{Q}$ can be computed from $\mathbf{A}$, $\mathbf{\Gamma}$, $\mathbf{V}$, and δT using Van Loan's method [12]. If $\mathbf{A}$ and Γ are both known, then this relationship also allows us to design a full 2×2 positive definite symmetric covariance matrix $\mathbf{Q}$ by tuning the corresponding scalar continuous time process noise acceleration intensity $\mathbf{V}$ only.

We examine GPBO-based Kalman filter tuning for the following cases:

1) tuning of unknown $\mathbf{Q}$ (i.e. unknown $\mathbf{V}$) with correctly known $\mathbf{R}$ and known model dynamics;
2) simultaneous tuning of unknown $\mathbf{Q}$ and unknown $\mathbf{R}$, with correctly known model dynamics;

All results here were obtained using the open source BayesOpt library [13] to apply the update steps and evaluate the surrogate and acquisition functions, with the rest of the code developed by the authors in C++.

A. Case 1: Unknown $\mathbf{Q}$

In this case, $\mathbf{q} = \mathbf{V}$ and the true process noise intensity for ground truth simulations is $\mathbf{V} = 1$ (m/s^2)^2/s, which results in a true discrete time process noise covariance of

$$\mathbf{Q} = \begin{bmatrix} 3 \times 10^{-4} & 5 \times 10^{-3} \\ 5 \times 10^{-3} & 0.1 \end{bmatrix}.$$

GPBO was used to tune the Kalman filter design by searching over $\mathbf{V}$ and using (39) to construct $\mathbf{Q}$, using true measurement noise variance $\mathbf{R} = 1$ m^2 and the true dynamics model.

Figure 1 shows six different iterations of a NEES-based GPBO search using the J_{NEES} cost function over the range $\mathbf{V} = [0, 10]$. In these figures, $N = 10$ Monte Carlo truth model simulations are used per J_{NEES} evaluation, with $T = 200$ time steps.

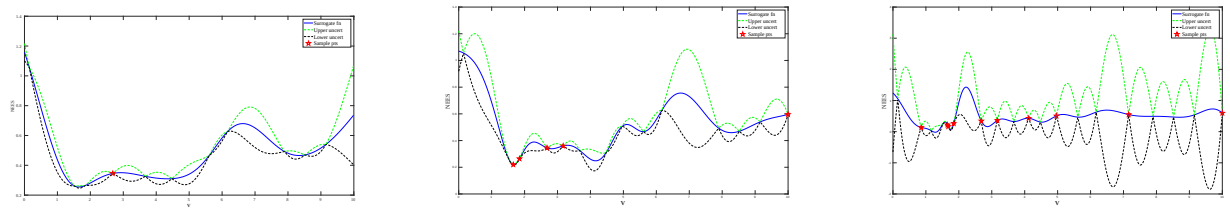
The final result of this trial demonstrates that the minimum of the surrogate function obtained through Bayesian optimization is very close to the ground truth, i.e. that $\mathbf{V} = 1$. Of particular value is also the uncertainty bounds on the surrogate function that clearly demonstrate the uncertainty on these parameters. The figures also demonstrate a clustering of sampling around the true minimum of the objective function, but also a spread of samples in other places to lower uncertainty of minima being in those regions.

B. Case 2: Unknown $\mathbf{Q}$ and $\mathbf{R}$

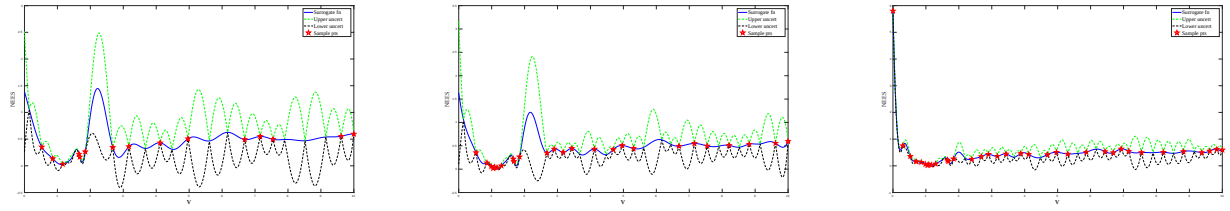
In this case, GPBO was used to tune the Kalman filter design by searching over $\mathbf{V}, \mathbf{Q}$ and using (39) to construct $\mathbf{Q}$, with the noise variance $\mathbf{R}$ also to be estimated (cf. Case 1), while using the true dynamics model for the robot.

Figure 2 shows GPBO search using the J_{NEES} cost function over the range $\mathbf{V} = [0, 10]$ in a 1-dimensional cross section. In these figures, $N = 10$ Monte Carlo truth model simulations are used per J_{NEES} evaluation, with $T = 200$ time steps, just as in Case 1.

Figure 3 shows the full 2-dimensional surrogate function after 100 iterations of Bayesian optimisation. We note that the results for $\mathbf{R}$ appear more accurate than those for $\mathbf{V}$, which exhibits a large disturbance in the surrogate function near $\mathbf{V} \approx 4.5$, leading to a large local minimum nearby. Inspection of the uncertainty around the local minimum demonstrate that the surrogate function requires more iterations to hone in on the

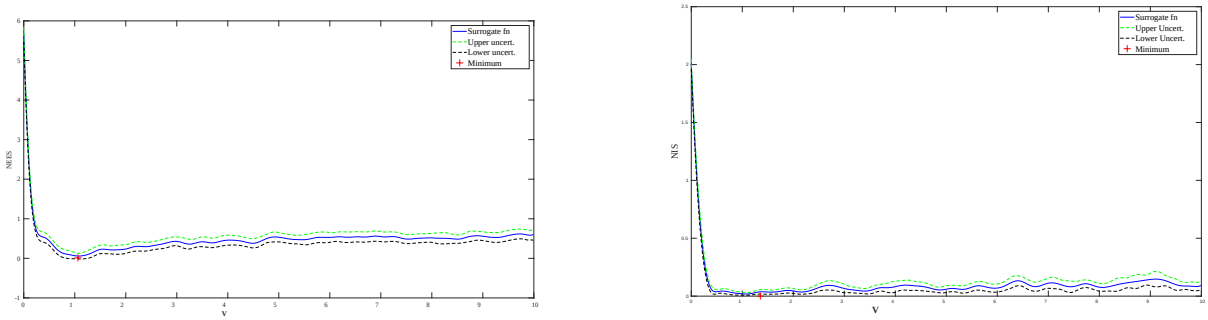


(a) iteration 0 (seed data) (b) iteration 5 (c) iteration 10



(d) iteration 15 (e) iteration 25 (f) iteration 35

Fig. 1. (a)-(f) GPBO iterations for Case 1, showing surrogate GP model (top, with sampled points, mean and 2σ bounds) and acquisition function (bottom).



(a) NEES results (b) NIS results

Fig. 2. Case 2, showing surrogate GP model with sampled points, mean and 2σ bounds after 100 iterations of GPBO: (a) NEES results; (b) NIS results.

global minimum. Fig. 4 shows estimates of the path overlaid on the ground truth path in state space, demonstrating that the two local minima are surprisingly close to one another and underscoring the need for a global optimizer that outputs information on the various optima present in parameter space.

V. Conclusions and Future Work

In this work we have developed a new approach to tuning Kalman filters which lead to optimal estimates on the filter parameters, and have demonstrated the method's success on the case of a single-dof robot in simulation. We have shown that the uncertainty estimates resulting from the use of this method are both a valuable addition to the classical optimization pipeline but also an important point of consideration for determining the dependability of its results.

In the future the authors will extend this work by turning to other estimation problems, including sensor calibration parameters for e.g. visual simultaneous localization and mapping. This will require the extension of the method to more complex linear and non-linear system models, as well as demonstrating its effectiveness with real hardware and experimental data. There is also a wealth of experimentation to be conducted in the study of other cost functions, acquisition functions, kernels, and parameterizations. Furthermore, so-called "pre-whitening" filters might be leveraged to possibly speed the convergence of GPBO to accommodate non-white noise processes, and other optimization methods besides DIRECT might improve the estimate of the global optimum once the method has reached a certain threshold. Finally, GPBO will be evaluated against alternative auto-tuning approaches, such as maximum likelihood estimation using expectation maximization [5], online adaptive noise covariance estimation, [14], [15], reinforcement learning [16], and simplex-based optimization [17].

References

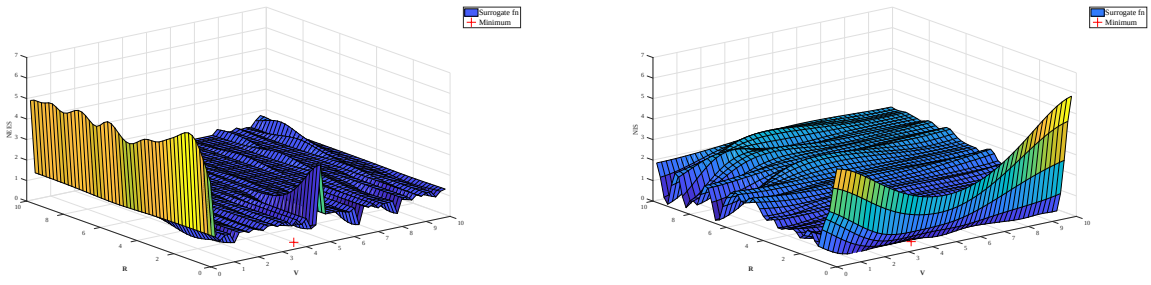
[1] R. E. Kalman and R. S. Bucy, "New results in linear filtering and prediction theory," *Journal of Basic Engineering*, vol. 83, no. 1, pp. 95–108, 1961.

[2] Y. Bar-Shalom, X. Li, and T.Kirubarajan, *Estimation with Applications to Navigation and Tracking*. New York: Wiley, 2001.

[3] R. F. Stengel, *Optimal control and estimation*. Courier Corporation, 1986.

[4] C. E. Rasmussen and C. K. I. Williams, *Gaussian processes for machine learning*, ser. Adaptive computation and machine learning. Cambridge, Mass: MIT Press, 2006.

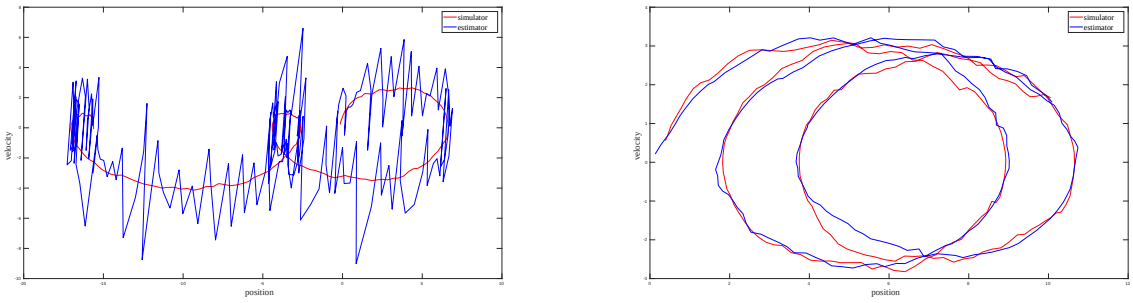
[5] C. M. Bishop, *Pattern recognition and machine learning*. New York: Springer, 2006.



(a) NEES results

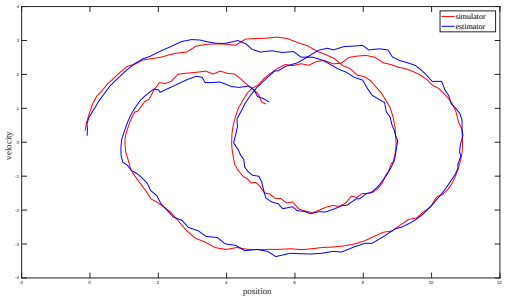
(b) NIS results

Fig. 3. Case 2, showing surrogate GP model with sampled points, mean and 2σ bounds after 100 iterations of GPBO, for both $\mathbf{V}$ and $\mathbf{R}$ values. (a) shows the NEES results, while (b) shows the NIS results.



(a) $\mathbf{R} = 4.623, \mathbf{V} = 0.001$

(b) $\mathbf{R} = 3.925, \mathbf{V} = 0.587$



(c) $\mathbf{R} = 1.063, \mathbf{V} = 1.000$

Fig. 4. Case 2 ξ vs. $\dot{\xi}$ plots of the system dynamics overlaid with the estimated path from the Kalman filter.

[6] 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, 2016.

[7] G. C. Cawley and N. L. C. Talbot, "Preventing Over-Fitting during Model Selection via Bayesian Regularisation of the Hyper-Parameters," *Journal of Machine Learning Research*, vol. 8, pp. 841–861, 2007.

[8] B. Israelsen, N. Ahmed, K. Center, R. Green, and W. B. Jr., "Adaptive simulation-based training of artificial-intelligence decision makers using Bayesian optimization," *Journal of Aerospace Information Systems*, vol. 15, no. 2, pp. 38–56, 2018.

[9] E. Brochu, V. M. Cora, and N. L. B. B. De Freitas, "A tutorial on Bayesian optimization of expensive cost functions, with application to active user modeling and hierarchical reinforcement learning," *arXiv preprint arXiv:1012.2599*, 2010.

[10] M. Hoffman, E. Brochu, and N. D. Freitas, "Portfolio Allocation for Bayesian Optimization," *Conference on Uncertainty in Artificial Intelligence*, pp. 327–336, 2011.

[11] D. R. Jones, M. Schonlau, and J. William, "Efficient Global Optimization of Expensive Black-Box Functions," *Journal of Global Optimization*, vol. 13, no. 4, pp. 455–492, 1998.

[12] R. G. Brown and P. Y. Hwang, *Introduction to random signals and applied Kalman filtering: with MATLAB exercises*. J. Wiley & Sons, 2012.

[13] R. Martinez-Cantin, "BayesOpt: A Bayesian optimization library for nonlinear optimization, experimental design and bandits," in *Journal of Machine Learning Research*, 2014, pp. 3735–3739.

[14] S. Akhlaghi, N. Zhou, and Z. Huang, "Parallel Bayesian Global Optimization of Expensive Functions," *arXiv preprint:1702.00884*, Feb 2017. [Online]. Available: https://arxiv.org/ftp/arxiv/papers/1702/1702.00884.pdf

[15] P. S. Maybeck, R. L. Jensen, and D. A. Harnly, "An adaptive extended Kalman filter for target image tracking," *IEEE Trans. on Aero. and Elec. Sys.*, vol. AES-17, no. 2, pp. 173–180, March 1981.

[16] C. Goodall and N. El-Sheimy, "Intelligent tuning of a Kalman filter using low-cost MEMS inertial sensors," in *Proceedings of 5th International Symposium on Mobile Mapping Technology (MMT07), Padua, Italy*, 2007, pp. 1–8.

[17] T. D. Powell, "Automated tuning of an extended Kalman filter using the downhill simplex algorithm," *Journal of Guidance, Control, and Dynamics*, vol. 25, no. 5, pp. 901–908, 2002.