51st IEEE Conference on Decision and Control
December 10-13, 2012. Maui, Hawaii, USA

Modelling and Control of Nonlinear Systems using Gaussian Processes with Partial Model Information

Joseph Hall, Carl Rasmussen and Jan Maciejowski

***Abstract*— Gaussian processes are gaining increasing popularity among the control community, in particular for the modelling of discrete time state space systems. However, it has not been clear how to incorporate model information, in the form of known state relationships, when using a Gaussian process as a predictive model. An obvious example of known prior information is position and velocity related states. Incorporation of such information would be beneficial both computationally and for faster dynamics learning. This paper introduces a method of achieving this, yielding faster dynamics learning and a reduction in computational effort from $\mathcal{O}(Dn^2)$ to $\mathcal{O}((D-F)n^2)$ in the prediction stage for a system with D states, F known state relationships and n observations. The effectiveness of the method is demonstrated through its inclusion in the PILCO learning algorithm with application to the swing-up and balance of a torque-limited pendulum and the balancing of a robotic unicycle in simulation.**

I. Introduction

Gaussian Processes (GPs) are a powerful modelling framework that incorporate modelling uncertainty in predictions in a systematic manner. They were introduced into the machine learning community by [1] where their use is now widespread. See [2] for an introduction to GPs in this area. The use of Gaussian processes in system identification for control has been limited, yet has produced some remarkable results in contexts where conventional methods have struggled or failed. Examples of this include learning an accurate control policy for an autonomous blimp [3], the swing up and balance of a pendulum on a moving cart, and balancing a full nonlinear simulated model of a robotic unicycle [4].

The framework of Gaussian processes is effective since very few prior assumptions are placed on the nature of the dynamics (smoothness and stationarity of the underlying function for example) and it is left to the model to infer all dependencies between states. Although appealing for its generality, partial knowledge of the system dynamics may be available and a framework for including this as prior information would be useful both computationally and statistically. An obvious example of this is states that are related through time derivatives e.g. positions and velocities. The contribution of this paper is to provide such a method for the case of known (possibly uncertain) state relationships.

The problem under consideration is to infer a model for the discrete time dynamical system $\Delta\mathbf{x}_k = \mathbf{f}(\mathbf{x}_{k-1}, \mathbf{u}_{k-1}) + \boldsymbol{\epsilon}$ with state $\mathbf{x} \in \mathbb{R}^D$, input $\mathbf{u} \in \mathbb{R}^E$ and process noise $\boldsymbol{\epsilon} \sim \mathcal{N}$. This can be achieved using a Gaussian process model trained with n observations of state-input pairs $(\mathbf{x}_{k-1}, \mathbf{u}_{k-1})$ and the associated change in state $\Delta\mathbf{x}_k = \mathbf{x}_k - \mathbf{x}_{k-1}$. The trained GP can then make predictions of system trajectories along with a measure of the uncertainty. In this paper it is assumed that the state consists of $\mathbf{s} \in \mathbb{R}^{D-F}$, $\hat{\mathbf{s}} \in \mathbb{R}^F$. Given a set of known relationships $\Delta\hat{\mathbf{s}}_k = \mathbf{h}(\mathbf{x}_{k-1}, \mathbf{u}_{k-1}, \Delta\mathbf{s}_k) + \boldsymbol{\kappa}$, with uncertainty parameter $\boldsymbol{\kappa}$, a method of including this information in prediction has been derived. The method yields a reduction in the computational requirement for prediction from $\mathcal{O}(Dn^2)$ to $\mathcal{O}((D-F)n^2)$. Particular focus has been given to deriving a method for including approximate time derivative information. Within this scheme, a means of learning the accuracy of the approximation, given the observed data, has been derived.

In the remainder of the paper, Section II will provide the basic theory of Gaussian process modelling of discrete-time state space systems. Section III presents the method of including known state relationships in GP predictions, the core contribution of this paper. In Section IV this framework is applied to the case of time-derivative-related states, and outlines an inference scheme for calculating its predictive accuracy given the data. Section V gives an overview of the PILCO algorithm of [4] in which this approximation has been employed in a control policy learning framework. Section VI presents simulation results for the application of learning control to a simulated torque-limited pendulum swing-up problem and a full nonlinear unicycle balancing problem.

II. Gaussian Processes

A. Prior to Posterior

To introduce the general concept of training and prediction with a Gaussian process model, consider the problem of inferring the underlying function of a noisy static map $y = f(\mathbf{z}) + \epsilon$ with $\mathbf{z} \in \mathbb{R}^A$ and $\epsilon \sim \mathcal{N}(0, \sigma_\epsilon^2)$ from training data. The training data set $\mathcal{D} = \{\mathbf{Z}, \mathbf{y}\}$ consists of regression vectors $\mathbf{Z} = [\mathbf{z}_1 \ldots \mathbf{z}_n]^\top \in \mathbb{R}^{n\times A}$ and the corresponding observed outputs $\mathbf{y} = [y_1 \ldots y_n]^\top \in \mathbb{R}^n$. A GP can be viewed as an extension of the multivariate Gaussian distribution to infinitely long vectors, or functions. Formally, GPs are defined by [2] as *a collection of random variables, any finite number of which have a joint Gaussian distribution.* This can be expressed

$$f|\boldsymbol{\theta} \sim \mathcal{GP}(m, k), \tag{1}$$

J. Hall, C. Rasmussen and J. Maciejowski are with Deparment of Engineering, University of Cambridge, UK {jah215, cer54}@cam.ac.uk, and jmm@eng.cam.ac.uk

where f is the random process and $\boldsymbol{\theta}$ is the hyperparameter vector parameterising the mean function m, and the covariance function k. Informally, GPs are a means of defining a prior distribution over a space of functions. Now, a common choice of prior is a zero mean with squared exponential covariance function

$$k(\mathbf{z},\tilde{\mathbf{z}}) = \alpha^2 \exp\left(-\tfrac{1}{2}(\mathbf{z}-\tilde{\mathbf{z}})^\top \boldsymbol{\Lambda}^{-1}(\mathbf{z}-\tilde{\mathbf{z}})\right), \tag{2}$$

where $\boldsymbol{\Lambda} = \mathrm{diag}\left(\lambda_1^2 \ldots \lambda_D^2\right)$ and $\mathbf{z}$, $\tilde{\mathbf{z}}$ are two arbitrary points in the regressor space. The covariance matrix $\mathbf{K} := \mathbf{K}(\mathbf{Z},\mathbf{Z}) \in \mathbb{R}^{n\times n}$ and the vector $\mathbf{k}(\mathbf{z}) := \mathbf{k}(\mathbf{Z},\mathbf{z}) \in \mathbb{R}^n$ are defined to have elements $K_{ij} = k(\mathbf{z}_i,\mathbf{z}_j)$ and $k_i = k(\mathbf{z}_i,\mathbf{z})$ respectively.

Given the training data $\mathcal{D}$, the problem is to find the posterior distribution $p(f|\mathcal{D},\boldsymbol{\theta})$ over the space of functions. The relationship follows directly from Bayes' rule

$$p(f|\mathcal{D},\boldsymbol{\theta}) = \frac{p(\mathbf{y}|f,\mathbf{Z},\boldsymbol{\theta})}{p(\mathbf{y}|\mathbf{Z},\boldsymbol{\theta})}\,p(f|\boldsymbol{\theta}), \tag{3}$$

since $f|\boldsymbol{\theta} \sim \mathcal{GP}$ is independent of $\mathbf{Z}$. It turns out that this posterior distribution is also a Gaussian process $f|\mathcal{D},\boldsymbol{\theta} \sim \mathcal{GP}(m_+, k_+)$ with mean and covariance function

$$m_+(\mathbf{z}) = m(\mathbf{z}) + \mathbf{k}(\mathbf{z})^\top\left(\mathbf{K}+\sigma_\epsilon^2\mathbf{I}\right)^{-1}\left(\mathbf{y}-m(\mathbf{Z})\right), \tag{4}$$

$$k_+(\mathbf{z},\tilde{\mathbf{z}}) = k(\mathbf{z},\tilde{\mathbf{z}}) - \mathbf{k}(\mathbf{z})^\top\left(\mathbf{K}+\sigma_\epsilon^2\mathbf{I}\right)^{-1}\mathbf{k}(\tilde{\mathbf{z}}). \tag{5}$$

This can be found by considering the joint Gaussian distribution $p\left(\mathbf{y}, f(\mathbf{z}), f(\tilde{\mathbf{z}})|\boldsymbol{\theta}\right)$ and conditioning on the observed data $\mathbf{y}$.

B. Prediction

Prediction of a single functional output $f(\mathbf{z})$ under the posterior GP is simply given by the Gaussian distribution $f(\mathbf{z}) \sim \mathcal{N}\left(m_+(\mathbf{z}), k_+(\mathbf{z},\mathbf{z})\right)$. The computation of the mean and variance prediction is of the order $\mathcal{O}\left(n^2\right)$.

To make predictions of functions with multivariate outputs one can either train a full multivariate GP (an extension of the concepts above) or one can simply train an independent model for each output dimension. In most cases the benefits of the increased predictive accuracy gained by a full multivariate GP is outweighed by the much larger computational effort required in training and prediction.

If the test input is now itself a random variable $\mathbf{z} \sim \mathcal{N}(\boldsymbol{\mu},\boldsymbol{\Sigma})$ then the output distribution of the GP will no longer be Gaussian, as illustrated in Fig. 1. However, the mean and variance can be evaluated analytically (given a squared exponential covariance function) therefore the output distribution can be approximated by a Gaussian using exact moment matching using the method of [5].

C. Modelling State Space Systems

Now in order to use GPs to model the nonlinear system $\Delta\mathbf{x}_k = \mathbf{f}(\mathbf{x}_{k-1},\mathbf{u}_{k-1}) + \boldsymbol{\epsilon}$ with system state $\mathbf{x} \in \mathbb{R}^D$ and input $\mathbf{u} \in \mathbb{R}^E$ where $\mathbf{x}_k = \Delta\mathbf{x}_k + \mathbf{x}_{k-1}$, simply use training data of the form $\mathbf{Z} = \left\{[\mathbf{x}_{k_i-1};\mathbf{u}_{k_i-1}]\right\}$, $\mathbf{y} = \left\{\Delta\mathbf{x}_{k_i}\right\}$, $i \in \{1..n\}$. Note that it is more appropriate to infer state differences instead of the next states themselves given a zero-mean prior over the space of functions. Prediction then has computation order of $\mathcal{O}\left(Dn^2\right)$ given D independent GP models.

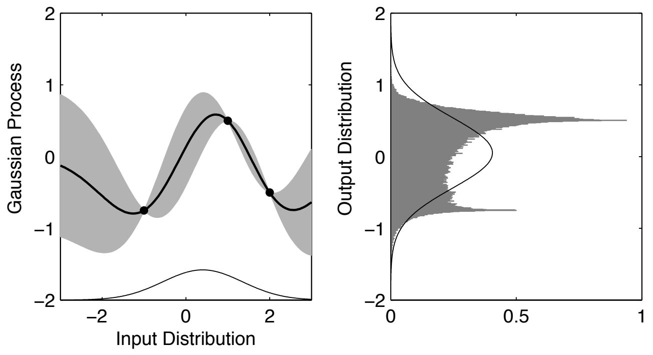


Fig. 1. Illustration of the exact moment matching scheme for propagation of uncertainty through a Gaussian process model. The left hand plot depicts the input distribution and the GP model where the black dots denote training data $\mathcal{D}$ and the shaded region in the left hand plot shows the 95% confidence region. The right hand plot shows the Gaussian approximation for the output distribution and an estimate of the output distribution obtained by simple Monte Carlo sampling.

D. Training

Ideally, the full posterior $p(f|\mathcal{D})$ would be obtained by integrating over all possible $\boldsymbol{\theta}$ values with respect to some prior $p(\boldsymbol{\theta})$. This would only be possible through the use of computationally demanding sampling methods. Alternatively, a point estimate can be obtained through maximisation of the marginal likelihood $p(\mathbf{y}|\mathbf{Z},\boldsymbol{\theta}) = \mathcal{N}\left(\mathbf{0},\mathbf{K}+\sigma_\epsilon^2\mathbf{I}\right)$. This is an appropriate metric to optimise since it is equivalent to finding the “maximum a posteriori" estimate $\max p(\boldsymbol{\theta}|\mathcal{D})$ given a “flat" prior $p(\boldsymbol{\theta})$. This optimisation procedure constitutes the training of the hyperparameters.

III. Known State Relationships

A. General Relationships

Consider a system of the form $\Delta\mathbf{x}_k = \mathbf{f}(\mathbf{x}_{k-1},\mathbf{u}_{k-1}) + \boldsymbol{\epsilon}$ as before but with a system state $\mathbf{x} = [\mathbf{s};\hat{\mathbf{s}}]$, which is made up of states $\mathbf{s} \in \mathbb{R}^{D-F}$ and $\hat{\mathbf{s}} \in \mathbb{R}^F$. There is a known relationship between $\mathbf{s}$ and $\hat{\mathbf{s}}$ of the form

$$\Delta\hat{\mathbf{s}}_k = \mathbf{h}\left(\mathbf{x}_{k-1},\mathbf{u}_{k-1},\Delta\mathbf{s}_k\right) + \boldsymbol{\kappa}, \tag{6}$$

with noise term $\boldsymbol{\kappa} \sim \mathcal{N}(\mathbf{0},\boldsymbol{\Sigma}_{\boldsymbol{\kappa}})$ where $\boldsymbol{\Sigma}_{\boldsymbol{\kappa}} = \mathrm{diag}(\sigma_{\kappa a}^2)$, $a \in \{1..F\}$ can represent uncertainty in the relationship. With this information, only $D-F$ Gaussian process models need be trained to predict $\Delta\mathbf{s}_k \sim \mathcal{GP}$ and the remaining state differences $\Delta\hat{\mathbf{s}}_k$ can be reconstructed from Eq. 6. Uncertainty can be propagated through the state predictions provided the relationship $\mathbf{h}(\cdot)$ is of a form that will now be discussed.

B. Prediction for Uncertain State-Input Pairs

In order to propagate uncertainty through predictions, a framework is required where given a Gaussian distribution $\mathbf{x}_{k-1},\mathbf{u}_{k-1} \sim \mathcal{N}$, the posterior distribution $p(\mathbf{x}_k)$ can be approximated analytically. This will be possible provided that the mean $\mathbb{E}[\mathbf{h}|\boldsymbol{\mu},\boldsymbol{\Sigma}]$ and covariance $\mathrm{cov}[\mathbf{h}|\boldsymbol{\mu},\boldsymbol{\Sigma}]$ given $\mathbf{x}_{k-1},\mathbf{u}_{k-1},\Delta\mathbf{s}_k \sim \mathcal{N}(\boldsymbol{\mu},\boldsymbol{\Sigma})$ can be evaluated analytically. Given this, the output distribution can be approximated using exact moment matching as discussed in Section II-B. The procedure will then be as follows:

- **Gaussian Process** $p(\mathbf{x}_{k-1},\mathbf{u}_{k-1}) \to p(\mathbf{x}_{k-1},\mathbf{u}_{k-1},\Delta\mathbf{s}_k)$. Achieved using the framework of [5].
- **State Relationships** $p(\mathbf{x}_{k-1},\mathbf{u}_{k-1},\Delta\mathbf{s}_k) \to p(\mathbf{x}_{k-1},\Delta\mathbf{x}_k)$. Dependant on the structure of $\mathbf{h}(\cdot)$.
- **Next State** $p(\mathbf{x}_{k-1},\Delta\mathbf{x}_k) \to p(\mathbf{x}_k)$. Follows from their linear relationship.

C. Linear Relationships

If the known state relationships are of a linear form $\Delta\hat{\mathbf{s}}_k = \mathbf{M}\boldsymbol{\xi}_{k-1} + \boldsymbol{\kappa}$ for some mapping $\mathbf{M}$ and input vector $\boldsymbol{\xi}_{k-1} := [\mathbf{x}_{k-1};\mathbf{u}_{k-1};\Delta\mathbf{s}_k]$ then propagating the uncertainty is simple. Given $\boldsymbol{\xi}_k \sim \mathcal{N}(\boldsymbol{\mu},\boldsymbol{\Sigma})$, the joint distribution is

$$\begin{bmatrix}\boldsymbol{\xi}_{k-1}\\ \Delta\hat{\mathbf{s}}_k\end{bmatrix} \sim \mathcal{N}\left(\begin{bmatrix}\boldsymbol{\mu}\\ \mathbf{M}\boldsymbol{\mu}\end{bmatrix},\begin{bmatrix}\boldsymbol{\Sigma} & \boldsymbol{\Sigma}\mathbf{M}^\top\\ \mathbf{M}\boldsymbol{\Sigma} & \mathbf{M}\boldsymbol{\Sigma}\mathbf{M}^\top + \boldsymbol{\Sigma}_{\boldsymbol{\kappa}}\end{bmatrix}\right), \tag{7}$$

and the distribution $\mathbf{x}_{k-1},\Delta\mathbf{x}_k \sim \mathcal{N}$ can be lifted straight out. A useful example of such a relationship will be developed in the following section.

IV. Postion-Velocity Relationships

A. Constant Acceleration Approximation

Consider the common case that the system state is made up of states $\mathbf{s} \in \mathbb{R}^{D/2}$ and their time derivatives $\dot{\mathbf{s}}$. The exact relationship between *position* states $\mathbf{s}$ and the associated *velocity* states $\dot{\mathbf{s}} := \frac{\mathrm{d}}{\mathrm{d}t}\mathbf{s}$ is given by $\Delta\mathbf{s}_k = \int_{t-\Delta_t}^{t}\dot{\mathbf{s}}(\tau)\mathrm{d}\tau$ where $\mathbf{s}_k = \mathbf{s}(t)$ and $\mathbf{s}_{k-1} = \mathbf{s}(t-\Delta_t)$. If the acceleration $\ddot{\mathbf{s}}$ over the time interval $\tau \in [t-\Delta_t, t]$ is taken to be constant then the relationship becomes

$$\Delta\mathbf{s}_k = \Delta_t\left(\tfrac{1}{2}\Delta\dot{\mathbf{s}}_k + \dot{\mathbf{s}}_{k-1}\right) \quad \text{or} \quad \Delta\dot{\mathbf{s}}_k = \tfrac{2}{\Delta_t}\Delta\mathbf{s}_k - 2\dot{\mathbf{s}}_{k-1}, \tag{8}$$

in terms of the change in position and velocity respectively. These relationships are in a linear form with $\hat{\mathbf{s}} = \mathbf{s}$ and $\mathbf{M} = [\mathbf{0},\Delta_t\mathbf{I},\mathbf{0},\frac{\Delta_t}{2}\mathbf{I}]$ in the case of position reconstruction or $\hat{\mathbf{s}} = \dot{\mathbf{s}}$ and $\mathbf{M} = [\mathbf{0},-2\mathbf{I},\mathbf{0},\frac{2}{\Delta_t}\mathbf{I}]$ in the case of velocity reconstruction.

A graphical depiction of the constant acceleration assumption is shown in Fig. 2. This approximation reduces the number of GP models to be trained from D to F in the general case where there are F time derivative related state pairs and therefore the cost of prediction from $\mathcal{O}(Dn^2)$ to $\mathcal{O}((D-F)n^2)$.

Of course the constant acceleration may be too stringent. In this case one can resort to using additional information from previous timesteps. For example, fitting a quadratic curve to the three points $\dot{\mathbf{s}}_k$, $\dot{\mathbf{s}}_{k-1}$ and $\dot{\mathbf{s}}_{k-2}$. Or even using the additional constraint that the previous area be equal to the previous estimate $\Delta\mathbf{s}_{k-1}$ to fit a cubic curve. However, there is no guarantee that these approximations will perform any better, and may in fact degrade performance so they are not considered here.

In reality there will be some amount of uncertainty in this approximation which will be accounted for using an additive noise term $\boldsymbol{\kappa}$ as in the previous section. The selection of the variance $\boldsymbol{\Sigma}_{\boldsymbol{\kappa}}$ plays a significant role in prediction and policy learning as will be discussed later. A scheme has been devised in order to infer this parameter from the training data set.

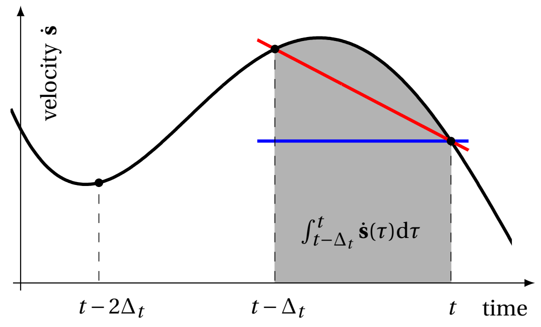

velocity $\dot{\mathbf{s}}$
$\int_{t-\Delta_t}^{t}\dot{\mathbf{s}}(\tau)\mathrm{d}\tau$
$t-2\Delta_t$
$t-\Delta_t$
t
time

Fig. 2. The constant acceleration time integral approximation for reconstructing change in position $\Delta\mathbf{s}_k$ given the velocities $\dot{\mathbf{s}}_k$ and $\dot{\mathbf{s}}_{k-1}$ is given in red. A simple Euler constant velocity approximation is shown in blue.

B. Training

Consider the prediction of a single state pair $(s^a,\dot{s}^a)$ where $a \in \{1..D/2\}$ denotes the relevant dimension. Training data consists of inputs $\mathbf{Z}$ with rows $[\mathbf{s}^\top,\dot{\mathbf{s}}^\top,\mathbf{u}^\top]_{k_i-1}$ and observations $[\mathbf{y}_{\mathrm{p}a},\mathbf{y}_{\mathrm{v}a}]$ with rows $[s^a+e_{\mathrm{p}},\dot{s}^a+e_{\mathrm{v}}]_{k_i}$ where $i \in \{1..n\}$, e_{p} and e_{v} are additive noise terms and the subscripts 'p' and 'v' denote position and velocity respectively. A naive approach to training the hyperparameters would be to train a single Gaussian process on the data set $\{\mathbf{Z},\mathbf{y}_{\mathrm{v}a}\}$ in the case of integral reconstruction or $\{\mathbf{Z},\mathbf{y}_{\mathrm{p}a}\}$ for derivative reconstruction. However, this would ignore half of the available information and provides no framework for training the approximation uncertainty $\sigma^2_{\kappa a}$.

Consider the case of integral reconstruction. The full distribution of the prior belief regarding observed data is given by the multivariate Gaussian

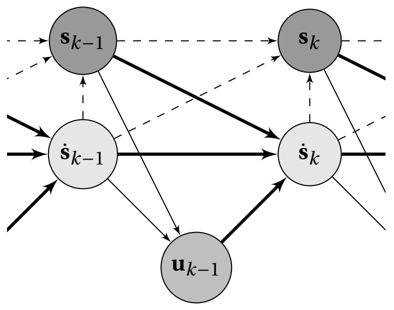
$$\begin{bmatrix}\mathbf{y}_{\mathrm{p}a}\\ \mathbf{y}_{\mathrm{v}a}\end{bmatrix}\Bigg|\mathbf{Z},\boldsymbol{\theta} \sim \mathcal{N}\left(\begin{bmatrix}\Delta_t\mathbf{Z}\mathbf{e}_{D/2+a}\\ \mathbf{0}\end{bmatrix},\right. \tag{9}$$
$$\left.\begin{bmatrix}\frac{\Delta_t^2}{4}\mathbf{K}_{\mathrm{v}a} & \frac{\Delta_t}{2}\mathbf{K}_{\mathrm{v}a}\\ \frac{\Delta_t}{2}\mathbf{K}_{\mathrm{v}a} & \mathbf{K}_{\mathrm{v}a}\end{bmatrix} + \begin{bmatrix}(\sigma^2_{\mathrm{p}a}+\sigma^2_{\kappa a})\mathbf{I} & \mathbf{0}\\ \mathbf{0} & \sigma^2_{\mathrm{v}a}\mathbf{I}\end{bmatrix}\right),$$

where $\mathbf{e}_i$ is the i^{th} column of the identity matrix $\mathbf{I} \in \mathbb{R}^{(D+E)\times(D+E)}$. Training of the hyperparameters can then be carried out in the usual manner by maximising the marginal likelihood $p([\mathbf{y}_{\mathrm{p}a},\mathbf{y}_{\mathrm{v}a}]|\mathbf{Z},\boldsymbol{\theta})$ as outlined in Sec. II-D. Note that it is impossible to distinguish between the noise term $\sigma^2_{\mathrm{p}a}$ and the approximation uncertainty $\sigma^2_{\kappa a}$ using this method. To overcome this, another GP with prior belief $\mathbf{y}_{\mathrm{p}} \sim \mathcal{N}(\mathbf{0},\mathbf{K}_{\mathrm{p}a}+\sigma^2_{\mathrm{p}a}\mathbf{I})$ can be trained simultaneously in order to learn the true noise parameter $\sigma^2_{\mathrm{p}a}$. A similar procedure is followed in the case of derivative reconstruction.

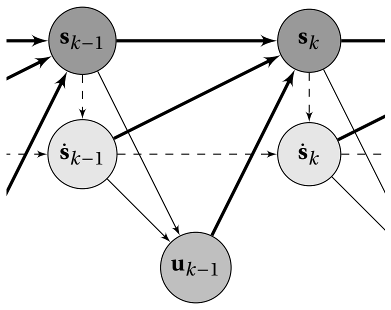
V. The PILCO Algorithm

A. Overview

The PILCO algorithm [4] is a Bayesian framework for reinforcement learning in systems with continuous state spaces. The goal is to learn a time-invariant parameterised



(a) State uncertainty propagation with time integral reconstruction.



(b) State uncertainty propagation with time derivative reconstruction.

Fig. 3. Graphical model representation of the propagation of uncertainty over the previous state $p(\mathbf{s}_{k-1},\dot{\mathbf{s}}_{k-1})$ to the distribution at the current timestep $p(\mathbf{s}_k,\dot{\mathbf{s}}_k)$. Steps involving the dynamics GP, policy and time operator reconstruction are denoted by bold, thin and dashed lines respectively.

control policy $\mathbf{u} = \boldsymbol{\pi}(\mathbf{x})$ that satisfies the finite horizon optimisation problem

$$\min V^{\boldsymbol{\pi}} = \min \frac{1}{T}\sum_{k=1}^{T} \mathbb{E}[c(\mathbf{x}_k)], \tag{10}$$

where $c : \mathbb{R}^D \to [0,1]$ is the cost function, $V^{\boldsymbol{\pi}}$ is the cost-to-go and T is the optimisation horizon. The algorithm consists of iterating over the following steps:

- **Learn Dynamics.** Interact with the system based on the current best policy $\boldsymbol{\pi}_j$. Update a Gaussian process model of the transition dynamics based on all interaction data so far.
- **Policy Evaluation.** Evaluate the (approximate) expected cost-to-go $V^{\boldsymbol{\pi}_j}$ given the current Gaussian process model of the dynamics and policy $\boldsymbol{\pi}_j$.
- **Policy Improvement.** Using the derivatives $\mathrm{d}V^{\boldsymbol{\pi}}/\mathrm{d}\boldsymbol{\psi}$, and the evaluation step along with gradient descent optimisation, find $\boldsymbol{\pi}_{j+1} = \operatorname{argmin} V^{\boldsymbol{\pi}}$ where $\boldsymbol{\psi}$ are the tuneable parameters.

B. Learn Dynamics

The first step of the process is to generate a data set $\mathscr{D}$ by applying an arbitrary policy to the system. This data set is then used to train a Gaussian process model of the transition dynamics as outlined in Section II-C. After each optimisation of the policy parameters a new rollout is performed. The data generated from this rollout is then used to augment the training data set and the hyperparameters are re-trained.

C. Policy Evaluation

In order to evaluate the cost-to-go $V^{\boldsymbol{\pi}}$ of a given set of policy parameters $\boldsymbol{\psi}$ a method of obtaining a distribution over the resulting trajectory $\boldsymbol{\tau} := \{\mathbf{x}_0 \ldots \mathbf{x}_H\}$ is needed. This requires a way of propagating the distribution at one timestep $p(\mathbf{x}_{k-1})$ to the next $p(\mathbf{x}_k)$. The first step is to obtain the joint distribution $p(\mathbf{x}_{k-1},\mathbf{u}_{k-1})$ from the policy $\boldsymbol{\pi}$. If the policy is linear then $(\mathbf{x}_{k-1},\mathbf{u}_{k-1}) \sim \mathcal{N}$ given $\mathbf{x}_{k-1} \sim \mathcal{N}$. For other policy structures the joint distribution can be approximated using exact moment matching. In Section III-B it was shown how to carry out the step $p(\mathbf{x}_{k-1},\mathbf{u}_{k-1}) \to p(\mathbf{x}_k)$ which completes the process. Graphical models of the procedure for integral reconstruction and derivative reconstruction are shown in Fig. 3.

The second step in policy evaluation is to find the expected cost at each stage $\mathbb{E}\big[c(\mathbf{x}_k)\big]$ given $\mathbf{x}_k \sim \mathcal{N}$ from the previous stage. This can be done analytically given various choices of cost function. Note that the input $\mathbf{u}$ is not explicitly penalised because input constraints can be incorporated directly through the policy structure.

D. Policy Improvement

Since analytical expressions for the cost-to-go function have been found in the previous step the derivative vector $\mathrm{d}V^{\boldsymbol{\pi}}/\mathrm{d}\boldsymbol{\psi}$ can also be obtained analytically using

$$\frac{\mathrm{d}V^{\boldsymbol{\psi}}}{\mathrm{d}\boldsymbol{\psi}} = \sum_{k=1}^{T}\left[\left(\frac{\partial}{\partial\boldsymbol{\mu}_k}\mathbb{E}\big[c(\mathbf{x}_k)\big]\right)\frac{\mathrm{d}\boldsymbol{\mu}_k}{\mathrm{d}\boldsymbol{\psi}} + \left(\frac{\partial}{\partial\boldsymbol{\Sigma}_k}\mathbb{E}\big[c(\mathbf{x}_k)\big]\right)\frac{\mathrm{d}\boldsymbol{\Sigma}_k}{\mathrm{d}\boldsymbol{\psi}}\right]. \tag{11}$$

Taking $p(\mathbf{x}_k) \sim \mathcal{N}(\boldsymbol{\mu}_k,\boldsymbol{\Sigma}_k)$ the derivatives of the mean and covariance at each timestep can be found using the recursive rule

$$\frac{\mathrm{d}\boldsymbol{\alpha}_k}{\mathrm{d}\boldsymbol{\psi}} = \frac{\partial\boldsymbol{\alpha}_k}{\partial\boldsymbol{\mu}_{k-1}}\frac{\mathrm{d}\boldsymbol{\mu}_{k-1}}{\mathrm{d}\boldsymbol{\psi}} + \frac{\partial\boldsymbol{\alpha}_k}{\partial\boldsymbol{\Sigma}_{k-1}}\frac{\mathrm{d}\boldsymbol{\Sigma}_{k-1}}{\mathrm{d}\boldsymbol{\psi}} + \frac{\partial\boldsymbol{\alpha}_k}{\partial\boldsymbol{\psi}}, \tag{12}$$

where $\boldsymbol{\alpha} \in \{\boldsymbol{\mu},\boldsymbol{\Sigma}\}$. The availability of these derivatives means that optimisation in Eq. 10 can be achieved using gradient descent methods.

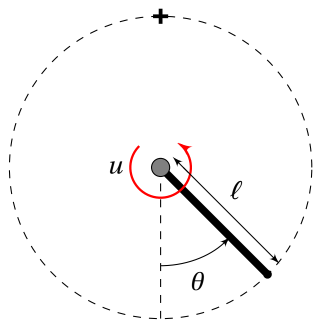
VI. Simulations

A. Pendulum

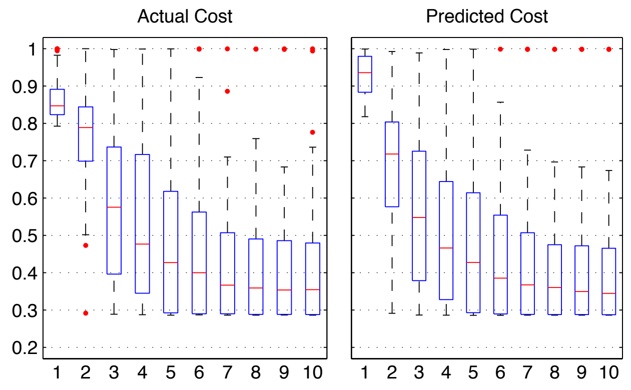
In order to test the validity of these approximation schemes, the torque-limited pendulum swing up control problem was considered as shown in Fig. 4(a). The equation of motion for this system is

$$\ddot{\theta} = \frac{u - b\dot{\theta} - \frac{1}{2}mlg\sin\theta}{\frac{1}{4}ml^2 + I}, \tag{13}$$

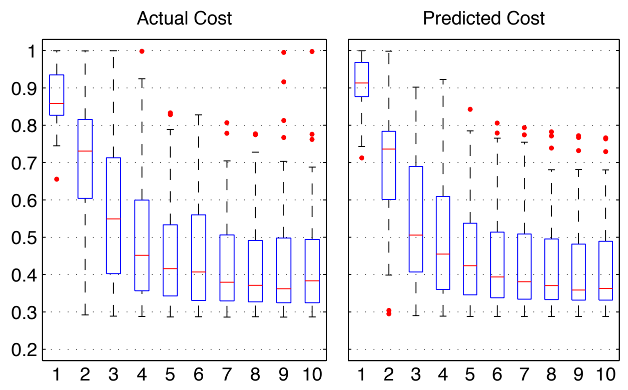
with states $\mathbf{x} = [\theta,\dot{\theta}]^\top$, constrained input torque $u \in [-3,3]$ N m (which is insufficient to swing the pendulum up directly), coefficient of friction $b = 0.1$ N m s, mass $m = 1$ kg, length $l = 1$ m, gravitational acceleration $g = 9.81$ m s^{-2} and moment of inertia I. The state measurement was



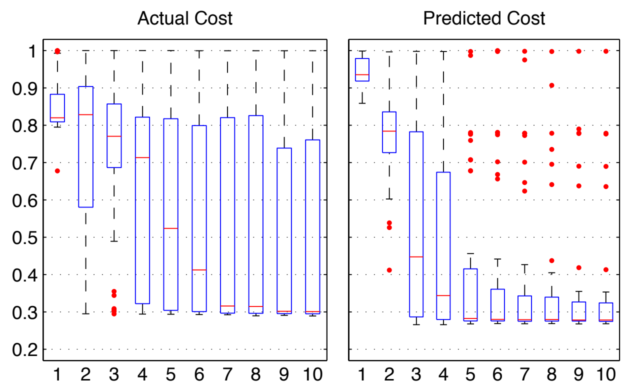
(a) Torque-limited pendulum



(b) Standard method



(c) Position reconstruction



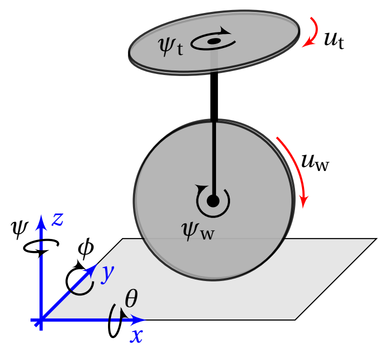
(d) Velocity reconstruction

Fig. 4. Box plots depicting the actual cost-to-go and the associated mean cost predicted by the GP of 50 Monte Carlo runs with different initial training data sets. Each box bounds the 25th to 75th percentile of the data with the 50th percentile as a line in between. The dashed lines extend to the furthest outlier that lies within reach of 1.5 times the length of the box. The x-axis denotes the algorithm iteration and the y-axis denotes the cost (actual or mean predicted) incurred during a trial run.

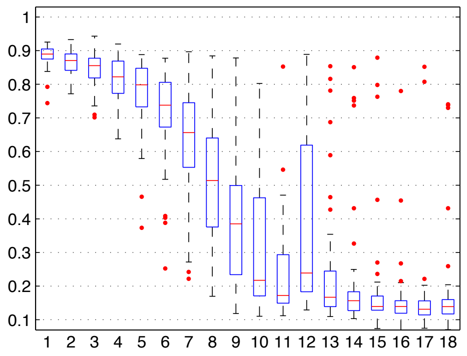
corrupted with white noise of variance $0.01^2\mathbf{I}$. The cost function used for learning was the normalised squared distance to the upright position $c(\mathbf{x}) = \frac{1}{4}\sin^2\theta + \frac{1}{4}(\cos\theta + 1)^2 \in [0,1]$. This cost makes no distinction between swinging the pendulum up to $\theta = \pi$ or $\theta = -\pi$. The discrete timestep was set to $\Delta_t = 0.1$ s (short compared to the natural period of the pendulum ≈ 1.6 s) and the prediction horizon to $H = 100$. The control policy consisted of a radial basis function with 20 Gaussian kernels in which the positions, widths and magnitudes of each kernel were free parameters to be optimised. This was then passed through a saturating function in order to respect the input constraints.

It should be noted that the incorporation of prior position-velocity information in this case will reduce predictive computation by the order of a half since $D = 2$ and $F = 1$. However, as the following results show, this may not lead to improved learning performance.

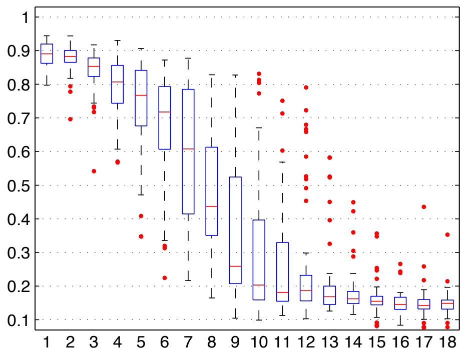
The pendulum set-up and the results of 50 Monte Carlo runs of 10 iterations of the learning algorithm are depicted in Fig. 4. The optimal solution (a single back swing then swing up) incurs a cost-to-go of roughly 0.3 but obviously there are many local minima in this problem corresponding to multiple swings. Fig. 4(b) shows the results of the standard PILCO learning algorithm where individual GPs are trained for each state. It can be observed that, on average, the mean cost predicted by the algorithm corresponds closely with the actual observed cost. Fig. 4(c) is obtained using the method of integral reconstruction and shows essentially equivalent performance to the standard method. Finally, Fig. 4(d) shows the results of derivative reconstruction, which reaches the global optimal solution for over half of the trial runs. However, using this method has led to a large discrepancy between predicted and actual performance and has caused the algorithm to get stuck in a situation where it persistently predicts that it will perform well but in reality performs very poorly. The problem in this instance can be remedied by inducing greater uncertainty in the modelling approximation $\boldsymbol{\Sigma}_{\boldsymbol{\kappa}}$. Doing this yields convergence of almost all the training cases to the global minimum solution. The same trick can be applied to the case of position reconstruction and an increase in the number of trials that reach the global minimum can be observed. This phenomenon points to the fact that an overly pessimistic or cautious (i.e. assuming greater uncertainty than is justified by the data) approach to learning is advantageous.



(a) Robotic unicycle



(b) Standard method



(c) Position reconstruction

Fig. 5. Box plots depicting the actual cost-to-go for 50 Monte Carlo runs of the algorithm on the robotic unicycle problem using the standard method, integral reconstruction and derivative reconstruction for $\Delta_t = 0.15$ s. The x-axis denotes the algorithm iteration and the y-axis denotes the cost-to-go.

B. *Unicycle*

The scheme was then tested on a highly nontrivial control problem, the balancing of a robotic unicycle as shown in Fig. 5(a). The equations of motion for this system can be found in the thesis of [6]. The state is $\mathbf{x} = [x_c, y_c, \theta, \phi, \psi, \dot{\theta}, \dot{\phi}, \dot{\psi}, \dot{\psi}_w, \dot{\psi}_t]^\top \in \mathbb{R}^{10}$ where (x_c, y_c) is the position of the desired unicycle location in a coordinate system centred on the unicycle itself, θ is the roll angle, ϕ is the pitch angle, ψ is the yaw angle, ψ_w is the angular position of the wheel and ψ_t is the angular position of the turntable. The input vector is $\mathbf{u} = [u_t, u_w]^\top \in \mathbb{R}^2$, where u_t is the torque applied to the turntable and u_w is the torque applied to the wheel. The state measurement was corrupted with white noise of variance $0.002^2\mathbf{I}$ and the discrete timestep was set to $\Delta_t = 0.15$ s which is very large given the dynamics of the unicycle and therefore makes the task of control even harder. The prediction horizon was set to $H = \lceil 10/\Delta_t \rceil$ to give a simulation time of 10 s.

Since there are three time derivative state pairs, the inclusion of prior information about the position-velocity relationships reduces predictive computation by 30%. As shown in Fig. 5(b) the standard method generally solves the balancing task in around 10–14 iterations. Fig. 5(c) shows that in making a constant acceleration approximation to reconstruct position variables there is actually an improvement in performance. The number of trials that fail to learn the task is also reduced. These results are surprising given the large time discretisation of the dynamics. In this example, the uncertainty parameter was simply set to $\mathbf{\Sigma}_{\boldsymbol{\kappa}} = 10^{-8}\mathbf{I}$ to encode confidence in the approximation. However, when using derivative reconstruction, the algorithm performed very poorly and even led to numerically unstable results in some cases. This was regardless of the value of $\mathbf{\Sigma}_{\boldsymbol{\kappa}}$. Therefore integral reconstruction is to be recommended over derivative reconstruction.

VII. Conclusion

Gaussian processes are a powerful tool for the modelling and learning of nonlinear state space systems. Until now there has not been a clear way of incorporating useful prior information in terms of known relationships between states. This paper has outlined a method of incorporating this information and using it in making predictions of state trajectories. This is advantageous for two reasons. Learning the nature of the true dynamics is faster since the Gaussian process does not have to infer known relationships. The computational cost of prediction is then reduced from $\mathcal{O}(Dn^2)$ to $\mathcal{O}((D-F)n^2)$ for a system with D states, F known state relationships and n observations. In particular, a simple approximation method to include time derivative information has been developed. When incorporated into the PILCO learning algorithm framework the time integral reconstruction approximation has been shown to yield comparable performance with the standard method on the torque-limited pendulum swing up and unicycle balancing problems even when the discrete timestep is large.

VIII. Acknowledgments

Joe would like to thank Richard Pates for enlightening discussion regarding the generalisation of the contribution of this paper.

References

[1] Williams, C. K. & Rasmussen, C. E. (1996) Gaussian processes for regression, In *Advances in Neural Information Processing Systems 8*, 514–520

[2] Rasmussen, C. E. & Williams, C. K. (2006) *Gaussian Processes for Machine Learning*, The MIT Press, Cambridge, MA.

[3] Ko, J., Klein, D., Fox, D. & Haehnel, D. (2007) Gaussian processes and reinforcement learning for identification and control of an autonomous blimp, In *Proceedings of the International Conference on Robotics and Automation (ICRA)*

[4] Deisenroth, M. P. & Rasmussen, C. E. (2011) PILCO: A model-based and data-efficient approach to policy search, In *Proceedings of the 28th International Conference on Machine Learning (ICML)*, Bellevue, WA.

[5] Quiñonerno-Candela, J., Girard, A., Larsen, J. & Rasmussen, C. E. (2003) Propagation of uncertainty in Bayesian kernel models - Application to multiple-step ahead forecasting, In *Proceedings of the IEEE Conference on Acoustics, Speech and Signal Processing*, **2**:701–704, Hong Kong, China.

[6] Forster, D. (2009) Robotic unicycle, *Technical Report*, Department of Engineering, University of Cambridge, Cambridge, UK.