Approximate Uncertainty Propagation for Continuous Gaussian Process Dynamical Systems

Steffen Ridderbusch[*]
Department of Engineering
University of Oxford
`steffen@robots.ox.ac.uk`

Sina Ober-Blöbaum
Numerical Mathematics and Control
University of Paderborn

Paul Goulart
Department of Engineering
University of Oxford

Abstract

When learning continuous dynamical systems with Gaussian Processes, computing trajectories requires repeatedly mapping the distributions of uncertain states through the distribution of learned nonlinear functions, which is generally intractable. Since sampling-based approaches are computationally expensive, we consider approximations of the output and trajectory distributions. We show that existing methods make an incorrect implicit independence assumption and underestimate the model-induced uncertainty. We propose a piecewise linear approximation of the GP model yielding a class of numerical solvers for efficient uncertainty estimates matching sampling-based methods.

1 Introduction

Gaussian Processes (GPs) represent a distribution over a function space, assumed to contain the true underlying function, conditioned on available pairs of input-output data. This means, that GP models capture *epistemic uncertainty* [1], as we explicitly assume that with additional data the uncertainty about the true model decreases.

The context for this work is learning the vector field of a dynamical system using GPs [2]. Specifically, we consider an ordinary differential equation (ODE)

$$\dot{x} = f(x), \tag{1}$$

where f is a GP. This is conceptually similar to using NeuralODEs, which use a Neural Network in place of the GP [3].

We note explicitly that the resulting model is not a stochastic differential equation or random dynamical system with uncertainty arising from inherent stochasticity [1]. While we assume the presence of measurement noise in the data, the GP model aims to identify the true underlying function. The context of this work is also different from the contributions of Schober et al. [4], who assume that the solution of the ODE is described by a GP, and the discrete steps of the solver represent observations affected by noise through computational error.

A more popular alternative to learning the vector-field representing the continuous dynamics is directly learning the flow map via a *state-space GP* [5, 6, 7, 8].

[*]Corresponding author

These models generally have the form

$$x_{k+1} = f(x_k), \tag{2}$$

directly predicting the next state of the trajectory after some fixed time step. Some of these models use Gaussian auto-regression, taking into account l past states [8, 9, 10]. While these approaches make it easy to use collected data, which is often available as a time series, they are implicitly learning a flow map with a fixed step size and are therefore unable to adapt to speed of the dynamics in a particular location. In addition, for many dynamical systems the flow map is a highly complex function that cannot be expressed analytically, even for simple vector fields.

Learning the continuous dynamics from time series data is more difficult, but also allows the inclusion of additional data, such as known terms from first principle models [3], symmetries [2], and symplectic structure [11].

In most cases, we obtain a trajectory from the continuous vector field via numerical integration methods. The simplest such method is the explicit Euler

$$x_{n+1} = x_n + hf(x_n), \tag{3}$$

where h is the step size, which can be varied over subsequent steps. There exists a wide range of more complex numerical solvers, differing in their order of convergence, stability, or computational cost. One of the key motivations for this work is applying those methods to GP-based ODEs and including uncertainty propagation.

The challenge with uncertainty propagation is that mapping a random variable with a tractable, parametrizable distribution, like a normal distribution, through a nonlinear function is generally intractable.

In the context of dynamical system models, this problem is compounded. For state-space GPs, one repeatedly maps a random variable X_n through the distribution of nonlinear functions represented by the GP to obtain the distribution of X_{n+1}, X_{n+2} and so forth. In the context of this work, the vector field is non-linear, which means that we obtain a distribution of gradients $f(X)$ for state X, which for numerical solvers needs to be combined with the distribution of current state X_n and, for higher order methods, the gradients at other states.

To our knowledge, this work is the first to consider approximate uncertainty propagation through numerical integrators for ODEs. However, there has been some work on uncertainty propagation in state-space GPs.

Girard et al. published a series of results on approximating the output distribution when mapping a normal distribution through a GP. In [12] the authors show an approximation for the mean and variance of the output based on Taylor series approximations of the predicted mean and variance, using derivatives for the GP kernel function. A subsequent result [7] showed that when using the squared exponential kernel specifically, the mean and the variance of the output distribution can be obtained analytically. These results are called *moment matching* and have been used for state-space GPs [5]. Other work has applied sampling-based approaches to estimate uncertainty in state-space GPs [13] and for continuous GP dynamics [14].

For comparison, we use a similar option in this work: We sample a representative number of vector field functions from the GP distribution and integrate each function over the desired time frame to get an estimate of the state distributions.

We briefly discuss the expected outcome. We assume that the data-generating system has an unknown, but fixed model. While the observations are noisy, the implicit parameters of the system are constant, and the finite GP conditioned on the collected data represents a distribution over different possible models and their likelihood. An ideal experimental setup would allow us to add additional data and narrow down to the true underlying model for the system in the Reproducing Kernel Hilbert Space. When considering the resulting trajectories, a fixed point of the system should become a fixed point distribution which a numerical integrator converges to through repeated steps, assuming that the step size is chosen from within the stability region.

Our main contributions are the following: We demonstrate how existing approaches to uncertainty propagation based on approximating the output distribution of the exact model make an incorrect implicit independence assumption. We further propose an alternative approach based on a piecewise linear model approximation that can be solved exactly, resulting in what we call the PULL (Propagating Uncertainty through Local Linearization) class of solvers.

2 Review

2.1 Gaussian Processes

We first review briefly the basics of Gaussian Processes [15]. We assume we have N output observations $y^i = f(x^i) + \epsilon^i$ with Gaussian noise $\epsilon^i \sim \mathcal{N}(0, \sigma_n^2)$ for known inputs x^i to an unknown function $f : \mathbb{R}^m \to \mathbb{R}^d$. We now express f as a GP, that is as a distribution over functions.We specify a prior $\mathbb{P}(f)$ in the form of a mean function $m(x)$, generally assumed to be zero, and a kernel function $k(x, x')$, which uniquely determines a Reproducing Kernel Hilbert Space containing all possible realizations of f.

For any finite subset of random variables f_i corresponding to known inputs x_i the GP determines a joint Gaussian distribution, and by conditioning on the initial observations $\mathcal{D} = (x^i, y^i)$ we obtain the posterior distribution $\mathbb{P}(f|\mathcal{D})$, which allows us to predict the output $f(\hat{x})$ at a new input $\hat{x}$. As we are considering a distribution of functions, we will obtain a distribution of outputs with the mean $\mu_f(\hat{x})$ and variance $\sigma_f^2(\hat{x})$ determined by

$$\mu_f(\hat{x}) = K_{\hat{x},\mathbf{x}}(K_{\mathbf{x},\mathbf{x}} + \sigma_n^2 I)^{-1}\mathbf{y} \tag{4a}$$

$$\sigma_f^2(\hat{x}) = K_{\hat{x},\hat{x}} - K_{\hat{x},\mathbf{x}}(K_{\mathbf{x},\mathbf{x}} + \sigma_n^2 I)^{-1}K_{\mathbf{x},\hat{x}} \tag{4b}$$

where $\mathbf{x} = [x^1, \ldots, x^N]$, $\mathbf{y} = [y^1, \ldots, y^N]$, and $K_{x,x'} = [k(x^i, x'^j)]_{i,j}$ is the kernel or covariance matrix. The most commonly used Kernel is the squared exponential kernel, which we also use in the examples shown in this work. It is defined as

$$k(x, x') = \exp\left(-\frac{1}{2}(x - x')^T W^{-1}(x - x')\right), \tag{5}$$

where $W = \text{diag}(w_1, \ldots, w_M)$ is the diagonal matrix of length scales.

In the context of dynamical systems the dimension of the input and the output are the same, hence $m = d$ While a number of vector-valued kernels exist [16], it is often assumed that each output can be treated independently with d GPs $f : \mathbb{R}^m \to \mathbb{R}$. In this work, we further restrict to one-dimensional dynamical systems, allowing us to consider only one-dimensional GPs.

2.2 Sampling GPs

GPs are distributions over, generally infinite dimensional, function spaces and therefore inherently difficult to sample numerically. To obtain some approximate function realizations f^* from the posterior distribution at input locations x^*, the standard option [17] is to generate normally distributed random variables $\zeta \sim \mathcal{N}(0, I)$, and transform them according to the GP posterior

$$f^*|\mathbf{y}, \mathbf{x} = m^* + K_{*,*}^{1/2}\zeta. \tag{6}$$

Here, $(\cdot)^{1/2}$ indicates a matrix square root like the Cholesky factor, m^* is the GP mean $\mu_f(x^*)$ from (4a) and $K_{*,*}$ is the covariance matrix for the sample input x^* via (4b). In essence, this is a *grid-based* approach, and while it is numerically exact, it is computationally costly due to cubic scaling with the number of function values sampled. To evaluate the function at arbitrary locations between grid points, one can use standard interpolation methods.

To mitigate the cubic scaling, a number of alternate algorithms have been developed. Wilson et al. [17] propose a combination of decoupled bases, specifically *Fourier basis functions* [18], and kernel bases via a sparse GP approach. This has also been used in other recent work [11, 14] in the context of vector field models.

An approach of this kind was also mentioned in [13], along with a *memory based* approach, which generates samples subsequently while conditioning each sample on previous ones. It has also been used in a similar context as in this work [2].

For this work, we will use samples via (6) as a source of ground truth for comparison, as it is the most accurate option to compare against.

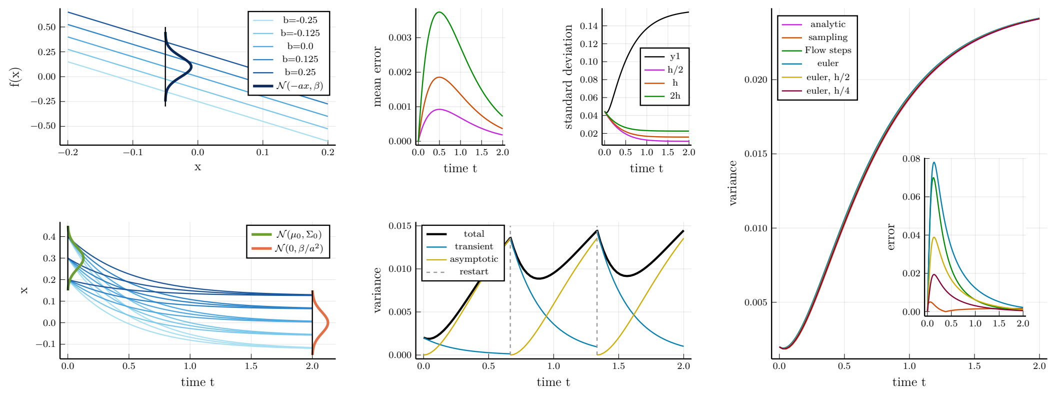

Figure 1: Linear Model: *Left:* The distribution is represented by a family of parallel lines. Solving linear ODEs for samples b of B for a distribution of initial values shows trajectories converge to the model parameter-dependent fixed point. *Middle:* Decreasing the Euler step size decreases the mean error, but also incorrectly decreases the variance as a result of (17b). The reason becomes apparent in the bottom plot, where the black line represents sampling, while regularly restarting and independently sampling from the final distribution of each segment. It matches the sum of the analytic transient and asymptotic terms. *Right:* Iterating the flow and taking Euler steps with the correct covariance, we obtain results that match the analytic solution and sampling. Decreasing the step size reduces the error compared to the analytic solution.

3 Linear Approximations

Using a linear approximation is attractive in many applications, as it avoids many of the complex behaviours that appear in nonlinear functions. However, the question is what part of the problem requires, or is best suited for, linearization.

3.1 Approximating the Output Distribution

Even though the distribution of each subsequent state random variable X_n is generally not normally distributed, we can compute its mean and variance and instead of the unknown general distribution use a normal distribution with the same moments.

Applying the approach of Girard et al. [7] in the context of the explicit Euler method (3), an initial idea might use to apply it to the GP f and then treat the sum of X_n and $f(X_n)$ as independent. However, this leads to an inaccurate growth in the uncertainty, as they are clearly not independent.

Instead, we can define a new function $g(x) = x + hf(x)$ and derive the equivalent expressions. Let the input X_n be distributed as

$$X_n \sim \mathcal{N}(\nu_n, \Sigma_n), \tag{7}$$

and the output of the GP for a particular sample x^* of X_n is distributed as

$$f(x^*) \sim \mathcal{N}(\mu(x^*), \sigma^2(x^*)). \tag{8}$$

The distribution of $X_{n+1} = g(X_n)$ will generally not be normal, but we can compute its mean $\nu_{n+1} = \mathbb{E}[x_{n+1}]$ and variance $\Sigma_{n+1} = \text{var}(x_{n+1})$.

From the law of iterated expectations follows that the mean of the output distribution of one Euler step is given by

$$\nu_{n+1}(\nu_n, \Sigma_n) = \mathbb{E}\left[x_n + hf(x_n)\right] = \nu_n + h\,\mathbb{E}[\mu(x_n)] \tag{9}$$

The corresponding variance is given by

$$\begin{aligned}\Sigma_{n+1}(\nu_n, \Sigma_n) &= \text{var}\big(x_n + hf(x_n)\big)\\ &= \text{var}(x_n) + h^2\text{var}\big(f(x_n)\big) + 2h\,\text{cov}\big(x_n, f(x_n)\big),\end{aligned} \tag{10}$$

and from the law of total variance follows

$$\text{var}(f(x_n)) = \mathbb{E}\left[\sigma^2(x_n)\right] + \mathbb{E}\left[\mu(x_n)^2\right] - \mathbb{E}\left[\mu(x_n)\right]^2 \tag{11}$$

This leaves the covariance between X_n and $f(X_n)$. In [7], the authors include a linearized expression in the footnote. In [8], the authors suggest

$$\text{cov} = (x_n, f(x_n)) = \mathbb{E}\left[x_n \mu(x_n)\right] - \nu_n \mathbb{E}\left[\mu(x_n)\right], \tag{12}$$

which results in

$$\begin{aligned} \Sigma_{n+1}(\nu_n, \Sigma_n) =& \Sigma_n + h^2\big(\mathbb{E}[\sigma^2(x_n)] + \mathbb{E}[\mu(x_n)^2] - \mathbb{E}[\mu(x_n)]^2\big) \\ &+ 2h\big(\mathbb{E}[x_n\mu(x_n)] - \nu_n\mathbb{E}[\mu(x_n)]\big) \end{aligned} \tag{13}$$

To compute values for the mean $m(\nu_{n+1}, \Sigma_{n+1})$ and the variance $v(\nu_{n+1}, \Sigma_{n+1})$, we need expressions for the expected values $\mathbb{E}[\mu(x_n)]$, $\mathbb{E}[\sigma^2(x_n)]$, $\mathbb{E}[\mu(x_n)^2]$ and $\mathbb{E}[x_n\mu(x_n)]$. For the first three terms, we can find analytical expressions [12] for the squared exponential kernel, and approximate expressions using linearization for all other kernels in [7]. In [8] we also find the previously listed expressions for the squared exponential kernel and in addition an expression for the term $\mathbb{E}[\mu(x_n)]$ for the covariance.

However, we will demonstrate in the next section that this approach makes an incorrect implicit assumption about the independence of the states X_n and the model f.

3.2 A linear model

Consider a very simple model, a distribution of stable linear functions of the form

$$f(x) = -ax + B, \quad \text{with } a \in \mathbb{R}^+,\ B \sim \mathcal{N}(0, \beta), \tag{14}$$

as shown in Fig. 1, which means that

$$\mu(x) = -ax \quad \text{and} \quad \sigma^2(x) = \beta. \tag{15}$$

The integrals from the previous sections resolve to

$$\mathbb{E}_{X_n}[\mu(x_n)] = -a\nu_n \tag{16a}$$

$$\mathbb{E}_{X_n}[\sigma^2(x_n)] = \beta \tag{16b}$$

$$\mathbb{E}_{X_n}[\mu(x_n)^2] = a^2(\Sigma_n + \nu_n^2) \tag{16c}$$

$$\mathbb{E}_{X_n}[x_n\mu(x_n)] = -a(\Sigma_n + \nu_n^2), \tag{16d}$$

which results in the iterations

$$\nu_{n+1} = (1 - ah)\nu_n \tag{17a}$$

$$\Sigma_{n+1} = \Sigma_n + h^2\beta + h^2a^2\Sigma_n - 2ha\Sigma_n. \tag{17b}$$

In addition, the analytical flow for (14) is

$$\varphi^t X_0 = \mathrm{e}^{-at} X_0 + \frac{B}{a}(1 - \mathrm{e}^{-at}), \tag{18}$$

which has the time-dependent mean and variance

$$\nu_t = \mathbb{E}[\varphi^t X_0] = \mathrm{e}^{-at} \mu_0 \tag{19a}$$

$$\Sigma_t = \text{var}(\varphi^t X_0) = \mathrm{e}^{-2at} \Sigma_0 + \frac{\beta}{a^2}(1 - \mathrm{e}^{-at})^2. \tag{19b}$$

Here, we assume that the initial value and the model parameter B are independent, so $\text{cov}(X_0, B) = 0$.

Note that the variance from the Euler step (17b) and the exact solution (19b) do not have the same fixed points:

$$\hat{\Sigma}^{\text{exact}} = \frac{\beta}{a^2}, \tag{20}$$

$$\hat{\Sigma}^{\text{euler}} = \frac{\beta}{a^2}\left(\frac{ah}{2 - ah}\right) \tag{21}$$

The result for $\hat{\Sigma}^{\text{exact}}$ matches the variance of the fixed point distribution of the model (14).

The Euler steps converge to a fixed point that depends on the step size h, as we demonstrate in Fig. 1. This is problematic, as the dynamics should not depend on solver parameters. Remarkably, we obtain a similar behaviour if we consider (19b) as an iterable function of X_i and step size h, which has the fixed point

$$\hat{\Sigma}^{\text{iter. flow}} = \frac{\beta}{a^2} \tanh\left(\frac{ah}{2}\right) \tag{22}$$

This is surprising, as the flow map should have the semi-group property, such that $\varphi^t \circ \varphi^s X_0 = \varphi^{t+s} X_0$. Therefore, there should be no difference between applying the flow over multiple smaller intervals compared to one large interval.

Taking a closer look at (18), we note that it is made up of two terms. For stable systems, the first term captures the *transient* effect of the initial value, whereas the second term represents the *asymptotic* behaviour determined by the model parameter B. If we apply the flow to some state X_n instead of the initial value X_0, the state has already been affected by the model and some decay of the initial value has occurred. Thus, the iterative schemes mentioned above are equivalent to starting over with an independent initial value after each time step. This can be seen in more detail in Fig. 1.

The repeated transient explains the additional factor in $\hat{\Sigma}^{\text{iter. flow}}$. As $ah \to \infty$, we see either increasingly large steps or increasingly fast dynamics, which means that the transient behaviour fully concludes in each step. The additional factor in $\hat{\Sigma}^{\text{euler}}$ also depends on ah. However, the expression breaks down as $ah \to 2$, which matches the stability region of the explicit Euler method.

Therefore, the issue lies in the covariance between the states and the model parameter. While it is valid to assume that $\text{cov}(X_0, B) = 0$, all subsequent states X_i depend on the model, specifically this case on the random model parameter B, which means that they are not independent. Instead, we find

$$\text{cov}(X_n, B) = \text{cov}\left(\mathrm{e}^{-anh} X_0 + \frac{B}{a}(1 - \mathrm{e}^{-anh}),\, B\right) = \frac{\beta}{a}(1 - \mathrm{e}^{-ahn}) \tag{23}$$

Appending this to (19b) yields the correct result, as demonstrated on the right side of Fig. 1.

If X_n is the result of subsequent Euler steps,

$$X_{n+1} = X_n + h(-aX_n + B) = (1 - ah)X_n + hB \tag{24}$$

we find a telescope sum, such that

$$\begin{aligned} &\text{cov}(X_n, B) = \text{cov}\left((1 - ah)X_{n-1} + hB,\, B\right) \\ &= (1 - ah)\text{cov}(X_{n-1}, B) + h\,\text{cov}(B, B) = \dots \\ &= (1 - ah)^n \underbrace{\text{cov}(X_0, B)}_{0} + h \sum_{i=0}^{n-1} (1 - ah)^{n-1-i} \underbrace{\text{cov}(B, B)}_{\beta}. \end{aligned} \tag{25}$$

which results in

$$\Sigma_{n+1} = s(1 - ah)^2 \Sigma_n + h^2 \beta + 2h^2(1 - ah)\,\text{cov}(X_n, B) \tag{26}$$

Using this expression, we find that the variance from the iterative solutions matches the analytic solution and the variance of the sampled solutions (see Fig. 1).

This result is noteworthy beyond the continuous dynamics discussed in this work. Applying the flow map iteratively without the additional covariance term for some fixed step size h is equivalent to repeatedly applying the learned flow map in the state-space GP context. As this phenomenon is thus far unaddressed in previous work using moment matching for multiple step ahead predictions, the uncertainty has likely been underestimated. However, as the effect depends on the implicit step size h in each step of the flow and might be affected by the inclusion of multiple previous points in the auto-regression context, further study is needed to understand the scope.

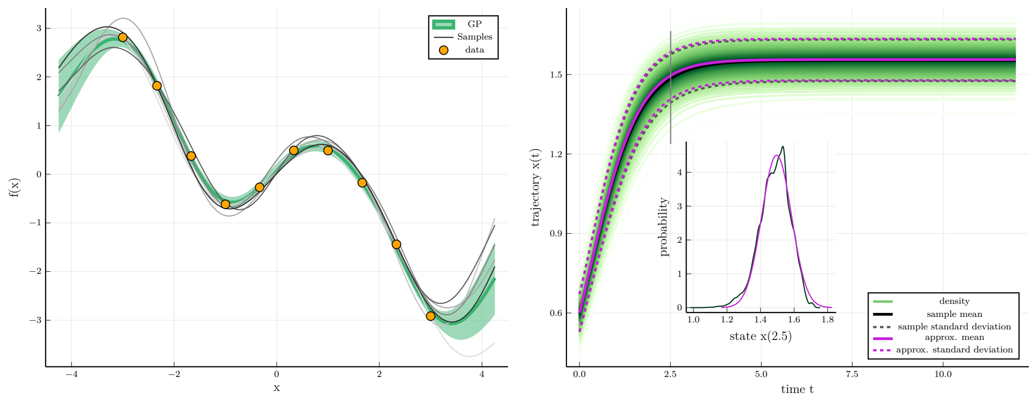

Figure 2: Nonlinear example function: *Left:* The example function, including the noisy data points, the mean and the variance of the GP and a few samples from the GP distribution. *Right:* The distribution of the trajectories resulting from the GP samples, as well as their mean and standard deviation. We compare with the results from the approximate local linearization-based solver.

4 Local Linearization

We demonstrated in the previous section that we have to take the history of each state into account to take another iteration step. However, doing so for a general nonlinear GP model is substantially more complicated.

To address this, instead of approximating the output distribution of the exact GP, we propose a piece-wise linear approximation of the GP by defining a series of linear ODEs. Therefore, approximate the model as a whole with a model for which we can compute the exact state distributions.

We define

$$f_i(x) = a_i x + B_i, \quad X(0) = X_i,\ t \in [t_i, t_i + h), \tag{27}$$

where

$$a_i = \mu_f(\nu_i), \text{ and } B_i = f(\nu_i) - a_i X_i, \tag{28}$$

which match (14) as we have

$$B_i \sim \mathcal{N}\left(\mu_f(\nu_i) - a_i\nu_i,\, \sigma_f^2(\nu_i)\right). \tag{29}$$

While it would be more accurate to have a_i as a random variable, similar to Nghiem's *linGP* [10], this would introduce the product of two random variables and additional covariances.

With this approximation we can take Euler steps on the linear system, resulting in the iterative scheme

$$\nu_{n+1} = \nu_x + h\mu_f(\nu_n) \tag{30a}$$

$$\Sigma_{n+1} = (1 + a_n h)^2 \Sigma_n + h^2\sigma_f^2(\nu_n) + 2h(1 + a_n h)\operatorname{cov}(X_n, B_n). \tag{30b}$$

Our approximation allows us to find an expression for the covariance in (30b) using a similar tele-scope sum as in (25). This results in

$$\operatorname{cov}(X_n, B_n) = h \sum_{i=0}^{n-1} \prod_{j=i+1}^{n-1} (1 + a_j h) \operatorname{cov}(\nu_i, \nu_n) \tag{31}$$

This expression requires storing all previous a_i, but since the trajectory states X_i already need to be stored, the solver still has only linearly scaling storage requirements. The bigger computational challenge is computing the covariances $\operatorname{cov}(\nu_i, \nu_n)$ between the current state and all previous ones. This is done via (4b) which causes quadratic scaling with the number of steps already taken.

Fortunately, there are two possible justifications to truncate the series in (31). Firstly, in practice, we often operate within the basin of attraction of a fixed point. Then it will generally be the case that

$a_i < 0$ and $\prod_{j=i+1}^{n-1}(1 + a_j h) \to 0$ for $n \to \infty$. Secondly, when using a stationary kernel for the GP, the covariance is determined by the distance of the two points and a length scale, where distant points are assumed to be nearly uncorrelated. Hence, we will generally find $\mathrm{cov}(\nu_i, \nu_n) \approx 0$ for $i \ll n$.

Both terms have complementary behaviour. Under stable dynamics, successive x_n will be close and therefore correlated, but since we have $a_i < 0$ we can truncate based on the first term. Under unstable dynamics, we will have $a_i > 0$, but successive points will be further apart and the covariance term decreases faster.

As the terms of the series are simple to compute, it is possible to implement adaptive truncation based on the smallest term in the sum. We also note that we have made no assumption about the underlying kernel beyond stationarity.

5 Numerical Experiments

To demonstrate the effectiveness of the local linearization, we consider the ODE

$$\dot{x} = g(x) = x\,cos(x), \tag{32}$$

and sample 10 data points in the interval $[-4, 4]$ and condition a GP with zero mean and the squared exponential kernel (5). Via (6), we generate 3000 samples (as seen on the left in Fig. 2) and integrate each one with initial values from 150 samples of the input distribution $\mathcal{N}(0.6, 0.005)$, resulting in the distribution of trajectories shown on the right of Fig. 2.

We then apply the local-linearization Euler's method to the same GP, starting from the same initial distribution, and see that the results agree very well.

However, using the full history of past states makes this method very expensive. Therefore, we consider using only a truncated version of the sum in (31) with only a limited buffer of past slopes a_i, as discussed in the previous section, and also highlight the effect of different step sizes. The results in Fig. 2 show that the behaviour of the mean is as expected for Euler's method, the error decreases with a smaller step size.

We also see that if the buffer is insufficient, the variance is again step size dependent, but increasing the buffer results in convergence towards the correct value. While a smaller step size does decrease the error in both the mean and the variance, the quadratic scaling of the computational cost for each step with the buffer size, as a result of (4b), provides an incentive to maximize the step size.

Our solver returns accurate results as the estimated trajectory distribution converges towards a deterministic, model-dependent fixed point distribution. While the results can still be affected by solver parameters, their effect is understood and can be mitigated by adjusting the parameters.

Our slightly underestimates variance compared to the ground truth sampling, likely due to linearization error. One option to improve the piecewise linear approximation might be incorporating the work of Girard et al. [7] in a way that correctly incorporates the correlation of subsequent states. In this context, we note that using (13) is conceptually equivalent to using our solver with zero buffer.

5.1 Limitations

The linearization-based approach detailed in this work has substantially lower computational cost than a sampling-based approach, but has a fundamental limitation. While linear models only have a single fixed point, nonlinear models introduce a wide array of additional behaviours, like additional fixed points and complex basins of attraction.

In the context of this section's model (32), we note that there is an unstable fixed point in 0, and trajectories starting to the right of it will converge towards $\pi/4$, as shown in Fig. 3. However, trajectories starting on the left of 0 will converge towards $-\pi/4$. This means that starting with a distribution of initial values near 0, the distribution of trajectories will be bi-modal, as shown in Fig. 4. This behaviour fundamentally cannot be captured by a linear approximation that assumes a uni-modal distribution for all states.

However, Fig. 4 highlights the usefulness of a cheap approximate solver. Only solving the ODE defined by the GP mean would completely miss the presence and effects of an added fixed point.

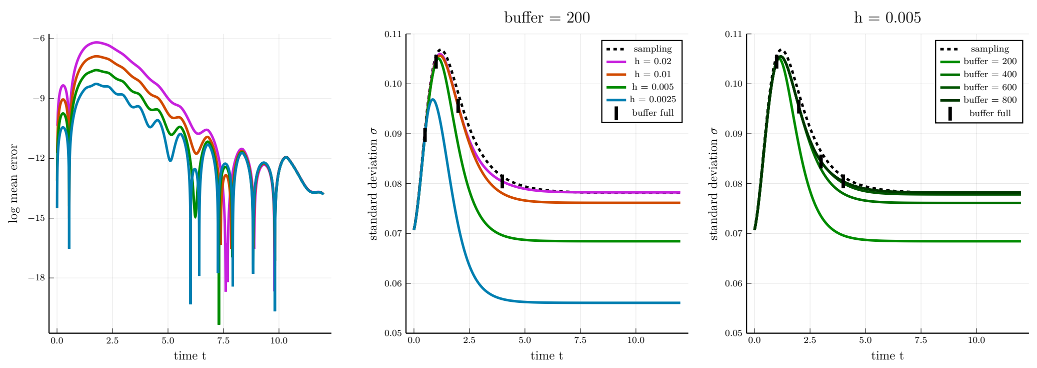

Figure 3: Local Linearization Solver: *Left:* The logarithmic error of the solver compared to the solution of the ODE defined by the GP mean. Decreasing the step size also decreases the mean error. *Middle:* The results from the Local Linearization solver. With an insufficient buffer size the standard deviation shows step size dependence. We note that the standard deviation begins to diverge once the buffer fills up. *Right:* With increasing buffer size for a given step size the standard deviation converges towards the result from sampling.

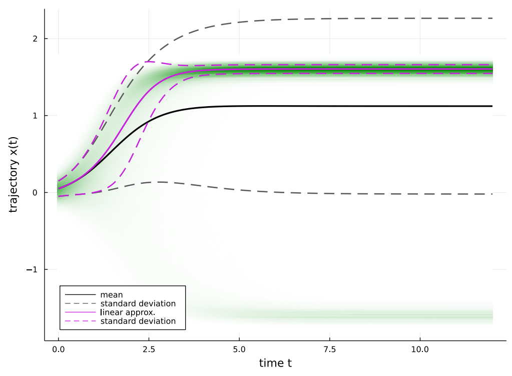

Figure 4: Nonlinear behaviours: Starting at $X_0 \sim \mathcal{N}(0.05, 0.01)$, some combinations of initial value and function samples converge to a different fixed point, creating a bi-modal distribution.

Our solver converges to the sensible fixed point, as the initial value distribution has its mean to the right of 0, but the variance indicates increased initial uncertainty compared to the previous result from Fig. 3.

6 Conclusion

Accurately evolving dynamical systems expressed by GPs, either by numerically integrating a continuous vector field or subsequently applying a learned flow map, introduces a correlation between the distribution of functions expressed by the GP and subsequent states. We show that current approaches incorrectly assume independence between the distribution of the states and the GP distribution, resulting in an incorrect step size dependence of the trajectory variance.

We derive and demonstrate the correct correlation for a simple linear model, and leverage those findings to combine local linearization and the explicit Euler method, resulting in a computationally efficient and accurate solver.

A possible application for this solver lies in creating a likelihood-based cost function for training GP-based models, as an alternative to multiple shooting methods [14]. Another option is using it to make ahead of time predictions for model predictive control and take the prediction uncertainty into account.

Our results highlight a fundamental consideration for solving dynamical system and creates scope for several extensions. This includes improving the linear approximation, combining local linearization with higher order methods than the explicit Euler, and studying the effect of incorporating the correct correlation in the state-space GP context.

References

[1] H. T. Banks and S. Hu. *Uncertainty Propagation and Quantification in a Continuous Time Dynamical System*. 2012.

[2] S. Ridderbusch, C. Offen, S. Ober-Blöbaum, and P. Goulart. "Learning ODE Models with Qualitative Structure Using Gaussian Processes". *2021 60th IEEE Conference on Decision and Control (CDC)*. 2021 60th IEEE Conference on Decision and Control (CDC). Dec. 2021.

[3] C. Rackauckas, Y. Ma, J. Martensen, C. Warner, K. Zubov, R. Supekar, D. Skinner, and A. Ramadhan. "Universal Differential Equations for Scientific Machine Learning". Jan. 13, 2020. arXiv: `2001.04385`.

[4] M. Schober, S. Särkkä, and P. Hennig. "A Probabilistic Model for the Numerical Solution of Initial Value Problems". *Statistics and Computing* (Jan. 1, 2019).

[5] S. Kamthe and M. P. Deisenroth. "Data-Efficient Reinforcement Learning with Probabilistic Model Predictive Control". June 20, 2017. arXiv: `1706.06491`.

[6] M. Buisson-Fenet, F. Solowjow, and S. Trimpe. "Actively Learning Gaussian Process Dynamics". *Proceedings of the 2nd Conference on Learning for Dynamics and Control*. Proceedings of Machine Learning Research. The Cloud: PMLR, June 10–11, 2020.

[7] A. Girard, C. E. Rasmussen, and R. Murray-Smith. "Multiple-Step Ahead Prediction for Non Linear Dynamic Systems – A Gaussian Process Treatment with Propagation of the Uncertainty". *Advances in Neural Information Processing Systems* (2003).

[8] P. Groot, P. Lucas, and P. Bosch. "Multiple-Step Time Series Forecasting with Sparse Gaussian Processes". *http://allserv.kahosl.be/bnaic2011/* (2011).

[9] J. Kocijan. *Modelling and Control of Dynamic Systems Using Gaussian Process Models*. Advances in Industrial Control. Springer International Publishing, 2015.

[10] T. X. Nghiem. "Linearized Gaussian Processes for Fast Data-driven Model Predictive Control". Oct. 1, 2019. arXiv: `1812.10579 [cs]`.

[11] K. Ensinger, F. Solowjow, S. Ziesche, M. Tiemann, and S. Trimpe. "Structure-Preserving Gaussian Process Dynamics". Jan. 9, 2022. arXiv: `2102.01606 [cs]`.

[12] A. Girard, C. Rasmussen, J. Q. Candela, and R. Murray-Smith. "Gaussian Process Priors with Uncertain Inputs Application to Multiple-Step Ahead Time Series Forecasting". *Advances in Neural Information Processing Systems* (2002).

[13] L. Hewing, E. Arcari, L. P. Fröhlich, and M. N. Zeilinger. "On Simulation and Trajectory Prediction with Gaussian Process Dynamics". *Proceedings of the 2nd Conference on Learning for Dynamics and Control*. Proceedings of Machine Learning Research. The Cloud: PMLR, June 10–11, 2020.

[14] P. Hegde, Ç. Yildiz, H. Lähdesmäki, S. Kaski, and M. Heinonen. *Variational Multiple Shooting for Bayesian ODEs with Gaussian Processes*. Jan. 26, 2022. arXiv: `2106.10905 [cs, stat]`.

[15] C. E. Rasmussen and C. K. I. Williams. *Gaussian Processes for Machine Learning*. Adaptive Computation and Machine Learning. Cambridge, Mass: MIT Press, 2006. 248 pp.

[16] M. A. Alvarez, L. Rosasco, and N. D. Lawrence. "Kernels for Vector-Valued Functions: A Review". Apr. 16, 2012. arXiv: `1106.6251 [cs, math, stat]`.

[17] J. T. Wilson, V. Borovitskiy, A. Terenin, P. Mostowsky, and M. P. Deisenroth. *Efficiently Sampling Functions from Gaussian Process Posteriors*. Aug. 16, 2020. arXiv: `2002.09309 [cs, stat]`.

[18] A. Rahimi and B. Recht. "Random Features for Large-Scale Kernel Machines". *Advances in Neural Information Processing Systems*. Curran Associates, Inc., 2007.