# ONE-SHOT LEARNING OF STOCHASTIC DIFFERENTIAL EQUATIONS WITH COMPUTATIONAL GRAPH COMPLETION

MATTHIEU DARCY[1], BOUMEDIENE HAMZI[2], GIULIA LIVIERI[3], HOUMAN OWHADI[4], AND PEYMAN TAVALLALI[5]

Abstract. We consider the problem of learning Stochastic Differential Equations of the form $dX_t = f(X_t)dt + \sigma(X_t)dW_t$ from one sample trajectory. This problem is more challenging than learning deterministic dynamical systems because one sample trajectory only provides indirect information on the unknown functions $f$, $\sigma$, and stochastic process $dW_t$ representing the drift, the diffusion, and the stochastic forcing terms, respectively. We propose a simple kernel-based solution to this problem that can be decomposed as follows: (1) Represent the time-increment map $X_t \to X_{t+dt}$ as a Computational Graph in which $f$, $\sigma$ and $dW_t$ appear as unknown functions and random variables. (2) Complete the graph (approximate unknown functions and random variables) via Maximum a Posteriori Estimation (given the data) with Gaussian Process (GP) priors on the unknown functions. (3) Learn the covariance functions (kernels) of the GP priors from data with randomized cross-validation. Numerical experiments illustrate the efficacy, robustness, and scope of our method.

## 1. Introduction

The forecasting of a stochastic or a deterministic time series is a fundamental problem in, e.g., Econometrics or Dynamical Systems, which is commonly solved by learning and/or inferring a stochastic or a deterministic dynamical system model from the observed data, respectively; see, e.g., [31, 14, 15, 13, 45, 39, 2, 32, 4, 54, 30, 19, 50, 33, 22, 18, 40, 29], among many others.

### 1.1. On the kernel methods to forecasting time series.

Among the various learning-based approaches, methods based on kernels hold potential for considerable advantages over, e.g., methods based on variants of artificial neural networks (ANNs), in terms of theoretical analysis, numerical implementation, regularization, guaranteed convergence, automatization, and interpretability; see, e.g., [16, 43]. In particular, Reproducing Kernel Hilbert Spaces (RKHS) [17] have provided a strong mathematical foundations for studying dynamical systems [9, 24, 20, 23, 8, 25, 34, 35, 3, 36, 10, 11, 12, 26] and surrogate modeling (see, e.g., [49] for a survey). Yet, the accuracy of these emulators hinges on the choice of the kernel; however, the problem of selecting a good kernel has received less attention so far. Numerical experiments have recently shown that when the time series is regularly [27] or is irregularly sampled [37], simple kernel methods can successfully reconstruct the dynamics of prototypical chaotic dynamical systems when kernels are also learned from data via Kernels Flows (KF), a variant of cross-validation [42]. KF approach has then been applied to complex, large-scale systems, including geophysical data [41, 52, 53], and to learning non-parametric kernels for dynamical systems [44].

### 1.2. On the learning of Stochastic Differential Equations (SDEs).

While time series produced by deterministic dynamical systems offer a direct observation of the vector-field (i.e., of the drift) driving those systems, those produced by SDEs only present an indirect observation of the underlying drift, diffusion and stochastic forcing terms. A popular approach employed to recover the drift and the diffusion of an SDE is the so-called Kramers-Moyal expansion; see, e.g., [48, 22]. In this manuscript, we formulate the problem of learning stochastic dynamical systems described by SDEs as that of completing a computation graph [43], which represents the functional dependencies between the observed increments of the time-series and the unknown quantities. Our approach to solving this Computational Graph Completion (CGC) problem can be summarized as (1) replacing unknown functions and variables by Gaussian Processes (GPs), and (2) approximating those functions by the Maximum a Posteriori (MAP) estimator of those

GPs given available data. The covariance kernels of these GPs are learned from data via a randomized cross-validation procedure.

1.3. **Outline of the article.** Section 2 describes the problem we focus on this manuscript and our proposed solution. Section 3 describes the MAP estimator for the GPs. Section 4 describes the algorithm we propose to learn the kernels and the hyper-parameters, whereas Section 5 briefly presents the extension of the algorithm to the multivariate case. Sections 6 reports the numerical results, whereas Section 7 displays additional plots.

## 2. Statement of the problem and proposed solution

We first describe the type of SDEs used here. We consider SDEs of the form:

$$dX_t = f(X_t)dt + \sigma(X_t)dW_t \tag{1}$$

with initial condition $X_0 = x_0$. In the previous equation, $(W_t)_{t\in[0,T]}$ denotes a Wiener process. We assume that the process in Equation (1) is observed at discrete times $t_n$, $n = 1 \dots N$, such that the time intervals $\Delta t_n := (t_{n+1} - t_n)$ between observations $X_n := X_{t_n}$ of the time series are small enough so that the following approximation holds:

$$X_{n+1} = X_n + f(X_n)\Delta t_n + \sigma(X_n)\sqrt{\Delta t_n}\xi_n + \varepsilon_n\,, \tag{2}$$

where the *i.i.d.* random variables $\xi_n \overset{d}{\sim} \mathcal{N}(0,1)$ represent Brownian Motion increments and the *i.i.d.* random variables $\varepsilon_n \overset{d}{\sim} \mathcal{N}(0,\lambda)$ represent discretization noise/misspecification; henceforth, the notation " $\overset{d}{\sim}$ " stands for "distributed as". We seek to recover/approximate the unknown functions $f$ and $\sigma$ from the data $(\boldsymbol{X}, \boldsymbol{Y}) = \{(X_n, Y_n)\}_{1\le n\le N}$, where

$$Y_n := X_{n+1} - X_n\,.$$

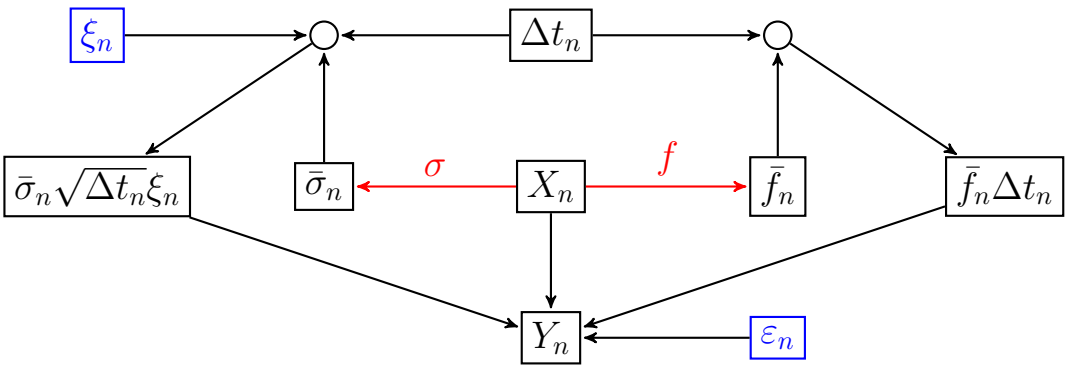
Therefore, the relation between $X_n$ and $Y_n$ is given by our modeling assumption:

$$Y_n = f(X_n)\Delta t_n + \sigma(X_n)\sqrt{\Delta t_n}\xi_n + \varepsilon_n\,. \tag{3}$$

2.1. **The Computational Graph Completion problem.** In general, a computational graph is defined as a graph representing functional dependencies between a finite number of (not necessarily random) variables and functions. We will use nodes to represent variables and arrows to represent functions. We will color known functions in black and unknown functions in red. Random variables are drawn in blue and primary variables as squares. We will distinguish nodes used to aggregate variables by drawing them as circles. Multiple incoming arrows into a square node are interpreted as a sum. Now, let $\bar{f}_n := f(X_n)$ and $\bar{\sigma}_n := \sigma(X_n)$ be two intermediate (unobserved) variables. Equation (3) can thus be rewritten as:

$$Y_n = \bar{f}_n\Delta t_n + \bar{\sigma}_n\sqrt{\Delta t_n}\xi_n + \varepsilon_n. \tag{4}$$

In particular, it can be represented as the following computational graph:




We formulate now the learning problem in this manuscript as the problem of completing the just displayed computational graph; see [43]. Let $X_1, \dots X_N$, $Y_1, \dots, Y_N$ and $\Delta t_1, \dots, \Delta t_N$ be the $N$ observations data: our goal is to approximate the unknown functions $f$ and $\sigma$ from these

observations. In order to solve this problem, we will use the GP framework: we replace $\sigma$ and $f$ by GPs and approximate them via MAP estimation given the data. More precisely, we assume that $f$ and $\sigma$ are mutually independent GPs, with centered Gaussian priors $f \overset{d}{\sim} \mathcal{GP}(\boldsymbol{O}, \boldsymbol{K}), \sigma \overset{d}{\sim} \mathcal{GP}(\boldsymbol{O}, \boldsymbol{G})$ defined by the covariance functions/kernels $\boldsymbol{K}$ and $\boldsymbol{G}$.

## 3. MAP estimator

Write $\bar{f}$ for the vector with entries $\{\bar{f}_n\}_{1\leq n\leq N}$ and $\bar{\sigma}$ for the vector with entries $\{\bar{\sigma}_n\}_{1\leq n\leq N}$. Observe that given $\bar{\sigma}$ and $\bar{f}$, the identification of the functions $f$ and $\sigma$ reduces to two separate simple kernel regression problems. We will therefore first focus on the estimation of $\bar{f}$ and $\bar{\sigma}$. Since $f$ and $g$ are independent, $\bar{f} = f(X)$ and $\bar{\sigma} = \sigma(X)$ are conditionally (on $X$) independent. We deduce that

$$p(\bar{f}, \bar{\sigma}|Y, X) = p(Y|\bar{f}, \bar{\sigma}, X)\frac{p(\bar{f}|X)p(\bar{\sigma}|X)}{p(Y|X)}\,.$$

It follows that a MAP estimator of $(\bar{f}, \bar{\sigma})$ is a minimizer of the loss

$$\begin{aligned}\mathcal{L}_1(\bar{f}, \bar{\sigma}) &:= -\ln\big(p(Y|X, \bar{f}, \bar{\sigma})p(\bar{f}|X)p(\bar{\sigma}|X)\big)\\ &= (Y - \Lambda\bar{f})^T(\Sigma + \lambda I)^{-1}(Y - \Lambda\bar{f}) + \bar{f}^T K(X, X)^{-1}\bar{f}\\ &+ \sum\nolimits_{n=1}^{N} \ln(\bar{\sigma}_n^2\Delta t_n) + \bar{\sigma}^T G(X, X)^{-1}\bar{\sigma}\,.\end{aligned} \tag{5}$$

where $K(X, X)$ is the $N \times N$ matrix with entries $K(X_i, X_j)$, $G(X, X)$ is the $N \times N$ matrix with entries $G(X_i, X_j)$, $\Sigma$ is the diagonal $N \times N$ matrix with diagonal entries $\bar{\sigma_n}^2\Delta t_n$, and $\Lambda$ is the diagonal $N \times N$ matrix with diagonal entries $\Delta t_n$.

### 3.1. Recovery of $f$.

First, observe that given $\bar{\sigma}$, (5) is quadratic in $\bar{f}$ and its minimizer in $\bar{f}$ is

$$\bar{f}^* = K(X, X)(K(X, X) + \Sigma + \lambda I)^{-1}Y\,. \tag{6}$$

Therefore $f \overset{d}{\sim} \mathcal{GP}(\boldsymbol{O}, \boldsymbol{K})$ conditioned on $(X, f(X) = \bar{f})$ is is normally distributed with (conditional) mean

$$f^*(x) := K(x, X)(K(X, X) + \Sigma + \lambda I)^{-1}Y\,, \tag{7}$$

and (conditional) covariance

$$K(x, x) - K(x, X)(K(X, X) + \Sigma + \lambda I)^{-1}K(X, x)\,. \tag{8}$$

We therefore estimate $f$ with $f^* =$(7). Note that (7) and (8) can also be recovered by observing that given $\bar{\sigma}$, Equation (4) corresponds to a noisy regression problem, with noise coming from two independent Gaussian variables. This is proved in Appendix A and it is an easy modification of the proof presented in [7, Page 306]

### 3.2. Recovery of $\sigma$.

Taking $\bar{f} = \bar{f}^* =$(7) in (5), the estimation of $\bar{\sigma}$ reduces to the minimization of the loss

$$\mathcal{L}_2(\bar{\sigma}) := (Y - \Lambda\bar{f}^*)^T(\Sigma + \lambda I)^{-1}(Y - \Lambda\bar{f}^*) + \sum\nolimits_{n=1}^{N} \ln(\bar{\sigma}_n^2\Delta t_n) + \bar{\sigma}^T G(X, X)^{-1}\bar{\sigma}\,. \tag{9}$$

Write $\bar{\sigma}^\dagger$ for a minimizer of (9) obtained through a numerical optimization algorithm (e.g., gradient descent). Because the numerical approximation of $\bar{\sigma}^\dagger$ is noisy, we then further estimate $\bar{\sigma}$ as the mean of the Gaussian vector $\sigma(X)$ conditioned on $\sigma(X) = \bar{\sigma}^\dagger + Z$ where the entries $Z_i$ of the (noise) vector $Z$ are *i.i.d.* Gaussian with variance $\gamma$. We therefore approximate $\bar{\sigma}$ with

$$\bar{\sigma}^* = \mathbb{E}[\sigma(X)|\sigma(X) + Z = \bar{\sigma}^\dagger] = G(X, X)(G(X, X) + \gamma I)^{-1}\sigma^\dagger. \tag{10}$$

and $\sigma$ with

$$\sigma^*(x) = G(x, X)(G(X, X) + \gamma I)^{-1}\bar{\sigma}^\dagger\,.$$

3.3. **Alternating minimization.** The natural approach to identification of $\bar{\sigma}^\dagger$ is to minimize (with respect to $\bar{\sigma}$) (9) with $\bar{f}^*$ defined as the function (6) of $\bar{\sigma}$. In our numerical experiments, we employ an alternating minimization approach and alternate between minimizing (5) with respect to $\bar{f}$ and $\bar{\sigma}$: we alternate between identifying $\bar{f}^* = (6)$ and minimizing (9) with fixed $\bar{f}^*$.

## 4. Learning the kernels and the hyperparameters

The accuracy of our approach depends on the on the kernels $K, G$ and the hyperparameters $\lambda, \gamma$ of the observation noise and MAP recovery noise respectively. We select the kernels $K, G$ in a family of kernels parameterized by $\theta_k, \theta_g$, which we learn from data. Writing

$$\boldsymbol{\theta} := (\theta_k, \theta_g, \lambda, \gamma)\,, \tag{11}$$

for the vector formed by all the hyperparameters of our approach, we learn $\theta$ through a robust-learning cross-validation approach which we will now describe. Consider the set of all sets of possible partitions of the training data $\mathcal{D}_\mathcal{T}$ with indices $\mathcal{I}$ into two mutually disjoint subsets of equal size

$$\mathcal{A} = \{(\Pi, \Pi^c) | \Pi \cup \Pi^c = \mathcal{I}, \Pi \cap \Pi^c = \emptyset, |\Pi| = \frac{|\mathcal{I}|}{2}\}. \tag{12}$$

Write $\mathcal{D}_\Pi = (x_j, y_j)_{j\in\Pi}$ for the set of points belonging to the first partition and $\mathcal{D}_{\Pi^c} = (x_j, y_j)_{j\in\Pi^c}$ for the set of points belonging to the second set. Write $\mathbb{E}_{(\Pi,\Pi^c)}$ for the uniform distribution over $\mathcal{A}$. For $(\Pi, \Pi^c) \in \mathcal{A}$ write $\mathcal{L}_1(\mathcal{D}_{\Pi^c}, \bar{f}, \bar{\sigma})$ for the MAP loss (5) calculated with dataset $\mathcal{D}_{\Pi^c}$. Write

$$\begin{aligned}\mathcal{L}_3(\boldsymbol{\theta}; \bar{f}^*, \bar{\sigma}^*, \mathcal{D}_\Pi) &= -\ln p(Y_\Pi | \bar{f}^*, \bar{\sigma}^*, X_\Pi) \\ &= \sum\nolimits_i \frac{(Y_i - \bar{f}^*)^2}{2(\bar{\sigma}_i^*)^2} + \frac{1}{2}\ln(\bar{\sigma}_i^*,),\end{aligned} \tag{13}$$

for the negative log likelihood of the validation data $\mathcal{D}_\Pi$ given $\bar{f}^*, \bar{\sigma}^*$. Our proposed cross-validation approach is then to select $\boldsymbol{\theta}^*$ as

$$\begin{aligned}&\boldsymbol{\theta}^* = \arg\min_{\boldsymbol{\theta}} \mathbb{E}_{(\Pi,\Pi^c)}\{\mathcal{L}_3(\boldsymbol{\theta}; \bar{f}^*, \bar{\sigma}^*, \mathcal{D}_\Pi\} \\ \text{subject to}\quad &\bar{f}^*, \bar{\sigma}^* = \arg\min_{\bar{f},\bar{\sigma}} \mathcal{L}_1(\mathcal{D}_{\Pi^c}, \bar{f}, \bar{\sigma})\end{aligned} \tag{14}$$

Note that $\bar{f}^*, \bar{\sigma}^*$ are selected as described in Section 3. In practice, it is intractable to compute $\mathbb{E}_{(\Pi,\Pi^c)}\{\mathcal{L}_3(\boldsymbol{\theta}; \bar{f}^*, \bar{\sigma}^*, \mathcal{D}_\Pi\}$ exactly. We instead approximate $\mathbb{E}_{(\Pi,\Pi^c)}\{\mathcal{L}_3\}$ with the empirical average

$$\frac{1}{M}\sum\nolimits_{i=1}^M \mathcal{L}_3(\boldsymbol{\theta}; \bar{f}_i^*, \bar{\sigma}_i^*, \mathcal{D}_{\Pi_i}) \tag{15}$$

where the $(\Pi_i, \Pi_i^c)$ are i.i.d. samples from $\mathbb{P}_{(\Pi,\Pi^c)}$ and $\bar{f}_i^*, \bar{\sigma}_i^* = \arg\min_{\bar{f},\bar{\sigma}} \mathcal{L}_1(\mathcal{D}_{\Pi_i^c}, \bar{f}, \bar{\sigma})$. We use a gradient free optimization algorithm to minimize (37) (see Subsection 4.1). This algorithm only uses noisy observations (15) of the true loss.

The proposed cross-validation algorithm can be summarized as follows:

(1) Select a gradient-free optimization algorithm.
(2) At each iteration, given the hyperparameters $\boldsymbol{\theta}^k$, select $M$ divisions of the data $\mathcal{D}_\mathcal{T}$ into a training set $\mathcal{D}_{\Pi_i^c}$ and a validation set $\mathcal{D}_{\Pi_i}$.
(3) For each $1 \le i \le M$, recover $\bar{f}_i^*, \bar{\sigma}_i^*$ using training data $\mathcal{D}_{\Pi_i^c}$ using hyper-paramters $\boldsymbol{\theta}^k$.
(4) For each $1 \le i \le M$, compute the loss $\mathcal{L}_3(\boldsymbol{\theta}^k; \bar{f}_i^*, \bar{\sigma}_i^*, \mathcal{D}_{\Pi_i})$ and the empirical average (15).
(5) Minimize (15) with the gradient free optimization algorithm to select $\boldsymbol{\theta}^{k+1}$.

4.1. **The active learning algorithm.** We choose the Bayesian optimization algorithm [51], where the loss function is modeled using a Gaussian Process with Matern kernel [47], implemented in the scikit-optimize library in Python [1].

## 5. Multivariate case

We now examine the case where the SDE is multidimensional. In such a case

$$d\boldsymbol{X}_t = \boldsymbol{f}(\boldsymbol{X_t})dt + \boldsymbol{\sigma}(\boldsymbol{X_t})d\boldsymbol{W}_t \tag{16}$$

where $\boldsymbol{X}_t \in \mathbb{R}^d$, $\boldsymbol{f} : \mathbb{R}^d \mapsto \mathbb{R}^d, \boldsymbol{\sigma} : \mathbb{R}^d \mapsto \mathbb{R}^{d\times m}$ and $\boldsymbol{W}_t = (W_t^i)_{i\leq m}$ is a *standard* multivariate Wiener process.

For simplicity, we place independent GP priors on each dimension, which is equivalent to using a matrix-valued kernel with diagonal entries:

$$K(x,y) = \begin{pmatrix} K_1(x,y) & 0 \\ 0 & K_2(x,y) \end{pmatrix} \tag{17}$$

More generally, we may consider matrix valued kernels with non zero off-diagonals, but the use of diagonal matrices is practical because it makes all optimization problem independent for each dimension. Then the problem leads to $d$ equations:

$$dX_t^i = f(\boldsymbol{X_t})^i dt + \sum\nolimits_{j=1}^{d} \sigma_{i,j}(\boldsymbol{X_t})dW_t^j \tag{18}$$

which can all be optimized independently. The optimal value of $\bar{f}(\boldsymbol{X_t})$ conditioned on $\bar{\boldsymbol{\sigma}}$ is given by the posterior mean equation:

$$\bar{\boldsymbol{f}}(\boldsymbol{X}_n)^i = \mathbb{E}[\boldsymbol{f}(\boldsymbol{X}_n)^i] = K^i(\boldsymbol{X_n}, \boldsymbol{X})(K^i(\boldsymbol{X},\boldsymbol{X}) + \Sigma^i + \lambda I)^{-1}\boldsymbol{Y}^i \tag{19}$$

where $\Sigma^i = \sum_j \Sigma^j$ is the sum of the (diagonal) matrices with entries $\Sigma^j_{k,k} = (\bar{\boldsymbol{\sigma}}(\boldsymbol{X}_k)^j)^2\Delta t_k, 1 \leq k \leq N, 1 \leq j \leq d$. On the other hand, the MAP estimate for can be computed jointly for all entries of the matrix $\boldsymbol{\sigma}^i$ with entries $\sigma_k^i$ as

$$\arg\min_{\boldsymbol{\sigma}^i} (\boldsymbol{Y^i} - \Lambda \boldsymbol{f}(\boldsymbol{X})^i)^T(\Sigma^i + \lambda I)^{-1}(\boldsymbol{Y^i} - \Lambda \boldsymbol{f}(\boldsymbol{X})^i) + \sum\nolimits_{k=1}^{N} \ln(\Delta t_k \sum\nolimits_{i=1}^{d} (\sigma_k^i)^2) + \sum\nolimits_{i=1}^{d} \boldsymbol{\sigma^i}^T G^i(\boldsymbol{X},\boldsymbol{X})^{-1}\boldsymbol{\sigma^i}. \tag{20}$$

Additional assumptions can be made regarding the structure of the noise values, such as assuming that the noise is separate across all dimensions (i.e., $\boldsymbol{\sigma}$ is a diagonal matrix) or that the diffusion function $\boldsymbol{\sigma}$ is identical across all dimensions (i.e., $\boldsymbol{\sigma}$ is a diagonal matrix with identical values along the diagonals).

## 6. Numerical experiments.

In this section, we test our methodology on some toy example SDEs. Our numerical results include comparisons with an initial guess of parameters and parameters obtained through negative log marginal likelihood minimization. We test the proposed methodology on the following one dimensional SDEs:

$$dX_t = (b - X_t)dt + \sigma dW_t \qquad \text{Vasicek model/OU process} \tag{21}$$

$$dX_t = (b - X_t)dt + \sigma\sqrt{X_t}dW_t \qquad \text{CIR model} \tag{22}$$

$$dX_t = b\sin(X_t)dt + \sigma|X_t|dW_t \qquad \text{Noisy sinusoidal} \tag{23}$$

$$dX_t = \mu X_t dt + \sigma X_t dW_t \qquad \text{Geometric Brownian motion (GBM).} \tag{24}$$

In all cases except the noisy sinusoidal, we discretize the dynamics with a constant time step $\Delta t = 0.1$. For the noisy sinusoidal, $\Delta t = 1$. We use 100 points for training and 100 points for testing.

We also apply our methodology to a two dimensional, fast-slow SDE of the form

$$\begin{aligned} dX_t &= \frac{1}{\varepsilon}f(X_t, Y_t)dt + \sigma_x \frac{F(X_t,Y_t)}{\sqrt{\varepsilon}}dW_t \\ dY_t &= g(X_t,Y_t)dt + \sigma_y G(X_t,Y_t)dB_t \end{aligned} \tag{25}$$

In this case, we generate a trajectory of 10000 steps with $\Delta t = 0.001$. We extract a sub-sequence of length 1000, which is split into a training set of 500 points and a testing set of 500 points. In all cases, $\lambda$ is set to a very small value ($1 \times 10^{-10}$) as our modeling assumption is accurate for these toy models. In all cases, we train on a single trajectory and predict future unobserved points.

6.1. **Benchmarks.** We compare our method and optimized parameters with two baselines. A first baseline that uses our method and unoptimized parameters; this method is labeled as "initial guess". The second baseline does not use our method to recover the drift and diffusion separately but instead uses standard gaussian process regression with a white noise kernel. The posterior distribution of the GP conditioned on the data provides a prediction for the mean (the drift of the SDE) and the variance (the volatility of the sde). The parameters of the kernel are optimized through the minimization of the negative log marginal likelihood. This method is labeled as "log marginal method" and uses the implementation present in [46]. The details are presented in Appendices A and B. Note that a major drawback of this method is the assumption that the noise $\sigma(X_t)$ is identically distributed Gaussian noise, modeled through the white noise kernel $\delta(x_i - x_j)$. Such an assumption is incorrect for some types of SDEs.

6.2. **Metrics.** To measure the performance of each method, we use three metrics. The first is the likelihood of the model given the data of the test set defined as

$$\mathcal{L}(\mathcal{M}|(X,Y)) = -\log(p(Y|\mathcal{M},X)) \propto \sum\nolimits_i^K \frac{(Y_i - \bar{f}_i)^2}{2\bar{\sigma}_i^2} + \frac{1}{2}\log(\bar{\sigma}_i^2).$$

The other two metrics are the relative error of the test drift and volatility at the test points:

$$\delta_f = \frac{||f - \bar{f}||}{||f||}$$
$$\delta_\sigma = \frac{||\sigma - \bar{\sigma}||}{||\sigma||}$$

where $f$ is the vector of drift values at the test points $(f)_i = f(X_i)$ and $\bar{f}$ is the vector of prediction $(\bar{f})_i = \bar{f}(X_i)$ (likewise for $\sigma, \bar{\sigma}$). Note that in practice, only $\mathcal{L}(\mathcal{M}|(X,Y))$ may be observed. We present $\delta_f, \delta_\sigma$ to illustrate how a lower loss on the recovery of the drift and volatility yield a lower loss on the likelihood.

6.3. **Choice of kernels.** In most cases, we opt for the squared exponential kernel:

$$K(x_i, x_j) = \alpha^2 \exp\left(-\frac{||x_i - x_j||^2}{l^2}\right).$$

Such a kernel is parametrized by two parameters, the variance $\alpha^2$ and the length scale $l$. Observe that the mean of the predicted values $\bar{f}_i$ do not depend on $\alpha$, equation (19). Hence for the drift function $\bar{f}$, such a parameter does not affect the prediction and without loss of generality we set $\alpha = 1$. For the diffusion function $\bar{\sigma}$, while the predictive mean, Equation (10), does not depend on $\alpha$, the recovered values $\sigma_i$ do depend on such a parameter, Equation (9). One can think of such a value as indicating the strenght of the prior: as $\alpha \to \infty$, the prior term goes to 0 in Equation (9), leading to a negligible prior. On the other hand, a small $\alpha$ leads to a stronger prior. Therefore $\alpha$ is optimized for the diffusion function alone.

Thus, the hyper-parameters are

$$\boldsymbol{\theta} = (l_f, \alpha_\sigma, l_\sigma, \gamma)$$

and we set the search space to be

$$l_f, l_\sigma \in (0.1, 100)$$
$$\alpha_\sigma \in (1.0, 10.0)$$
$$\gamma \in (10^{-8}, 10^1).$$

6.4. **Ornstein–Uhlenbeck process.** The discrete OU process is defined as:

$$X_{n+1} - X_n = (b - X_n)\Delta t + \sigma\sqrt{\Delta t}\xi_n, \quad X_0 = x_0 \tag{26}$$

where $\xi_n \sim \mathcal{N}(0,1)$. We simulate two trajectories with parameters: $b = 10, \sigma = 2.0$ and initial conditions $x_0 = 1.0, 8.0$ respectively. The first trajectory is presented in figure 1. The results of both methods are recorded in table 1 with the MSE on the train set in parenthesis and the best results on the test set highlighted. The prediction on the first trajectory are presented in Figures 2 and 3 (see also Figures 18 and 19 in Section 7).

| | Trajectory 1 | | | Trajectory 2 | | |
|---|---|---|---|---|---|---|
| | $\mathcal{L}(\mathcal{M}\|Y)$ | $\delta_f$ | $\delta_\sigma$ | $\mathcal{L}(\mathcal{M}\|Y)$ | $\delta_f$ | $\delta_\sigma$ |
| Initial guess | 21.666 | 1.178 | **0.241** | 3.928 | 0.922 | 0.241 |
| Log marginal method | 8.388 | 0.578 | 0.0245 | **-1.418** | 0.789 | **0.105** |
| Our method | **7.820** | **0.378** | 0.0657 | 4.794 | **0.472** | 0.241 |

TABLE 1. Results on the OU process.


FIGURE 1. Trajectory 1 of the OU process: trajectory (left) and increments (right)

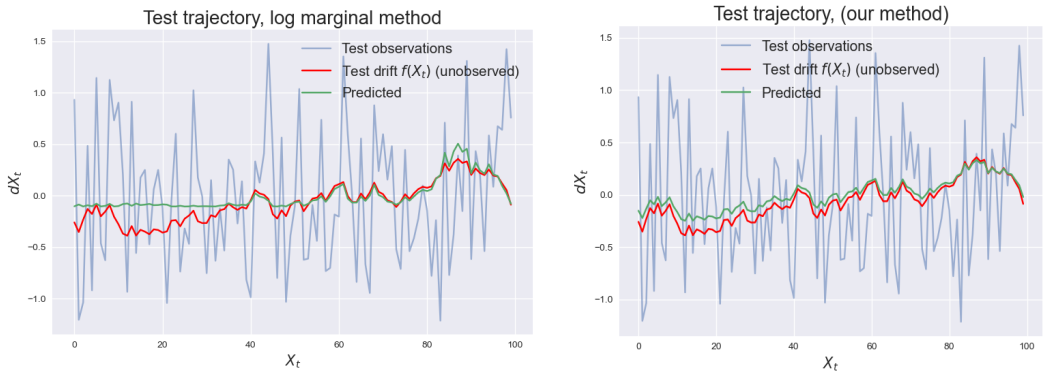



FIGURE 2. OU process, prediction of the drift on trajectory 1: marginal likelihood (left) and our method (right).

6.5. **CIR model.** The CIR model is similar to the OU process but with a non-constant and non-linear volatility:

$$X_{n+1} - X_n = (b - X_n)\Delta t + \sigma\sqrt{X_n}\sqrt{\Delta t}\xi_n, \quad X_0 = x_0$$

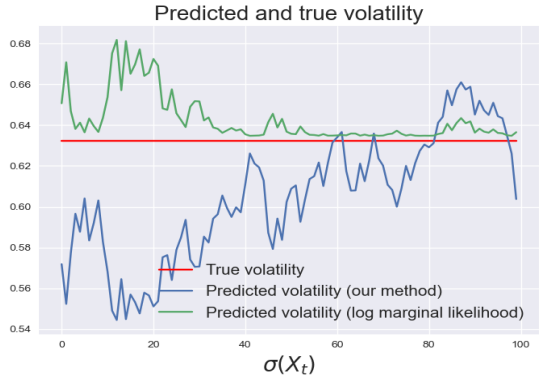



FIGURE 3. OU process, prediction of the volatility on trajectory 1.

where $\xi_n \overset{d}{\sim} \mathcal{N}(0,1)$. We simulate one trajectories with parameters: $b = 10, 8.0, \sigma = 0.2, 0.1$ and initial condition $x_0 = 9.0, 8.5$. The results of both methods are recorded in table 2 with the best results on the test set highlighted. The first trajectory with increments is presented in Figure 4 with the predicted drift and diffusion of both methods presented in Figures 5 and 6 (see also Figures 20 and 21 in the Section 7).

| | Trajectory 1 | | | Trajectory 2 | | |
|---|---|---|---|---|---|---|
| | $\mathcal{L}(\mathcal{M}\|Y)$ | $\delta_f$ | $\delta_\sigma$ | $\mathcal{L}(\mathcal{M}\|Y)$ | $\delta_f$ | $\delta_\sigma$ |
| Initial guess | 248.50 | 0.849 | 1.700 | -76.717 | 2.197 | 3.675 |
| Log marginal method | -105.172 | 1.000 | 0.0332 | -116.044 | 1.000 | 0.0332 |
| Our method | **-105.903** | **0.825** | **0.0317** | **-116.497** | **0.896** | **0.0283** |

TABLE 2. Results on the CIR process.


FIGURE 4. Trajectory of the CIR process: trajectory (left) and increments (right)

6.6. **Sinusoidal SDE.** The noisy sinusoidal has dynamics governed by

$$X_{n+1} - X_n = b\sin(X_n)\Delta t + \sigma|X_n|\sqrt{\Delta t}\xi_n, \quad X_0 = x_0 \tag{27}$$

where $\xi_n \sim \mathcal{N}(0,1)$. We simulate two trajectories with parameters: $b = 2.0, 1.5, \sigma = 0.2$ and initial conditions $x_0 = 0.1, 0.2$ respectively. The results are presented in Table 3 with the best results highlighted. The first trajectory with increments is presented in Figure 7 with the predicted drift and diffusion of both methods presented in Figures 8 and 9.

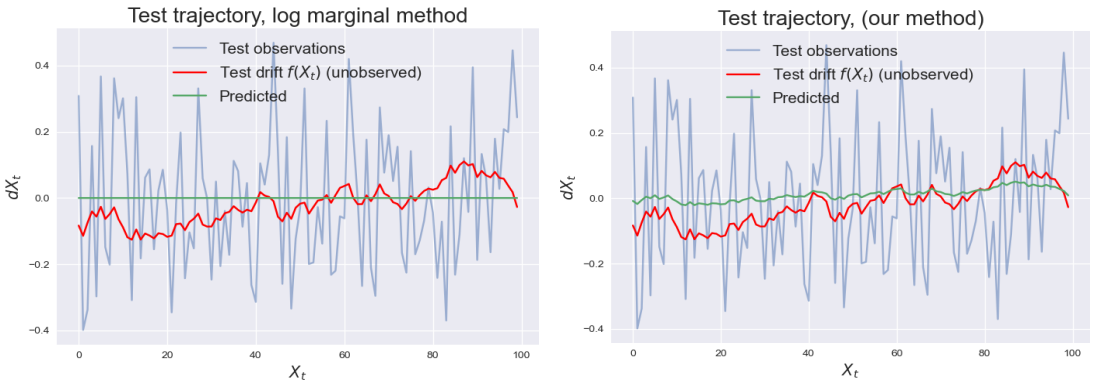



FIGURE 5. CIR process, prediction of the drift on trajectory 1: marginal likelihood (left) and our method (right)

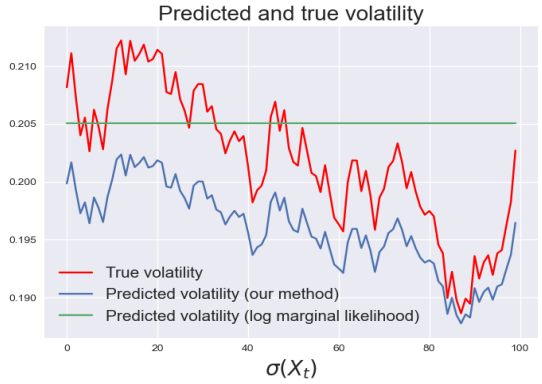



FIGURE 6. CIR process, prediction of the volatility on trajectory 1. The basic Gaussian process method has a constant volatility whereas our method is able to model an evolving volatility function.

| | Trajectory 1 | | | Trajectory 2 | | |
|---|---|---|---|---|---|---|
| | $\mathcal{L}(\mathcal{M}\|Y)$ | $\delta_f$ | $\delta_\sigma$ | $\mathcal{L}(\mathcal{M}\|Y)$ | $\delta_f$ | $\delta_\sigma$ |
| Initial guess | 11.553 | 0.455 | 53.981 | 53.981 | 0.326 | 0.304 |
| Log marginal method | **3.629** | **0.0481** | **0.251** | 4.205 | 0.112 | 0.209 |
| Our method | 8.185 | 0.0743 | 0.255 | **2.900** | **0.092** | **0.200** |

TABLE 3. Results on the Sinusoidal process.

6.7. **Geometric Brownian Motion.** The (discrete) GBM is generated by

$$X_{n+1} - X_n = \mu X_n \Delta t + \sigma X_n \sqrt{\Delta t}\xi_n, \quad X_0 = x_0. \tag{28}$$

We generate two trajectories, with parameters $\mu = 10.0, \sigma = 3.0, 1.5$, $x_0 = 1.0$ and $\Delta t = 0.001$. For this test, we use two linear kernels of the form

$$K(x, y) = \langle x, y \rangle + a \tag{29}$$

for both the drift and diffusion. The results for both trajectories are reported in table 4 with the predicted drift and diffusion of trajectory 2 presented in Figures 10, 11, 12 (see Figures 24, 11, 12 in the Appendix for trajectory 1).

6.8. **Fast Slow SDE.** We consider the Van der Pol oscillator as as an example of a fast slow SDE (25).

$$\begin{aligned} dX_t &= \frac{1}{\varepsilon} f(X_t, Y_t)dt + \sigma_x \frac{F(X_t, Y_t)}{\sqrt{\varepsilon}} dW_t \\ dY_t &= g(X_t, Y_t)dt + \sigma_y G(X_t, Y_t)dB_t \end{aligned} \tag{30}$$


FIGURE 7. Trajectory of the Sinusoidal SDE: trajectory (left) and increments (right)

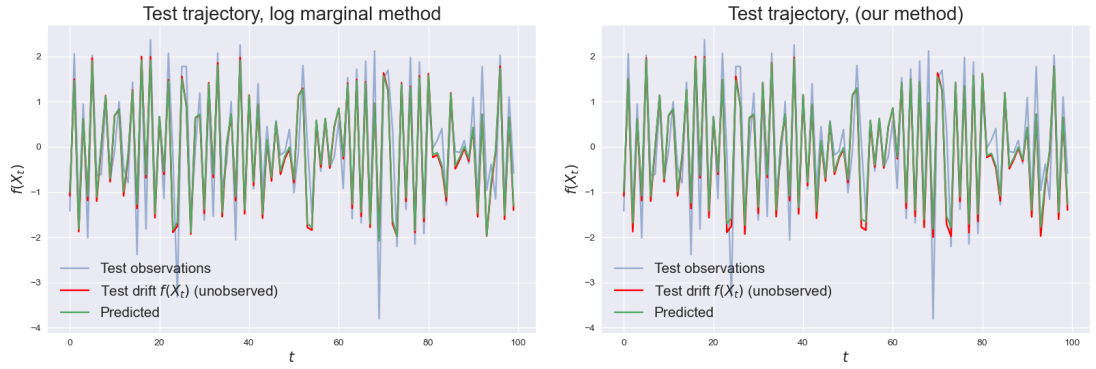



FIGURE 8. Sinusoidal process, drift prediction trajectory 1: marginal likelihood method (left) and our method (right)

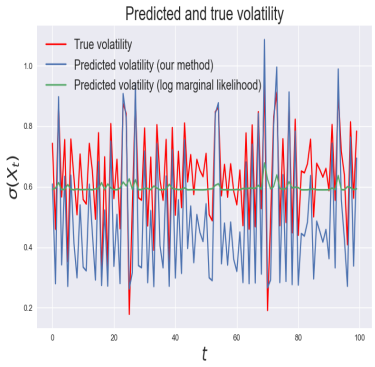



FIGURE 9. Sinusoidal process, volatility prediction on trajectory 1.

| | Trajectory 1 | | | Trajectory 2 | | |
|---|---|---|---|---|---|---|
| | $\mathcal{L}(\mathcal{M}\|Y)$ | $\delta_f$ | $\delta_\sigma$ | $\mathcal{L}(\mathcal{M}\|Y)$ | $\delta_f$ | $\delta_\sigma$ |
| Initial guess | 4.537 | 2.887 | 0.0271 | 67.691 | 0.429 | 0.302 |
| Log marginal method | 8393.935 | **1.006** | 0.917 | 445.721 | 0.379 | 0.655 |
| Our method | **4.444** | 2.873 | **0.0234** | **4.536** | **0.295** | **0.0106** |

TABLE 4. Results on the GBM process with the linear kernel.

6.8.1. *Van der Pol oscillator 1.* We first consider the case where

$$\begin{aligned} f(x,y) &= y - \frac{27}{4}x^2(x+1) \\ g(x,y) &= -\frac{1}{2} - x \\ F &= G = 1 \\ \varepsilon &= 0.01, \sigma_x = \sigma_y = 0.1 \end{aligned} \tag{31}$$


FIGURE 10. Trajectory of the second trajectory of GBM: trajectory (left) and increments (right)

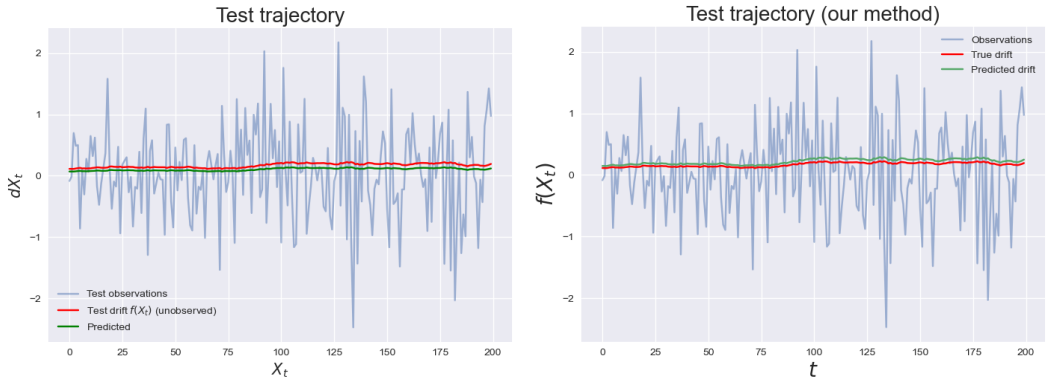



FIGURE 11. GBM process, drift prediction trajectory 2: marginal likelihood method (left) and our method (right)

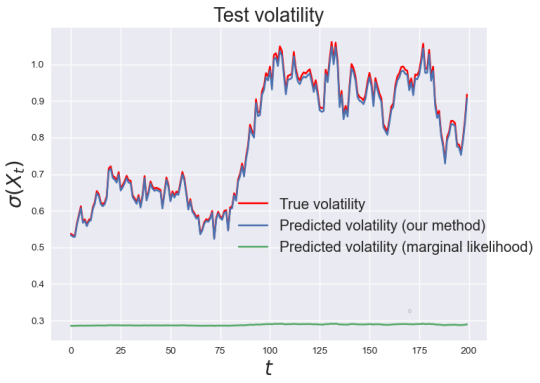



FIGURE 12. GBM process, volatility prediction on trajectory 2.

In this case, there is only one source of noise which is constant. We model this system using 4 SE kernels, 2 for each dimension, one per drift and diffusion. The results are presented in table 5. The trajectory is presented in Figure 13 and the drift and volatility predictions are presented in Figures 14 and 15.

6.8.2. *Van der Pol oscillator 2.* The second Van der Pol oscillator we consider is given by

| | $X_t$ | | | $Y_t$ | | |
|---|---|---|---|---|---|---|
| | $\mathcal{L}(\mathcal{M}\|Y)$ | $\delta_f$ | $\delta_\sigma$ | $\mathcal{L}(\mathcal{M}\|Y)$ | $\delta_f$ | $\delta_\sigma$ |
| Initial | **-1410.720** | 0.262 | 0.538 | -10.210 | 0.295 | 309.185 |
| Log marginal | -1383.57 | 1.0171 | **0.188** | **-2601.48** | **0.243** | **0.00563** |
| Our method | -1288.90 | **0.180** | 0.343 | -2596.06 | 0.357 | 0.0819 |

TABLE 5. Results on the stochastic Van der Pol 1.

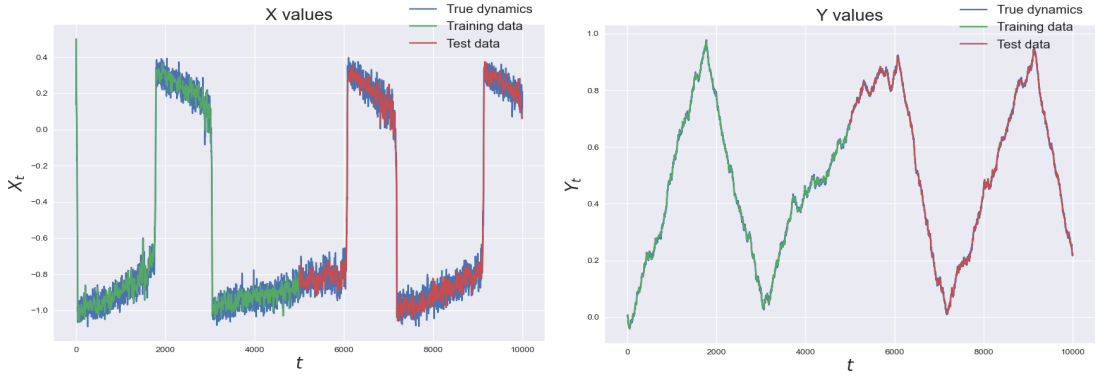



FIGURE 13. Stochastic Van der Pol 1.

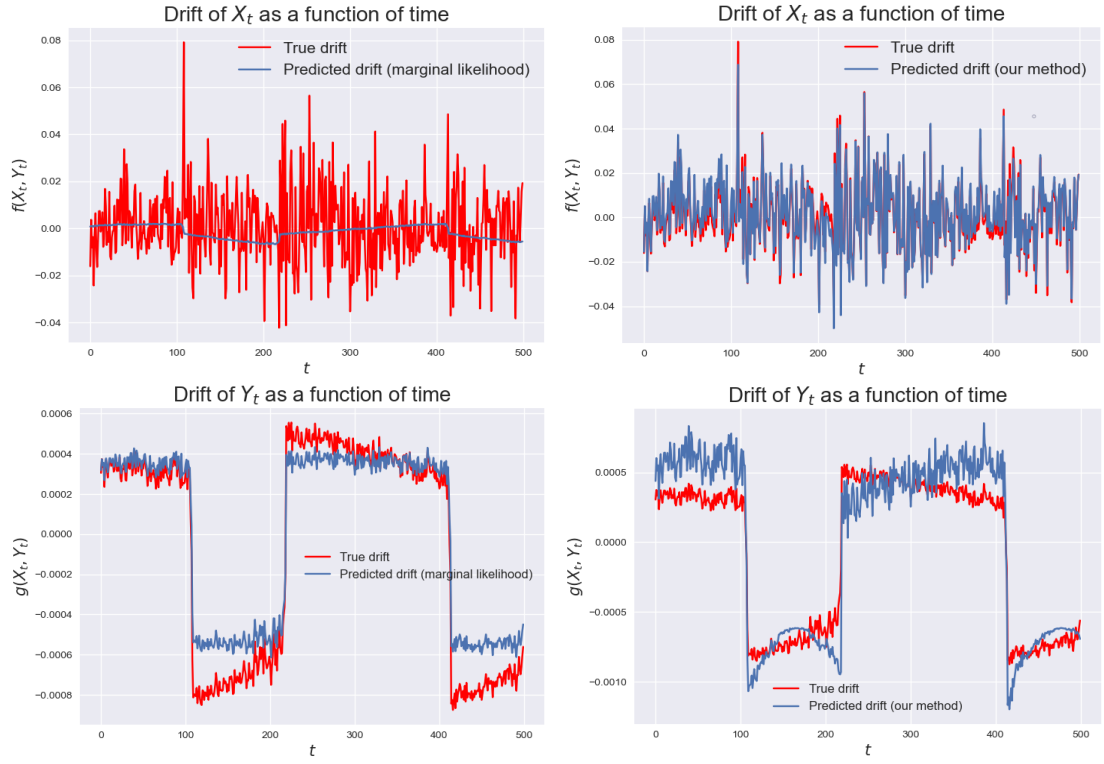



FIGURE 14. Stochastic Van der Pol 1: predictions on the drift (drift $X_t$ top, drift $Y_t$ bottom): comparison between maximizing the marginal likelihood (left) and our method (right).

$$
\begin{aligned}
f(x,y) &= y - \frac{27}{4}x^2(x+1) \\
g(x,y) &= -\frac{1}{2} - x \\
F(x,y) &= \sin(x) \\
G(x,y) &= \cos(y) \\
\varepsilon &= 0.01, \sigma_x = \sigma_y = 0.05.
\end{aligned}
\tag{32}
$$

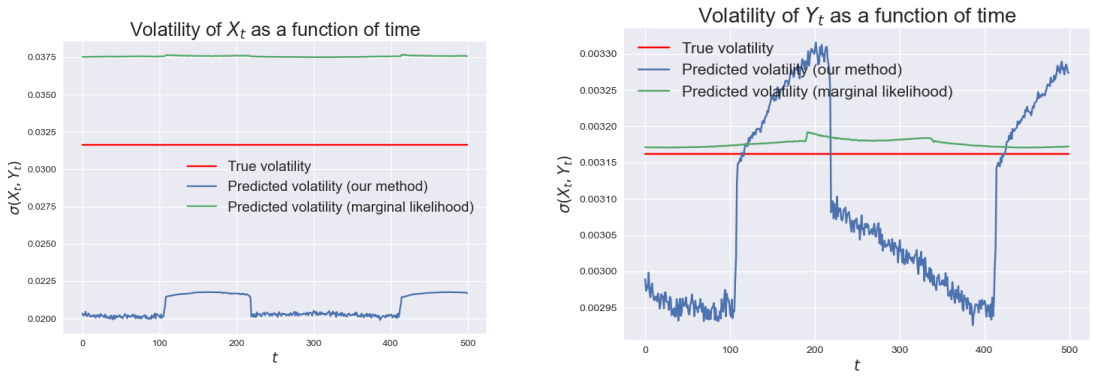



FIGURE 15. Stochastic Van der Pol 1: prediction of the volatility, $F(X_t, Y_t)$ (left) and $G(X_t, Y_t)$ (right).

This model has a more complex volatility source than the previous. The results are presented in Table 6. The trajectory is presented in Figure 27 and the drift and volatility predictions are presented in Figures 16 and 17.

| | $X_t$ | | | $Y_t$ | | |
|---|---|---|---|---|---|---|
| | $\mathcal{L}(\mathcal{M}\|Y)$ | $\delta_f$ | $\delta_\sigma$ | $\mathcal{L}(\mathcal{M}\|Y)$ | $\delta_f$ | $\delta_\sigma$ |
| Initial | -1741.38 | 0.558 | 3.204 | -3.339 | **0.204** | 905.007 |
| Log marginal | -1975.964 | 1.004 | 2.069 | -2843.491 | 0.322 | 2.069 |
| Our method | **-2090.291** | **0.357** | **0.479** | **-3138.301** | 0.209 | **0.4913** |

TABLE 6. Results on the stochastic Van der Pol 2.

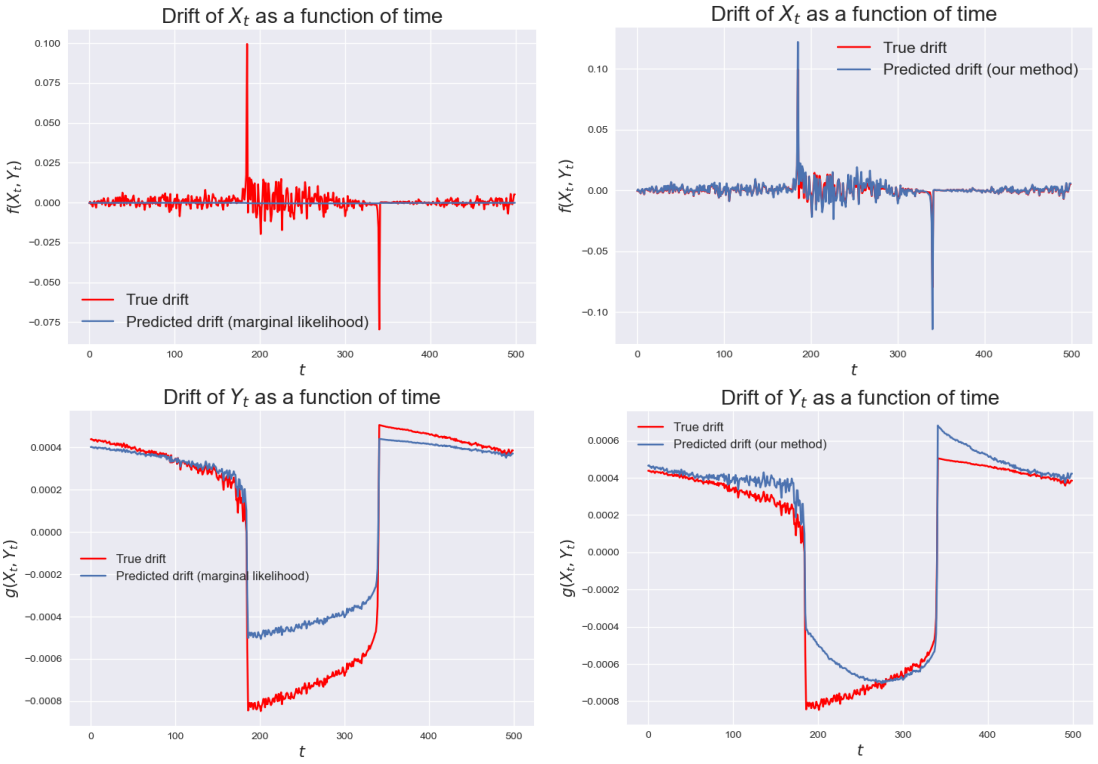



FIGURE 16. Stochastic Van der Pol 2: predictions on the drift (drift $X_t$ top, drift $Y_t$ bottom): comparison between maximizing the marginal likelihood (left) and our method (right).

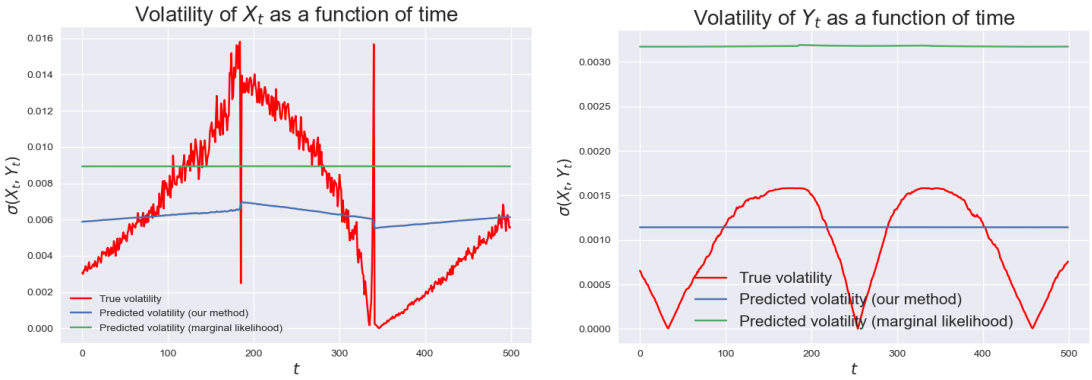



FIGURE 17. Stochastic Van der Pol 2: prediction of the volatility, $F(X_t, Y_t)$ (left) and $G(X_t, Y_t)$ (right).

6.9. **Comments on the results.** We make three observations regarding our results.

The first is that the randomized cross-validation algorithm for hyper-parameter optimization reliably improves the performance of our method. In all cases, the selected parameters have better performance compared to the initial guess, as measured by the likelihood of the model. Second, we observe that our method provides comparable or greater performance compared to simple kernel regression with hyper-parameters optimized through the minimization of the log marginal likelihood. Which method is preferable depends on the underlying SDE. We observe that for SDEs with constant volatility functions, such as the Ornstein–Uhlenbeck process or the first Van der Pol oscillator, the simple kernel regression generally outperforms our method as measured by the likelihood. This is likely due to the better recovery of the volatility as the modeling assumption of the kernel regression (i.i.d. noise) better captures the true model (constant volatility). Nonetheless, our method does occasionally outperform kernel regression in predicting the drift of the SDE. Our method, however, notably outperforms kernel regression for SDEs with non-constant volatility, such as the Cox–Ingersoll–Ross model, Geometric Brownian motion, and the second Van der Pol oscillator. In such cases, our model better captures the true volatility, including its trend (see for example figures 6 and 12). In the case of the sinusoidal process, kernel regression outperforms our model on trajectory 1, whereas our model outperforms kernel regression on trajectory 2. We hypothesize that this is due to the greater magnitude of the volatility for trajectory 2, which implies that accurately modeling the volatility is more important to capture the behavior of the SDE. Finally, we observe that a better likelihood generally implies a better capture of the drift and volatility. Hence, while these quantities are unobserved, better performance as measured by the likelihood generally implies that the model captures well both the drift and volatility. Exceptions generally occur when the volatility drives the evolution more than the drift (see, for example, the Geometric Brownian motion trajectory 1).

## 7. Additional Plots.

7.1. **OU process.** Additional plots for the Ornstein–Uhlenbeck process.

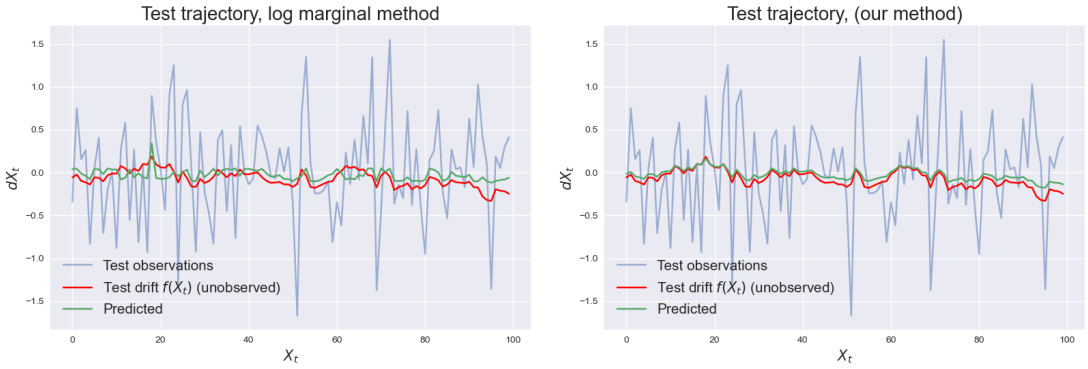



Figure 18. OU process, prediction of the drift on trajectory 2: marginal likelihood (left) and our method (right).

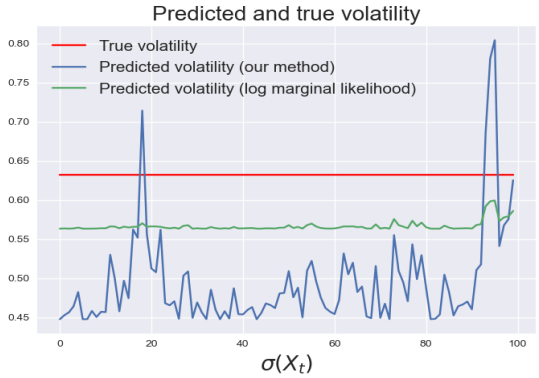



Figure 19. OU process, prediction of the volatility on trajectory 1 (left) and trajectory 2 (right).

7.2. **CIR model.** Additional plots for the Cox–Ingersoll–Ross process.

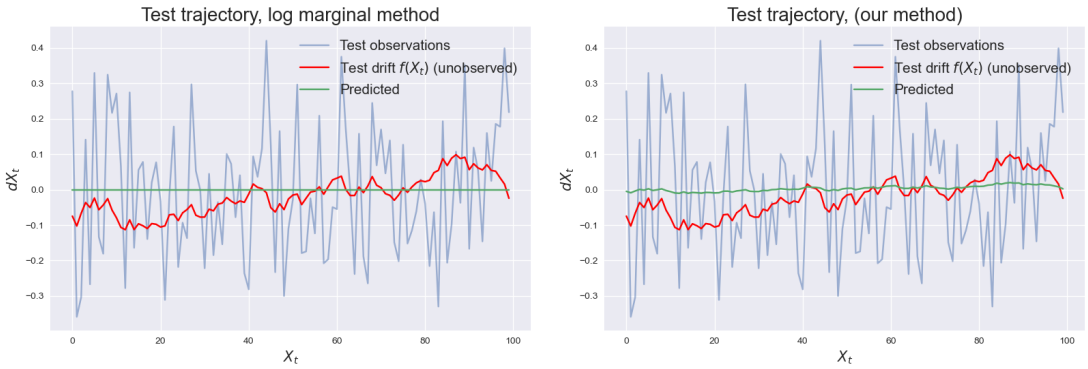



Figure 20. CIR process, prediction of the drift on trajectory 2: marginal likelihood (left) and our method (right)

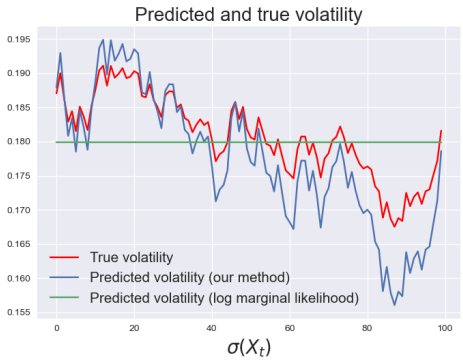



FIGURE 21. CIR process, prediction of the volatility on trajectory 2. The base Gaussian process method has a constant volatility whereas our method is able to model an evolving volatility function.

7.3. **Sinusoidal process.** Additional plots for the sinusoidal process.

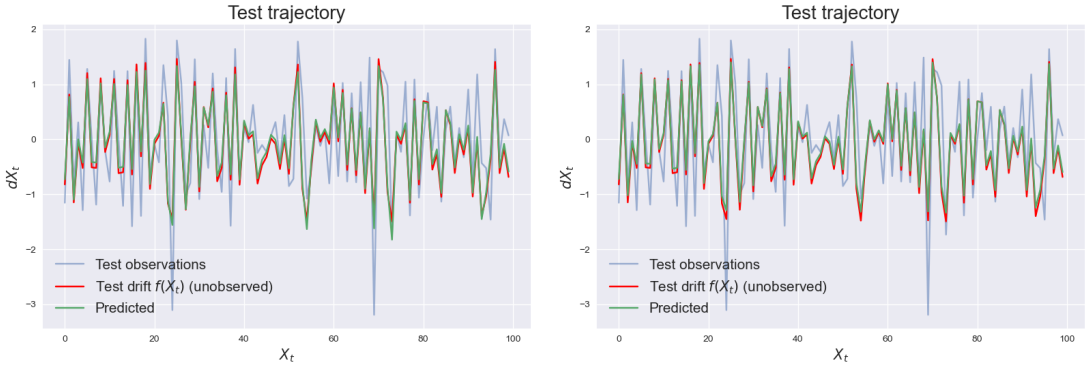



FIGURE 22. Sinusoidal process, drift prediction trajectory 2: marginal likelihood method (left) and our method (right)

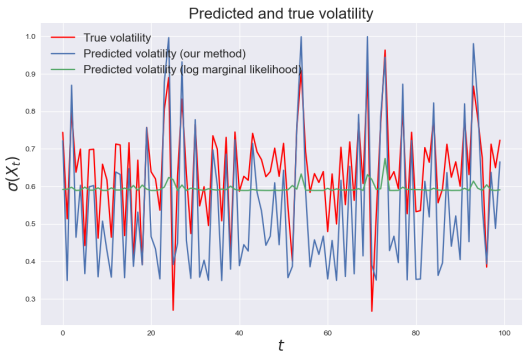



FIGURE 23. Sinusoidal process: volatility prediction, trajectory 1 (left), trajectory 2 (right).

7.4. **Geometric Brownian motion.** Additional plots for the Geometric Brownian motion process.


FIGURE 24. Trajectory of the first trajectory of GBM: trajectory (left) and increments (right)

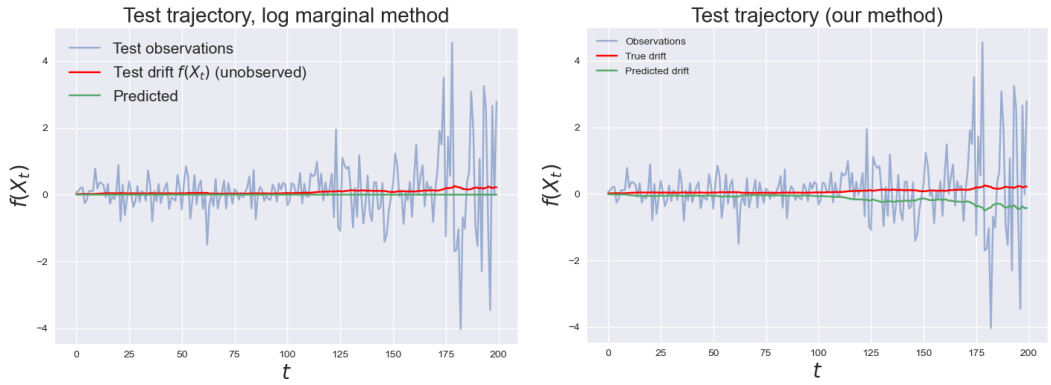



FIGURE 25. GBM process, drift prediction trajectory 1: marginal likelihood method (left) and our method (right)

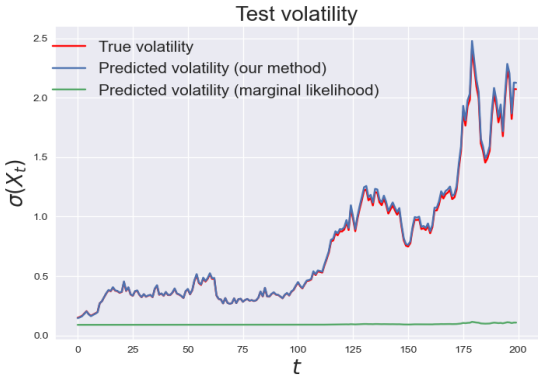



FIGURE 26. GBM process, volatility prediction on trajectory 2.

7.5. **Fast slow SDE.** Additional plots for the fast-slow SDE.

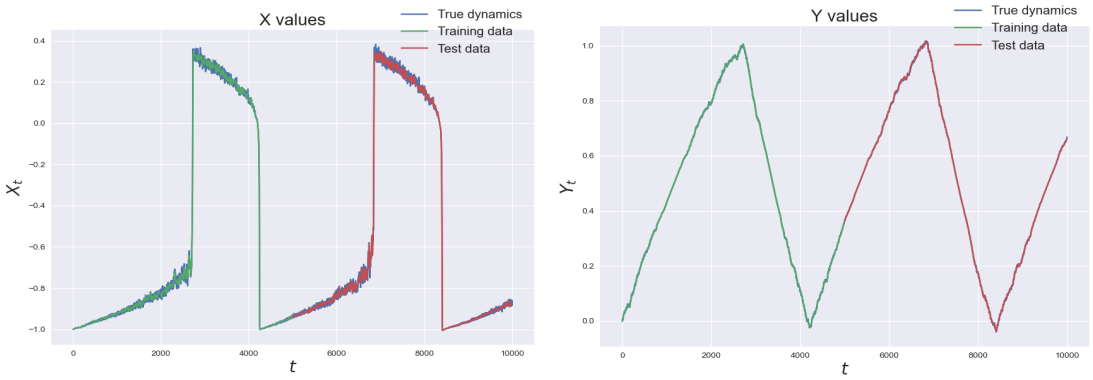



FIGURE 27. Stochastic Van der Pol 2.

## Appendix A. Gaussian Process Regression and Extension of [7, Page 306]

In this section, we give a very brief overview of Gaussian processes for regression. We suppose that the values of $Y(X)$ are distributed according to a Gaussian process, namely $Y(X) \overset{d}{\sim} \mathcal{GP}(\boldsymbol{O}, \boldsymbol{K})$. In particular, in the case of SDEs, given the data $(X, Y)$, the predicted drift and diffusion for a new point $x_*$ are given by

$$\begin{aligned}\bar{f}(x_*) &= K(x_*, X)K(X, X)^{-1}Y \\ \bar{\sigma}(x_*) &= K(x_*, x_*) - K(x_*, X)K(X, X)^{-1}K(X, x_*)\end{aligned}$$

The above expression are valid in the case of noisy observation with independent and identical Gaussian noise. Now, we derive these distribution in the case where the observations are noisy with independent, but not necessarily identical, Gaussian noise; the proof is generalized from the one presented in [7, 306].

Formally, suppose that we have at our disposal the noisy observations $(X_n, Y_n)_{1\leq n\leq N}$, where $Y_i = f(X_i)+W_i$ and the $W_i$ are independent but not necessarily identically distributed $\mathcal{N}(0, \sigma^2(X_i))$ random variables. The problem is the identification of the unknown function $f$ given these noisy observations. Set $f_i = f(X_i)$, so that $\boldsymbol{f} := (f_i)_{1\leq i\leq N}$. In addition, set $\boldsymbol{Y} := (Y_i)_{1\leq i\leq N}$. Observe that $\boldsymbol{Y}|\boldsymbol{f} \overset{d}{\sim} \mathcal{N}(\boldsymbol{f}, \boldsymbol{\Sigma})$ where $(\Sigma)_{ij} = \delta_{ij}\sigma^2(X_i)$, and assume that $\boldsymbol{f} \overset{d}{\sim} \mathcal{N}(\boldsymbol{0}, \boldsymbol{K})$. Then, [7, Page 93],

$$p(\boldsymbol{Y}) = \int p(\boldsymbol{Y}|\boldsymbol{f})p(\boldsymbol{f})d\boldsymbol{f} = \mathcal{N}(\boldsymbol{0}, \boldsymbol{K} + \boldsymbol{\Sigma}).$$

Now, let $\boldsymbol{C} = \boldsymbol{K} + \boldsymbol{\Sigma}$ so that $\boldsymbol{C}_{ij} = k(X_i, X_j) + \sigma_i^2\delta_{ij}$. Denote $\boldsymbol{Y}_{N+1} = (Y_1, ..., Y_{N+1})$, $\boldsymbol{Y}_N = (Y_1, ..., Y_N)$. We wish to derive the conditional distribution $p(Y_{N+1}|\boldsymbol{Y}_N)$. First observe that

$$p(\boldsymbol{Y}_{N+1}) = \mathcal{N}(\boldsymbol{0}, \boldsymbol{C}_{N+1}).$$

where $\boldsymbol{C}_{N+1}$ is the $(N + 1) \times (N + 1)$ with entries defined as previously for the vector $\boldsymbol{Y}_{N+1}$. We may partition the covariance matrix as

$$\boldsymbol{C}_{N+1} = \begin{pmatrix} \boldsymbol{C}_N & \boldsymbol{K} \\ \boldsymbol{K}^T & c \end{pmatrix}$$

where $c = K(X_{N+1}, X_{N+1}) + \sigma_{N+1}^2$ and $\boldsymbol{K}$ is the vector with entries $\overline{\boldsymbol{K}}_i = K(X_{N+1}, X_i)$. Then $p(Y_{N+1}|\boldsymbol{Y}_N) = \mathcal{N}(m(X_N), \sigma^2(X_N))$ and the conditional mean and covariance are given by

$$\begin{aligned}m(X_{N+1}) &= \overline{\boldsymbol{K}}^T\boldsymbol{C}^{-1}\boldsymbol{y}_N = \overline{\boldsymbol{K}}^T(\boldsymbol{K} + \boldsymbol{\Sigma})^{-1}\boldsymbol{y}_N \\ \sigma^2(X_N) &= c - \overline{\boldsymbol{K}}^T\boldsymbol{C}^{-1}\overline{\boldsymbol{K}} = c - \overline{\boldsymbol{K}}^T(\boldsymbol{K} + \boldsymbol{\Sigma})^{-1}\overline{\boldsymbol{K}}.\end{aligned}$$

## Appendix B. Log-marginal likelihood for hyper-parameter optimization

In this section, we briefly present the log-marginal likelihood method for hyper-parameter optimization. The marginal log-likelihood over the kernel parameters can be expressed as:

$$-\log(p(Y|\boldsymbol{\theta}, X)) \propto \frac{1}{2}Y^TK(X, X)^{-1}Y + \log(|K(X, X)|).$$

Therefore, the optimal parameters $\boldsymbol{\theta}$ are obtained via the minimization of the function in the previous equation (with respect to $\boldsymbol{\theta}$). In the case of noisy observations, the kernel $K$ can be defined as

$$K(X_1, X_2) = K'(X_1, X_2) + \delta(X_1, X_2)$$

where $K'$ is a standard kernel and $\delta$ is the white noise kernel defined as

$$\delta(X_1, X_2) := \begin{cases} c & \text{if } X_1 = X_2, \\ 0 & \text{otherwise,} \end{cases}$$

where $c$ is the noise level, a kernel parameter that must be optimized. It is important to note that this kernel can only account for a constant level of noise, which is not the case for many SDEs.

## Appendix C. Randomized cross-validation for hyper-parameter learning

In this section, we describe the general framework of *Randomized cross-validation* we use in our optimization.

The general problem is the following: we have a set of training data $(\boldsymbol{X}, Y) := \{(X_i, Y_i)\}_{1 \leq i \leq N}$ and we are trying to recover a function $f : \boldsymbol{X} \to Y$. In particular, we assume that we have a class of functions $\mathcal{S}$ indexed by a set of parameters $\boldsymbol{w}_p \in \mathcal{W}$, i.e. $\mathcal{S} := \{f(\,\cdot\,, \boldsymbol{w}_p) \,:\, \boldsymbol{w}_p \in \mathcal{W} \text{ and } f(\,\cdot\,, \boldsymbol{w}_p) : \boldsymbol{X} \to Y\}$. Notice that in practice this could be approximated via reproducing kernels as in this paper or via neural networks. We assume that we have a *method* to find the optimal parameter $\boldsymbol{w}_p^* \in \mathcal{W}$ for a given training set of data $(\boldsymbol{X}, Y)$. In our case, we assume that such a method involves the minimization of some loss function $\mathcal{L}(f(\boldsymbol{X}, \boldsymbol{w_p}), Y)$ with respect to $\boldsymbol{w}_p$, where $\mathcal{L} : Y \times Y \to \mathbb{R}$. Moreover, like in many Machine Learning (ML) algorithms, we assume that we have a set of hyper-parameters $\boldsymbol{\theta} \in \boldsymbol{\Theta}$ that affect the recovery of the optimal $\hat{f} \in \mathcal{S}$. Such hyper-parameters $\boldsymbol{\theta}$ can parametrize the function $\hat{f} := \hat{f}_{\boldsymbol{\theta}} := f(\,\cdot\,; \boldsymbol{w}_p, \boldsymbol{\theta})$, regularize the loss function $\mathcal{L}$ (often through some prior distribution on the parameters $\boldsymbol{w}_p$) or affect the minimization of the loss $\mathcal{L}$ (such as the learning rate of a gradient descent). We summarize this point by saying that $\hat{f}$ is recovered by minimizing $\mathcal{L}_{\boldsymbol{\theta}}$.

In the present work, in order to recover the function $f : \boldsymbol{X} \to Y$, we apply the Robust Learning Algorithm, whose rationale is explained in the following paragraph.

**Robust Learning Algorithm via Randomized cross-validation.** Many ML algorithms are built upon minimizing a loss function $\mathcal{L} : Y \times Y \to \mathbb{R}$ over the set of parameters $(\boldsymbol{w}_p, \boldsymbol{\theta})$ of a class of models $\hat{f}\,(\,\cdot\,, \boldsymbol{w}_p; \boldsymbol{\theta})$. In other words, *ideally*, we can define the risk function $R$ as the expected value of the loss function $\mathcal{L}$ with respect to the data probability density function $p\,(\boldsymbol{x}, y)$:

$$R\,(\boldsymbol{w}_p, \boldsymbol{\theta}) := \mathbb{E}_{(\boldsymbol{X},Y)\sim p(\boldsymbol{x},y)} \left\{ \mathcal{L}\left( \hat{f}(\boldsymbol{X}, \boldsymbol{w}_p; \boldsymbol{\theta}), Y \right) \right\} \tag{33}$$

and then find the set of optimal parameters according to

$$\left(\boldsymbol{w}_p^*, \boldsymbol{\theta}^*\right) := \underset{\boldsymbol{w}_p, \boldsymbol{\theta}}{\arg\min}\; R\,(\boldsymbol{w}_p, \boldsymbol{\theta})\,. \tag{34}$$

Notice that, in this setup, by the assumption that one has access to $p\,(\boldsymbol{x}, y)$, there is no theoretical distinction between parameters and hyper-parameters.

*However*, in practice, one only sees a realization of $(\boldsymbol{X}, Y)$, namely $\mathcal{D} = \{(\boldsymbol{X}_i, Y_i)\}_{i \in \mathcal{I}}$. Therefore, it is impossible to use Equations (33) and (34). Moreover, in order to achieve generalization, in practice $\boldsymbol{w}_p$ is optimized by minimizing a loss function dependent on $\boldsymbol{\theta}$:

$$\boldsymbol{w}_p^* = \underset{\boldsymbol{w}_p}{\arg\min}\, \mathcal{L}_{\boldsymbol{\theta}} \left( \hat{f}\,(\boldsymbol{X}; \boldsymbol{w}_p, \boldsymbol{\theta})\,, Y \right) \tag{35}$$

At this point, the best parameters $\boldsymbol{\theta}$ can be chosen via a cross validation approach where the function $\hat{f}_{\boldsymbol{\theta}}$ is evaluated on an unseen test set $(\boldsymbol{X_u}, Y_u)$:

$$R(\boldsymbol{\theta}) = \mathcal{L}\left( \hat{f}\left(\boldsymbol{X}_u; \boldsymbol{w}_p^*, \boldsymbol{\theta}\right), Y_u \right) \tag{36}$$

Generally, the optimization of hyper-parameters [5, 6, 28, 51, 38, 21] is usually done on a prefixed number of cross validation sets. In this work, we propose an approach that is not bound to a fixed number of cross validation sets. Our algorithm can be summarized as follows:

(1) Partition the available data $\mathcal{D} = \{(\boldsymbol{X}_i, Y_i)\}_{i \in \mathcal{I}}$ into a training subset $\mathcal{D}_{\mathcal{T}} = \{(\boldsymbol{X}_i, Y_i)\}_{i \in \mathcal{T}}$ and a test subsets $\mathcal{D}_{\mathcal{U}} = \{(\boldsymbol{X}_i, Y_i)\}_{i \in \mathcal{U}}$; the two subsets are mutually exclusive.
(2) Randomly partition the training set $\mathcal{D}_{\mathcal{T}}$ into two mutually exclusive subsets of almost equal size: $\mathcal{D}_{\prod} = \{(\boldsymbol{X}_i, Y_i)\}_{i \in \prod(\mathcal{T})}$ and $\mathcal{D}_{\prod^c} = \{(\boldsymbol{X}_i, Y_i)\}_{i \in \prod^c(\mathcal{T})}$. Here, $\prod(\mathcal{T})$ returns the first half of the random permutation of indices in $\mathcal{T}$ and $\prod^c(\mathcal{T})$ returns the second half of the same permutation.

(3) Train a ML model on $\mathcal{D}_{\prod}$ and evaluate the random loss on $\mathcal{D}_{\prod^c}$ representing a realization of the generalization error.
(4) Repeat steps (2) and (3) to optimize the expected loss over the random sets $\mathcal{D}_{\prod^c}$ with respect to $\boldsymbol{w}_p$.
(5) Check the goodness of fit by evaluating the loss over $\mathcal{D}_{\mathcal{U}}$.

More precisely, the optimization problem is the following one:

$$\begin{aligned} \boldsymbol{\theta}^* &= \arg\min_{\boldsymbol{\theta}} \mathbb{E}_{\Pi}\left\{R_{\mathbb{I}-\prod}\left(\bar{\boldsymbol{w}}_p, \boldsymbol{\theta}\right)\right\} \\ \text{s.t.} \quad \bar{\boldsymbol{w}}_p &= \arg\min_{\boldsymbol{w}_p} R_{\prod}\left(\boldsymbol{w}_p, \boldsymbol{\theta}\right). \end{aligned} \tag{37}$$

Where,

$$R_{\prod}\left(\boldsymbol{w}_p, \boldsymbol{\theta}\right) = \mathbb{E}_{\boldsymbol{X}, Y \sim \mathcal{D}_{\prod}}\left\{\mathcal{L}_{\boldsymbol{\theta}}\left(\hat{f}\left(\boldsymbol{X}; \boldsymbol{w}_p, \boldsymbol{\theta}\right), Y\right)\right\}, \tag{38}$$

and

$$R_{\mathbb{I}-\prod}\left(\boldsymbol{w}_p, \boldsymbol{\theta}\right) = \mathbb{E}_{\boldsymbol{X}, Y \sim \mathcal{D}_{\prod^c}}\left\{\mathcal{L}\left(\hat{f}\left(\boldsymbol{X}; \boldsymbol{w}_p, \boldsymbol{\theta}\right), Y\right)\right\}. \tag{39}$$

In the previous Equations, $\mathbb{E}_{\boldsymbol{X}, Y \sim \mathcal{D}_{\prod}}\{.\}$ means that the expected value is taken over the empirical distribution defined by $\mathcal{D}_{\prod}$. A similar notion applies to $\mathbb{E}_{\boldsymbol{X}, Y \sim \mathcal{D}_{\prod^c}}\{.\}$. Furthermore, $\mathbb{E}_{\Pi}\{.\}$ means that this expected value is taken over all permutation of the train set. Note that the recovery of $\bar{\boldsymbol{w}}_p$ is done through the minimization of $\mathcal{L}_{\boldsymbol{\theta}}$ which we refer to as the train loss function, whereas the evaluation of $\hat{f}\left(\boldsymbol{X}; \boldsymbol{w}_p, \boldsymbol{\theta}\right)$ is done through $\mathcal{L}$ which we refer to as the test loss function. The Robust Learning Algorithm is presented here below in (1):

**Algorithm 1 Robust Learning Algorithm**

**Require:**
(1) Pick a model class $\hat{f}_{\boldsymbol{\theta}}$
(2) Pick an initial guess $\boldsymbol{\theta}^{(0)}$.
(3) Pick an active learning algorithm $Al$.
(4) Set $\mathcal{S} = \phi$, $n = 0$ and $C = 0$.

**Ensure:** Partition the data $\mathcal{D} = \{(\boldsymbol{x}_i, y_i)\}_{i \in \mathcal{I}}$ into two mutually exclusive training $\mathcal{D}_{\mathcal{T}} = \{(\boldsymbol{x}_i, y_i)\}_{i \in \mathcal{T}}$ and test subsets $\mathcal{D}_{\mathcal{U}} = \{(\boldsymbol{x}_i, y_i)\}_{i \in \mathcal{U}}$.

**while** $C = 0$ **do**
  $n \leftarrow n + 1$
  **while** $1 \leq j \leq K$ **do**
    $j \leftarrow j + 1$
    Pick a random index permutation $\prod_n(\mathcal{T})$ of $\mathcal{T}$.
    Divide the train set $\mathcal{D}_{\mathcal{T}}$ into $\mathcal{D}_{\prod_n}$ and $\mathcal{D}_{\prod_n^c}$.
    Train $\hat{f}\left(\boldsymbol{X}; \boldsymbol{w}_p, \boldsymbol{\theta}^{(n-1)}\right)$ on $\mathcal{D}_{\prod_n}$ with respect to $R_{\prod_n}\left(\boldsymbol{w}_p, \boldsymbol{\theta}^{(n-1)}\right)$ to obtain $\bar{\boldsymbol{w}}_p$.
    Evaluate $e_j = R_{\mathbb{I}-\prod_n}\left(\bar{\boldsymbol{w}}_p, , \boldsymbol{\theta}^{(n-1)}\right)$ on $\mathcal{D}_{\prod_n^c}$.
  **end while**
  Set $R_{\mathbb{I}-\prod_n} = \frac{1}{K}\sum_{j=1}^{K} e_j$.
  Set $\mathcal{S} \leftarrow \mathcal{S} \cup \left\{\left(\boldsymbol{\theta}^{(n-1)}, R_{\mathbb{I}-\prod_n}\left(\bar{\boldsymbol{w}}_p, , \boldsymbol{\theta}^{(n-1)}\right)\right)\right\}$.
  **if** $Al\left(\mathcal{S}\right)$ has converged **then**
    $C = 1$
    $\left(\boldsymbol{w}_p^*, \boldsymbol{\theta}^*\right) = \left(\bar{\boldsymbol{w}}_p, , \boldsymbol{\theta}^{(n-1)}\right)$.
  **else if** $Al\left(\mathcal{S}\right)$ has not converged **then**
    $\boldsymbol{\theta}^{(n)} = Al\left(\mathcal{S}\right)$
  **end if**
**end while**
Check the the loss over $\mathcal{D}_{\mathcal{U}}$.

In particular, notice that $R_{\mathbb{I}-\prod_n} = \frac{1}{K}\sum_{j=1}^{K} e_j$ is a (noisy) approximation of

$$\mathbb{E}_{\boldsymbol{X},Y\sim\mathcal{D}_{\prod^c}}\left\{\mathcal{L}\left(\hat{f}\left(\boldsymbol{X};\boldsymbol{w}_p,\boldsymbol{\theta}\right),Y\right)\right\}.$$

**Acknowledgments.** MD, BH, HO acknowledge partial support by the Air Force Office of Scientific Research under MURI award number FA9550-20-1-0358 (Machine Learning and Physics-Based Modeling and Simulation). MD, PT and HO acknowledge support from Beyond Limits (Learning Optimal Models) through CAST (The Caltech Center for Autonomous Systems and Technologies).

## References

[1] Bayesian optimization with skopt. `https://scikit-optimize.github.io/stable/auto_examples/bayesian-optimization.html`. Accessed: 2021-09-07.

[2] H. Abarbanel. *Analysis of Observed Chaotic Data*. Institute for Nonlinear Science. Springer New York, 2012.

[3] Romeo Alexander and Dimitrios Giannakis. Operator-theoretic framework for forecasting nonlinear time series with kernel analog techniques. *Physica D: Nonlinear Phenomena*, 409:132520, 2020.

[4] Cedric Archambeau, Dan Cornford, Manfred Opper, and John Shawe-Taylor. Gaussian process approximations of stochastic differential equations. In Neil D. Lawrence, Anton Schwaighofer, and Joaquin Quiñonero Candela, editors, *Gaussian Processes in Practice*, volume 1 of *Proceedings of Machine Learning Research*, pages 1–16, Bletchley Park, UK, 12–13 Jun 2007. PMLR.

[5] James Bergstra, Rémi Bardenet, Yoshua Bengio, and Balázs Kégl. Algorithms for hyper-parameter optimization. In J. Shawe-Taylor, R. Zemel, P. Bartlett, F. Pereira, and K. Q. Weinberger, editors, *Advances in Neural Information Processing Systems*, volume 24. Curran Associates, Inc., 2011.

[6] James Bergstra and Yoshua Bengio. Random search for hyper-parameter optimization. *J. Mach. Learn. Res.*, 13(null):281–305, feb 2012.

[7] C. M. Bishop. *Pattern Recognition and Machine Learning*. Springer, 2006.

[8] Andreas Bittracher, Stefan Klus, Boumediene Hamzi, Péter Koltai, and Christof Schütte. Dimensionality reduction of complex metastable systems via kernel embeddings of transition manifolds. 2019. `https://arxiv.org/abs/1904.08622`.

[9] J. Bouvrie and B. Hamzi. Balanced reduction of nonlinear control systems in reproducing kernel hilbert space. *Proc. 48th Annual Allerton Conference on Communication, Control, and Computing*, pages 294–301, 2010. `https://arxiv.org/abs/1011.2952`.

[10] Jake Bouvrie and Boumediene Hamzi. Empirical estimators for stochastically forced nonlinear systems: Observability, controllability and the invariant measure. *Proc. of the 2012 American Control Conference*, pages 294–301, 2012. `https://arxiv.org/abs/1204.0563v1`.

[11] Jake Bouvrie and Boumediene Hamzi. Kernel methods for the approximation of nonlinear systems. *SIAM J. Control and Optimization*, 2017. `https://arxiv.org/abs/1108.2903`.

[12] Jake Bouvrie and Boumediene Hamzi. Kernel methods for the approximation of some key quantities of nonlinear systems. *Journal of Computational Dynamics*, 1, 2017. `http://arxiv.org/abs/1204.0563`.

[13] Steven L. Brunton, Joshua L. Proctor, and J. Nathan Kutz. Discovering governing equations from data by sparse identification of nonlinear dynamical systems. *Proceedings of the National Academy of Sciences*, 113(15):3932–3937, 2016.

[14] Martin Casdagli. Nonlinear prediction of chaotic time series. *Physica D: Nonlinear Phenomena*, 35(3):335 – 356, 1989.

[15] Ashesh Chattopadhyay, Pedram Hassanzadeh, Krishna V. Palem, and Devika Subramanian. Data-driven prediction of a multi-scale lorenz 96 chaotic system using a hierarchy of deep learning methods: Reservoir computing, ann, and RNN-LSTM. *CoRR*, abs/1906.08829, 2019.

[16] Yifan Chen, Bamdad Hosseini, Houman Owhadi, and Andrew M Stuart. Solving and learning nonlinear pdes with gaussian processes. *arXiv preprint arXiv:2103.12959*, 2021.

[17] Felipe Cucker and Steve Smale. On the mathematical foundations of learning. *Bulletin of the American Mathematical Society*, 39:1–49, 2002.

[18] Felix Dietrich, Alexei Makeev, George Kevrekidis, Nikolaos Evangelou, Tom Bertalan, Sebastian Reich, and Ioannis G. Kevrekidis. Learning effective stochastic differential equations from microscopic simulations: combining stochastic numerics and deep learning, 2021.

[19] Noura Dridi, Lucas Drumetz, and Ronan Fablet. Learning stochastic dynamical systems with neural networks mimicking the euler-maruyama scheme. *CoRR*, abs/2105.08449, 2021.

[20] B.Haasdonk ,B.Hamzi , G.Santin , D.Wittwar. Kernel methods for center manifold approximation and a weak data-based version of the center manifold theorems. *Physica D*, 2021.

[21] Luca Franceschi, Michele Donini, Paolo Frasconi, and Massimiliano Pontil. Forward and reverse gradient-based hyperparameter optimization, 2017.

[22] Rudolf Friedrich, Joachim Peinke, Muhammad Sahimi, and M. Reza Rahimi Tabar. Approaching complexity by stochastic methods: From biological systems to turbulence. *Physics Reports*, 506(5):87–162, 2011.

[23] P. Giesl, B. Hamzi, M. Rasmussen, and K. Webster. Approximation of Lyapunov functions from noisy data. *Journal of Computational Dynamics*, 2019. https://arxiv.org/abs/1601.01568.

[24] B. Haasdonk, B. Hamzi, G. Santin, and D. Wittwar. Greedy kernel methods for center manifold approximation. *Proc. of ICOSAHOM 2018, International Conference on Spectral and High Order Methods*, (1), 2018. https://arxiv.org/abs/1810.11329.

[25] Boumediene Hamzi and Fritz Colonius. Kernel methods for the approximation of discrete-time linear autonomous and control systems. *SN Applied Sciences*, 1(7):1–12, 2019.

[26] Boumediene Hamzi, Christian Kuehn, and Sameh Mohamed. A note on kernel methods for multiscale systems with critical transitions. *Mathematical Methods in the Applied Sciences*, 42(3):907–917, 2019.

[27] Boumediene Hamzi and Houman Owhadi. Learning dynamical systems from data: A simple cross-validation perspective, part i: Parametric kernel flows. *Physica D: Nonlinear Phenomena*, 421:132817, 2021.

[28] Frank Hutter, Holger H. Hoos, and Kevin Leyton-Brown. Sequential model-based optimization for general algorithm configuration. In Carlos A. Coello Coello, editor, *Learning and Intelligent Optimization*, pages 507–523, Berlin, Heidelberg, 2011. Springer Berlin Heidelberg.

[29] Rob Hyndman and G. Athanasopoulos. *Forecasting: Principles and Practice*. OTexts, Australia, 3rd edition, 2021.

[30] Saba Infante, César Luna, Luis Sánchez, and Aracelis Hernández. Approximations of the solutions of a stochastic differential equation using dirichlet process mixtures and gaussian mixtures. *Statistics, Optimization amp; Information Computing*, 4(4):289–307, Dec. 2016.

[31] Holger Kantz and Thomas Schreiber. *Nonlinear Time Series Analysis*. Cambridge University Press, USA, 1997.

[32] David Kleinhans and Rudolf Friedrich. Quantitative estimation of drift and diffusion functions from time series data. In Joachim Peinke, Peter Schaumann, and Stephan Barth, editors, *Wind Energy*, pages 129–133, Berlin, Heidelberg, 2007. Springer Berlin Heidelberg.

[33] Yu Klimontovich. The reconstruction of the fokker-planck and master equations on the basis of experimental data: H-theorem and s-theorem. *International Journal of Bifurcation and Chaos*, 3:113–, 02 1993.

[34] Stefan Klus, Feliks Nuske, and Boumediene Hamzi. Kernel-based approximation of the koopman generator and schrödinger operator. *Entropy*, 22, 2020. https://www.mdpi.com/1099-4300/22/7/722.

[35] Stefan Klus, Feliks Nüske, Sebastian Peitz, Jan-Hendrik Niemann, Cecilia Clementi, and Christof Schütte. Data-driven approximation of the koopman generator: Model reduction, system identification, and control. *Physica D: Nonlinear Phenomena*, 406:132416, 2020.

[36] Andreas Bittracher, Stefan Klus, Boumediene Hamzi, Peter Koltai, and Christof Schutte. Dimensionality reduction of complex metastable systems via kernel embeddings of transition manifold, 2019. https://arxiv.org/abs/1904.08622.

[37] Jonghyeon Lee, Edward De Brouwer, Boumediene Hamzi, and Houman Owhadi. Learning dynamical systems from data: A simple cross-validation perspective, part iii: Irregularly-sampled time series, 2021.

[38] Dougal Maclaurin, David Duvenaud, and Ryan P. Adams. Gradient-based hyperparameter optimization through reversible learning, 2015.

[39] A. Nielsen. *Practical Time Series Analysis: Prediction with Statistics and Machine Learning*. O'Reilly Media, 2019.

[40] Manfred Opper. Variational inference for stochastic differential equations. *Annalen der Physik*, 531(3):1800233, 2019.

[41] Boumediene Hamzi , Romit Maulik, Houman Owhadi. Simple, low-cost and accurate data-driven geophysical forecasting with learned kernels. *Proceedings of the Royal Society A: Mathematical, Physical and Engineering Sciences*, 477(2252), 2021.

[42] H. Owhadi and G. R. Yoo. Kernel flows: From learning kernels from data into the abyss. *Journal of Computational Physics*, 389:22–47, 2019.

[43] Houman Owhadi. Computational graph completion. *arXiv preprint arXiv:2110.10323*, 2021.

[44] M. Darcy , B. Hamzi , J. Susiluo , A. Braverman , H. Owhadi. Learning dynamical systems from data: a simple cross-validation perspective, part ii: nonparametric kernel flows. *preprint*, 2021.

[45] Jaideep Pathak, Zhixin Lu, Brian R. Hunt, Michelle Girvan, and Edward Ott. Using machine learning to replicate chaotic attractors and calculate lyapunov exponents from data. *Chaos: An Interdisciplinary Journal of Nonlinear Science*, 27(12):121102, 2017.

[46] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay. Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, 12:2825–2830, 2011.

[47] Carl Edward Rasmussen and Christopher K. I. Williams. *Gaussian Processes for Machine Learning (Adaptive Computation and Machine Learning)*. The MIT Press, 2005.

[48] H. Risken and H. Haken. *The Fokker-Planck Equation: Methods of Solution and Applications Second Edition*. Springer, 1989.

[49] Gabriele Santin and Bernard Haasdonk. Kernel methods for surrogate modeling. 2019. https://arxiv.org/abs/1907.105566.

[50] S Siegert, R Friedrich, and J Peinke. Analysis of data sets of stochastic systems. *Physics Letters A*, 243(5-6):275–280, Jul 1998.

[51] Jasper Snoek, Hugo Larochelle, and Ryan P. Adams. Practical bayesian optimization of machine learning algorithms, 2012.

[52] Sai Prasanth , Ziad S Haddad , Jouni Susiluoto , Amy J Braverman , Houman Owhadi, Boumediene Hamzi , Svetla M Hristova-Veleva , Joseph Turk. Kernel flows to infer the structure of convective storms from satellite passive microwave observations. *preprint*, 2021.

[53] Jouni Susiluoto , Amy Braverman , Philip G. Brodrick , Boumediene Hamzi , Maggie Johnson , Otto Lamminpaa , Houman Owhadi , Clint Scovel , Joaquim Texeira , Michael Turmon. Radiative transfer emulation for hyperspectral imaging retrievals with advanced kernel flows-based gaussian process emulation. *preprint*, 2021.

[54] Cagatay Yildiz, Markus Heinonen, Jukka Intosalmi, Henrik Mannerstrom, and Harri Lahdesmaki. Learning stochastic differential equations with gaussian processes without gradient matching. In *2018 IEEE 28th International Workshop on Machine Learning for Signal Processing (MLSP)*, pages 1–6, 2018.

[1] Department of Computing and Mathematical Sciences, Caltech, CA, USA.
*Email address*: mdarcy@caltech.edu

[4] Department of Computing and Mathematical Sciences, Caltech, CA, USA.
*Email address*: boumediene.hamzi@gmail.com

[3] Scuola Normale Superiore, Pisa, Italy
*Email address*: giulia.livieri@sns.it

[5]Department of Computing and Mathematical Sciences, Caltech, CA, USA.
*Email address*: owhadi@caltech.edu

[2] JPL, Caltech, CA, USA.
*Email address*: peyman.tavallali@jpl.nasa.gov