# Bayesian Reinforcement Learning with Gaussian Process Temporal Difference Methods

**Yaakov Engel** YAKI@CS.UALBERTA.CA
*AICML, Department of Computing Science*
*University of Alberta*
*Edmonton, Alberta, Canada T6G 2E8*

**Shie Mannor** SHIE@ECE.MCGILL.CA
*Department of Electrical and Computer Engineering*
*McGill University*
*Montreal QC Canada, H3V 2A7*

**Ron Meir** RMEIR@EE.TECHNION.AC.IL
*Department of Electrical Engineering*
*Technion Institute of Technology*
*Haifa 32000, Israel*

**Editor:** U. N. Known

## Abstract

Reinforcement Learning is a class of problems frequently encountered by both biological and artificial agents. An important algorithmic component of many Reinforcement Learning solution methods is the estimation of state or state-action values of a fixed policy controlling a Markov decision process (MDP), a task known as policy evaluation. We present a novel Bayesian approach to policy evaluation in general state and action spaces, which employs statistical generative models for value functions via Gaussian processes (GPs). The posterior distribution based on a GP-based statistical model provides us with a value-function estimate, as well as a measure of the variance of that estimate, opening the way to a range of possibilities not available up to now. We derive exact expressions for the posterior moments of the value GP, which admit both batch and recursive computations. An efficient sequential kernel sparsification method allows us to derive efficient online algorithms for learning good approximations of the posterior moments. By allowing our algorithms to evaluate state-action values we derive model-free algorithms based on Policy Iteration for improving policies, thus tackling the complete RL problem. A companion paper describes experiments conducted with the algorithms presented here.

**Keywords:** Reinforcement Learning, Bayesian Inference, Gaussian Processes, Temporal Differences, Online Learning

## 1. Introduction

Reinforcement Learning (RL) is a class of learning problems concerned with achieving long-term goals in unfamiliar, uncertain and dynamic environments. Such tasks are conventionally formulated as Markov decision process (MDPs) or more generally as partially observable MDPs (POMDPs). Many of the algorithms developed for solving RL problems may be traced back to the Dynamic Programming (DP) Policy Iteration and Value Iteration al-

gorithms (Bellman, 1957; Bertsekas, 1995; Bertsekas and Tsitsiklis, 1996). However, there are two major features distinguishing RL from the traditional planning framework. First, while in planning it is assumed that the environment is fully known, in RL no such assumption is made. Second, in RL the learning process is usually assumed to take place *online*, namely, concurrently with the accumulation of data acquired by the learning agent as it interacts with its environment. These two features make RL a significantly more difficult challenge, and place constraints on potential RL algorithms. Due to these constraints, the RL practitioner has at her disposal a rather limited arsenal of provably convergent algorithms that can tackle real-world RL problems, namely, problems characterized by large or infinite state and/or action spaces. In such problems it is necessary to use some form of function approximation (FA) to represent the value function (and possibly the policy), since a tabular representation is infeasible. The difficulty arises because many otherwise popular function approximation schemes (e.g. multilayer Perceptrons) interfere with the contraction properties of the DP operators, when they are used to represent the value function (Bertsekas and Tsitsiklis, 1996; Gordon, 1996). In reality, practitioners either ignore this problem and make do without any convergence guarantees, with mixed success (e.g., Tesauro, 1995; Schraudolph et al., 1994; Crites and Barto, 1996), or resort to using special forms of FA that are well behaved (Singh et al., 1995; Boyan and Moore, 1995; Tsitsiklis and Van Roy, 1996; Gordon, 1996; Munos, 2000).

One particular form of FA, in which the approximation is linear in its parameters has emerged as particularly useful. There is a significant body of work providing convergence guarantees for the Temporal Difference (TD) family of algorithms (Sutton, 1988) and several related algorithms, when used in conjunction with such linear approximation architectures (Tsitsiklis and Van Roy, 1996; Konda and Tsitsiklis, 2000; Nedic and Bertsekas, 2003; Munos, 2003). However, linear parametric FA architectures are inherently limited in their expressive powers, since one must choose *a priori* a finite set of basis functions, the span of which constitutes the hypothesis space of which the value approximation must be a member. If the true value function does not belong to this hypothesis space, the solutions to which these algorithms converge may be quite poor, depending on a measure of the distance between the hypothesis space and the true value function, as well as on the values of the TD parameter $\lambda$ and the discount factor $\gamma$ (see e.g., Bertsekas and Tsitsiklis, 1996, Example 6.5). Nonetheless, for the RL practitioner seeking a provably convergent, online, model-free algorithm for value estimation, the choice is limited to TD($\lambda$) and some of its variants, such as SARSA($\lambda$) (Sutton and Barto, 1998) and LSTD($\lambda$), used in conjunction with a linear function approximation architecture (Bradtke and Barto, 1996; Boyan, 1999; Lagoudakis et al., 2002).

The main contributions made in this paper are threefold. We present a non-parametric approach to value function approximation that is not generally limited, a priori, to the span of any finite number of basis functions. This approach may be viewed as a straightforward application of the "kernel trick" (as it is known in the kernel-machines community, see Schölkopf and Smola (2002)) to the problem of value estimation. The use of a non-parametric kernel-based architecture allows us to conduct the search directly in a generally infinite dimensional Hilbert space of functions, limiting the hypothesis space only by our choice of kernel function. This naturally alleviates the difficulty discussed above with regard to linear architectures, but entails some computational issues. Our second contribution is

due to our particular choice of kernel machine, namely Gaussian Processes (GPs). The use of a GP model for the value function allows us to apply Bayesian reasoning to the value estimation problem, resulting in a complete posterior distribution over value functions, rather than the point estimates provided by other kernel-based methods. This means that, in addition to the value function estimate provided by the mean of the GP, we are also provided with a measure of the uncertainty in this estimate, given by the GP's covariance. This opens the way to a wide range of applications that make use of these uncertainty measures. Finally, by considering parametric GPs, we derive parametric counterparts of our nonparametric algorithms. Beyond their practical utility, these algorithms help elucidate the relation between our GP approach and some of the most popular RL algorithms to date, such as TD($\lambda$) and LSTD($\lambda$).

In the next section we briefly overview Gaussian processes and Markov decision processes. We then describe our Gaussian process temporal difference (GPTD) model and find the posterior moments of the value GP. We then derive exact and approximate algorithms for computing the posterior value distribution. We then devote a section to discuss parallels between variants of our algorithms and some of the best known algorithms for policy evaluation. In the next section we propose extensions of our algorithms that allow us to improve policies, without having to learn a transition model of the MDP, thus making it possible to tackle the complete RL problem, in which the goal is to find optimal or, failing that, good suboptimal policies. We close with a discussion and suggestions for future work.

## 2. Preliminaries

Gaussian Processes (GPs) have been used extensively in recent years in supervised learning tasks such as classification and regression (e.g., O'Hagan, 1978; Gibbs and MacKay, 1997; Williams, 1999; Seeger, 2004; Rasmussen and Williams, 2006). Based on probabilistic generative models, GP methods are theoretically attractive since they allow a Bayesian treatment of these problems, yielding full posterior distributions based both on one's prior beliefs and on the data observed, rather than the point-estimates usually provided by other methods. Since GPs may be defined directly in function space, they are not as restrictive as parametric models in terms of the hypothesis space in which learning takes place. Moreover, when both the prior distribution and the likelihood are Gaussian, the posterior distribution, conditioned on the observations, is also Gaussian and Bayes' rule yields closed-form expressions for the posterior moments.

### 2.1 Bayesian Inference with Gaussian Processes

A random process (or random field) $F$ is a set of random variables, each of which is assigned an index. Here we will focus on random processes indexed by the state variable $\mathbf{x} \in \mathcal{X}$ of a MDP.[1] $F$ may be thought of as a random vector if $\mathcal{X}$ is finite, as a random series if $\mathcal{X}$ is countably infinite, and as a *random function* if $\mathcal{X}$ is uncountably infinite. In the latter case, each instantiation of $F$ is a function $f : \mathcal{X} \to \mathbb{R}$. On the other hand, for a given $\mathbf{x}$, $F(\mathbf{x})$ is a random variable (RV), jointly distributed with the other components of $F$ according to $F$'s probability law.

1. Later we will extend our discussion to random processes indexed by state-action pairs $(\mathbf{x}, \mathbf{u}) \in \mathcal{X} \times \mathcal{U}$.

A random process $F$ is said to be a Gaussian process if the variables corresponding to *any* finite subset of $\mathcal{X}$ are jointly Gaussian. In order to perform Bayesian inference using GPs we need to define a *statistical generative model*. Such models typically consist of the following ingredients:

1. A *model-equation* relating the observed and unobserved components of our model, in which the latter is usually transformed and corrupted by some additive measurement noise to produce the former. The unobserved, or latent process is the subject of our Bayesian inference effort.

2. A distribution of the measurement noise terms. By noise, we refer to any additive random process in the model equation, the statistics of which is known (at least up to a few undetermined hyperparameters), and which is *not* the subject of our inference problem.

3. A prior distribution of the unobserved process. This is a necessary ingredient required for employing Bayes' rule, and is used to express whatever prior information we may have concerning $F$.

Given that $F$ is a priori Gaussian, its prior distribution is fully specified by its mean and covariance,

$$\begin{aligned} \mathbf{E}\left[F(\mathbf{x})\right] &\stackrel{\text{def}}{=} f_0(\mathbf{x}), \\ \mathbf{Cov}\left[F(\mathbf{x}), F(\mathbf{x}')\right] = \mathbf{E}\left[F(\mathbf{x})F(\mathbf{x}')\right] - f_0(\mathbf{x})f_0(\mathbf{x}') &\stackrel{\text{def}}{=} k(\mathbf{x}, \mathbf{x}'), \end{aligned} \tag{2.1}$$

respectively, where $\mathbf{E}$ denotes the expectation operator with respect to the GP distribution. In order for $k(\cdot,\cdot)$ to be a legitimate covariance it is required to be symmetric and positive-definite. Interestingly, these are exactly the requirements made of Mercer kernels, used extensively in the field of kernel machines. In kernel methods, $k(\cdot,\cdot)$ is usually referred to as the kernel function, and is viewed as an inner product in some high dimensional feature space. As it turns out, these two views are in fact equivalent, which is the reason the same notation is used for both functions (see Schölkopf and Smola, 2002; Shawe-Taylor and Cristianini, 2004, for details).

As an illustrative example let us review the use of GPs for regression with white Gaussian noise. In this setup, we are provided with a sample of $t$ training examples $\{(\mathbf{x}_i, y_i)\}_{i=1}^t$. The model-equation for some $\mathbf{x} \in \mathcal{X}$ is

$$Y(\mathbf{x}) = F(\mathbf{x}) + N(\mathbf{x}), \tag{2.2}$$

where $F$ is the GP corresponding to the unknown function from which the data are generated, $N$ is a white noise GP independent of $F$, and $Y$ is the observable process, modeled here as a noisy version of $F$. $F$ is assumed to be, a priori, a GP with mean $f_0(\cdot)$ and covariance given by a kernel function $k(\cdot,\cdot)$ as in Eq. 2.1. Eq. 2.2, evaluated at $\{x_i\}_{i=1}^t$, may be written concisely as

$$Y_t = F_t + N_t, \tag{2.3}$$

where $Y_t = (Y(\mathbf{x}_1), \ldots, Y(\mathbf{x}_t))^\top$, $F_t = (F(\mathbf{x}_1), \ldots, F(\mathbf{x}_t))^\top$ and $N_t = (N(\mathbf{x}_1), \ldots, N(\mathbf{x}_t))^\top$. In our example, we assume that the noise terms corrupting each sample are independently

and identically distributed (IID), we therefore have $N_t \sim \mathcal{N}\left(\mathbf{0}, \sigma^2 \mathbf{I}\right)$, where $\sigma^2$ is the variance of each noise term. $F(\mathbf{x})$ for any $\mathbf{x} \in \mathcal{X}$ and $Y_t$ are jointly normally distributed:

$$\begin{pmatrix} F(\mathbf{x}) \\ Y_t \end{pmatrix} \sim \mathcal{N}\left\{ \begin{pmatrix} f_0(\mathbf{x}) \\ \mathbf{f}_0 \end{pmatrix}, \begin{bmatrix} k(\mathbf{x}, \mathbf{x}) & \mathbf{k}_t(\mathbf{x}) \\ \mathbf{k}_t(\mathbf{x})^\top & \mathbf{K}_t + \sigma^2 \mathbf{I} \end{bmatrix} \right\},$$

where we denoted $(\mathbf{f}_0)_i = f_0(\mathbf{x}_i)$, $[\mathbf{K}_t]_{i,j} = k(\mathbf{x}_i, \mathbf{x}_j)$ and $(\mathbf{k}_t(\mathbf{x}))_i = k(\mathbf{x}_i, \mathbf{x})$, for $i = 1, 2, \ldots, t$. We may now invoke Bayes' rule to find the posterior moments of $F$, conditioned on the observed data (see Appendix B.1):

$$\begin{aligned} (F(\cdot)|Y_t) &\sim \mathcal{N}\left\{ \hat{F}_t(\cdot), P_t(\cdot, \cdot) \right\}, \quad \text{where} \\ \hat{F}_t(\mathbf{x}) &= f_0(\mathbf{x}) + \mathbf{k}_t(\mathbf{x})^\top \left(\mathbf{K}_t + \sigma^2 \mathbf{I}\right)^{-1} (Y_t - \mathbf{f}_0), \\ P_t(\mathbf{x}, \mathbf{x}') &= k(\mathbf{x}, \mathbf{x}') - \mathbf{k}_t(\mathbf{x})^\top \left(\mathbf{K}_t + \sigma^2 \mathbf{I}\right)^{-1} \mathbf{k}_t(\mathbf{x}'). \end{aligned} \tag{2.4}$$

### 2.2 Markov Decision Processes

As mentioned above, Markov Decision Processes (MDPs), or a generalization thereof, known as partially observable MDPs (POMDPs, Kaelbling et al. (1998)), provide the formal basis underlying RL methodology. Let us denote by $\mathcal{P}(\mathcal{S})$ the set of probability distributions over (Borel) subsets of a set $\mathcal{S}$. A discrete time MDP is a tuple $(\mathcal{X}, \mathcal{U}, p_0, p, q, \gamma)$, where $\mathcal{X}$ and $\mathcal{U}$ are the state and action spaces, respectively; $p_0(\cdot) \in \mathcal{P}(\mathcal{X})$ is a probability density from which the initial state is drawn; $p(\cdot|\mathbf{x}, \mathbf{u}) \in \mathcal{P}(\mathcal{X})$ is a probability density over successor states, conditioned on the current state $\mathbf{x}$ and action $\mathbf{u}$; $q(\cdot|\mathbf{x}) \in \mathcal{P}(\mathbb{R})$ is a probability density over immediate single-step rewards, conditioned on the current state.[2] We denote by $R(\mathbf{x})$ the random variable distributed according to $q(\cdot|\mathbf{x})$. Finally, $\gamma \in [0, 1]$ is a discount factor. We assume that both $p$ and $q$ are stationary, that is, that they do not depend explicitly on time.

In the context of control it is useful to make several additional definitions. A *stationary policy* $\mu(\cdot|\mathbf{x}) \in \mathcal{P}(\mathcal{U})$ is a time-independent mapping from states to action selection probabilities. A given policy induces a *policy-dependent* state-transition probability distribution, defined as[3]

$$p^\mu(\mathbf{x}'|\mathbf{x}) = \int_{\mathcal{U}} d\mathbf{u}\, \mu(\mathbf{u}|\mathbf{x}) p(\mathbf{x}'|\mathbf{u}, \mathbf{x}). \tag{2.5}$$

Hence, for a fixed policy $\mu$ and a fixed initial state $\mathbf{x}_0$, the probability (density) of observing a sequence of states $\boldsymbol{\xi}_t = \mathbf{x}_0, \mathbf{x}_1, \ldots, \mathbf{x}_t$ is $\mathbb{P}(\boldsymbol{\xi}_t) = p_0(\mathbf{x}_0) \prod_{i=1}^t p^\mu(\mathbf{x}_i|\mathbf{x}_{i-1})$.

Another useful quantity is the *discounted return*. The discounted return is a random process, defined as

$$D^\mu(\mathbf{x}) = \sum_{i=0}^{\infty} \gamma^i R(\mathbf{x}_i) | (\mathbf{x}_0 = \mathbf{x}), \quad \text{where } \mathbf{x}_{i+1} \sim p^\mu(\cdot|\mathbf{x}_i) \text{ for all } i \geq 0, \tag{2.6}$$

2. In the general case rewards may also depend on the action and the next state; to simplify the exposition we assume that the reward associated with a transition from state $\mathbf{x}$ to state $\mathbf{x}'$ depends only on $\mathbf{x}$. However, the subsequent analysis can be easily generalized to accommodate less restrictive reward models.
3. Here and in the sequel, whenever integration is performed over a finite or discrete space, the integral should be understood as a summation.

and $\gamma \in [0,1]$ is the discount factor[4]. The randomness in $D^\mu(\mathbf{x}_0)$, for any given state $\mathbf{x}_0$, is due both to the nondeterminacy of the sequence of states that follow $\mathbf{x}_0$, and to the randomness, or noise, in the rewards $R(\mathbf{x}_0), R(\mathbf{x}_1), \ldots$ etc., both of which jointly constitute the *intrinsic* randomness of the MDP.

Eq. 2.6 together with the stationarity of the MDP yield the recursive formula

$$D^\mu(\mathbf{x}) = R(\mathbf{x}) + \gamma D^\mu(\mathbf{x}'), \quad \text{where } \mathbf{x}' \sim p^\mu(\cdot|\mathbf{x}). \tag{2.7}$$

Let us define the expectation operator $\mathbf{E}_\xi$ as the expectation over all possible trajectories and all possible rewards collected therein. This allows us to define the *value function* $V^\mu(\mathbf{x})$ as the result of applying this expectation operator to the discounted return $D^\mu(\mathbf{x})$, i.e.,

$$V^\mu(\mathbf{x}) = \mathbf{E}_\xi D^\mu(\mathbf{x}) \tag{2.8}$$

Thus, applying $\mathbf{E}_\xi$ to both sides of Eq. 2.7, and using the conditional expectation formula (Scharf, 1991, Chapter 8), we get

$$\begin{aligned} V^\mu(\mathbf{x}) \stackrel{\text{def}}{=} \mathbf{E}_\xi D^\mu(\mathbf{x}) &= \mathbf{E}_\xi \left[ R(\mathbf{x}) + \gamma D^\mu(\mathbf{x}') \right] \\ &= \bar{R}(\mathbf{x}) + \gamma \mathbf{E}_{\mathbf{x}'|\mathbf{x}} \mathbf{E}_\mu \left[ D^\mu(\mathbf{x}') | \mathbf{x}' \right] \\ &= \bar{R}(\mathbf{x}) + \gamma \mathbf{E}_{\mathbf{x}'|\mathbf{x}} V^\mu(\mathbf{x}'), \end{aligned}$$

where

$$\begin{aligned} \mathbf{E}_{\mathbf{x}'|\mathbf{x}} V^\mu(\mathbf{x}') &= \int_{\mathcal{X}} d\mathbf{x}' p^\mu(\mathbf{x}'|\mathbf{x}) V^\mu(\mathbf{x}'), \quad \text{and} \\ \bar{R}(\mathbf{x}) &= \int_{\mathbb{R}} dr q(r|\mathbf{x}) r \quad \text{is the expected reward at the state } \mathbf{x}. \end{aligned}$$

The equality we have just established, namely that

$$V^\mu(\mathbf{x}) = \bar{R}(\mathbf{x}) + \gamma \mathbf{E}_{\mathbf{x}'|\mathbf{x}} V^\mu(\mathbf{x}') \quad \forall \mathbf{x} \in \mathcal{X}, \tag{2.9}$$

is the fixed-policy (i.e., single action) version of the Bellman equation (Bellman, 1957)[5]. A policy that maximizes the expected discounted return from each state is called an optimal policy, and is denoted by $\mu^*$. In the case of stationary MDPs, there exists a *deterministic* optimal policy[6]. With some abuse of notation, we denote the action selected by a deterministic policy $\mu$, at a state $\mathbf{x}$, by $\mu(\mathbf{x})$.

The value function corresponding to an optimal policy is called the optimal value, and is denoted by $V^* = V^{\mu^*}$. While there may exist more than one optimal policy, the optimal value function is unique (Bertsekas, 1995), and may be computed by solving Bellman's optimality equation

$$V^*(\mathbf{x}) = \bar{R}(\mathbf{x}) + \gamma \max_{\mathbf{u}} \int_{\mathcal{X}} d\mathbf{x}' p(\mathbf{x}'|\mathbf{u}, \mathbf{x}) V^*(\mathbf{x}') \quad \forall \mathbf{x} \in \mathcal{X}. \tag{2.10}$$

4. When $\gamma = 1$ the policy must be proper, see Bertsekas and Tsitsiklis (1996)
5. A similar equation, satisfied by the variance of the discounted return, may be derived in an analogous manner, this time using the conditional variance formula, see Sobel (1982) for details.
6. This is no longer the case for POMDPs and Markov Games, see Kaelbling et al. (1998); Littman (1994).

Even if $\mathcal{X}$ is finite, and assuming for now, that both $p$ and $\bar{R}$ are known, solving the set of equations (2.10) remains a non-trivial endeavor if $|\mathcal{X}|$ is large; since we are faced with a set of $|\mathcal{X}|$ nonlinear equations. Ignoring this obstacle for the moment, let us assume that we have already solved Eq. 2.10 for $V^*$. Can we now compute an optimal policy $\mu^*$ from $V^*$? The answer is affirmative and is a consequence of Bellman's *principle of optimality*: For any $\mathbf{x} \in \mathcal{X}$, an optimal deterministic decision rule is given by (Bellman, 1957)

$$\mu^*(\mathbf{x}) = \operatorname*{argmax}_{\mathbf{u}} \int_{\mathcal{X}} d\mathbf{x}' p(\mathbf{x}'|\mathbf{u}, \mathbf{x}) V^*(\mathbf{x}'). \tag{2.11}$$

## 3. A Bayesian Approach to Value Estimation

In the preceding section we showed that the value $V$ is the result of taking the expectation of the discounted return $D$ with respect to the randomness in the trajectories and in the rewards collected therein[7]. In the classic, or frequentist approach $V(\cdot)$ is no longer random, since it is the true, albeit unknown value function induced by the policy $\mu$. Adopting the Bayesian approach, we may still view the value $V(\cdot)$ as a random entity by assigning it additional randomness that is due to our *subjective uncertainty* regarding the MDP's transition model $(p, q)$. We do not know what the true distributions $p$ and $q$ are, which means that we are also uncertain about the true value function. Previous attempts to apply Bayesian reasoning to RL modeled this uncertainty by placing priors over the MDP's transition and reward model $(p, q)$ and applying Bayes' rule to update a posterior based on observed transitions. This line of work may be traced back to the pioneering works by Bellman (1956) and Howard (1960), followed by more recent contributions by Dearden et al. (1998, 1999); Strens (2000); Duff (2002); Mannor et al. (2004); Wang et al. (2005); Poupart et al. (2006). A major shortcoming of this approach is that the resulting algorithms are limited to solving MDPs with finite (and typically rather small) state and action spaces, due to the need to maintain a probability distribution over the MDP's transition model. In this work we pursue a different path – we choose to model our uncertainty about the MDP by placing a prior (and updating a posterior) directly on $V$. We achieve this by modeling $V$ as a random process, or more specifically, as a Gaussian Process. This mirrors the traditional classification of classical RL algorithms to either model-based or model-free (direct) methods, (see Sutton and Barto, 1998, Chapter 9). Figure 1 illustrates these different approaches.

To give a hint as to how we intend to proceed, let us consider a decomposition of the discounted return into its mean (the value) and a zero-mean residual $\Delta V$:

$$D(\mathbf{x}) = \mathbf{E}_\xi D(\mathbf{x}) + (D(\mathbf{x}) - \mathbf{E}_\xi D(\mathbf{x})) \stackrel{\text{def}}{=} V(\mathbf{x}) + \Delta V(\mathbf{x}) \tag{3.1}$$

This decomposition is useful, since it separates the two sources of uncertainty inherent in the discounted return process $D$: For a known MDP model, $V$ is a (deterministic) function and the randomness in $D$ is fully attributed to the intrinsic randomness in the trajectories generated by the MDP and policy pair, modeled by $\Delta V$. On the other hand, in a MDP in which both transitions and rewards are deterministic but otherwise unknown, $\Delta V$ is

7. Here and in the sequel we simplify notation by omitting the superscript $\mu$ from $D^\mu$ and $V^\mu$.

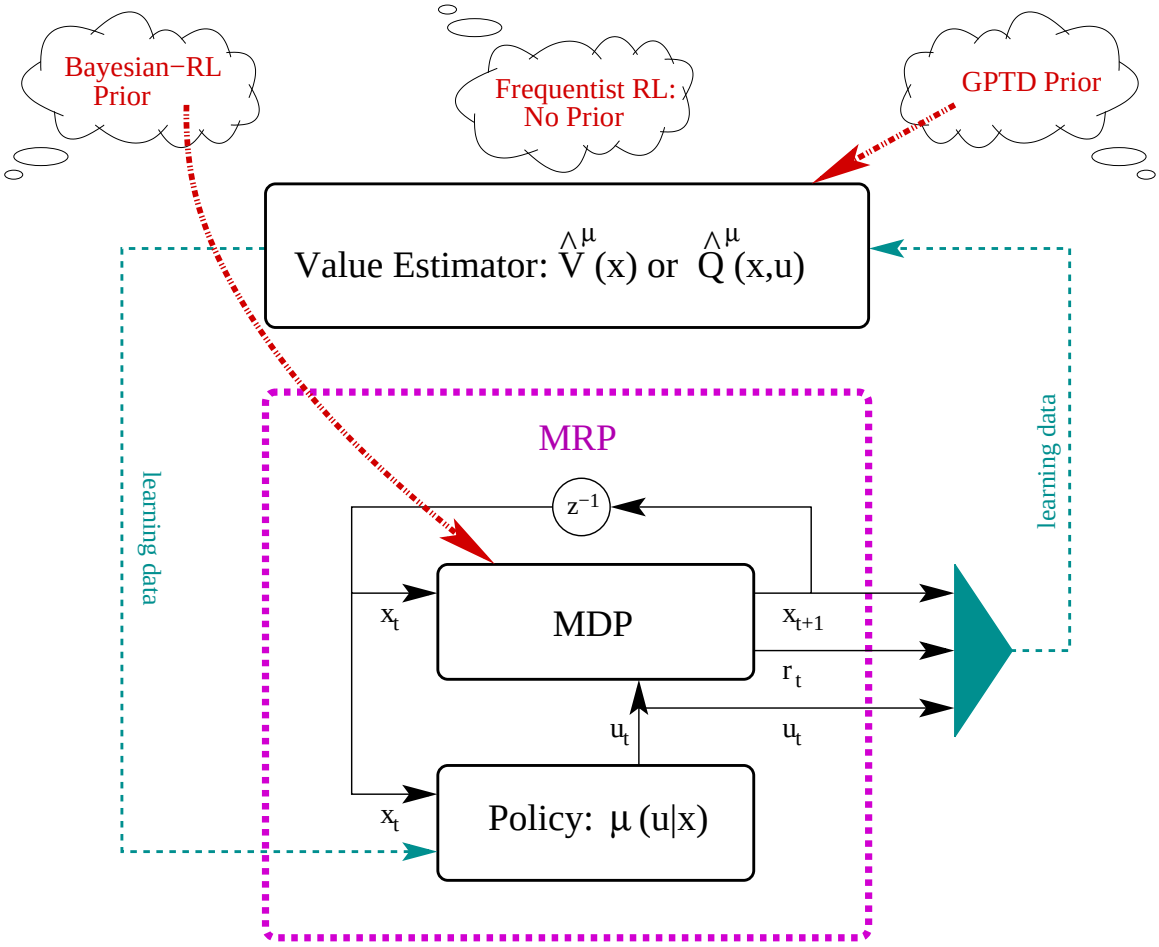


Figure 1: An illustration of the frequentist as well as the two different Bayesian approaches to value-function based reinforcement learning. In the traditional Bayesian RL approach a prior is placed on the MDP's model, whereas in our GPTD approach the prior is placed directly on the value function. $\mathbf{x}$, $\mathbf{u}$ and $r$ denote state, action and reward, respectively. The data required to learn value estimators typically consists of a temporal stream of state-action-reward triplets. Another stream of data is used to update the policy based on the current estimate of the value function. A MDP and a stationary policy controlling it, jointly constitute a Markov reward process (MRP). $z^{-1}$ denotes the 1-step time-lag operator.

deterministic (identically zero), and the randomness in $D$ is due solely to the extrinsic Bayesian uncertainty, modeled by the random process $V$.

In this section we use this insight to derive a Gaussian process model relating values and rewards.

### 3.1 A Gaussian Process Model for Value Functions

In Section 2.2 we noted that the value $V$ is the result of taking the expectation of the discounted return $D$ with respect to the randomness in the trajectories and in the rewards collected therein (Eq. 2.8). Substituting Eq. 3.1 into Eq. 2.7 and rearranging we get

$$\begin{aligned} R(\mathbf{x}) &= V(\mathbf{x}) - \gamma V(\mathbf{x}') + N(\mathbf{x}, \mathbf{x}'), \\ \text{where } \mathbf{x}' &\sim p^{\mu}(\cdot|\mathbf{x}), \quad \text{and } N(\mathbf{x}, \mathbf{x}') \stackrel{\text{def}}{=} \Delta V(\mathbf{x}) - \gamma \Delta V(\mathbf{x}'). \end{aligned} \tag{3.2}$$

Assume we are provided with a sample trajectory $\mathbf{x}_0, \mathbf{x}_1, \ldots, \mathbf{x}_t$, then we may write the model equations (3.2) for these samples, resulting in the following set of $t$ equations

$$R(\mathbf{x}_i) = V(\mathbf{x}_i) - \gamma V(\mathbf{x}_{i+1}) + N(\mathbf{x}_i, \mathbf{x}_{i+1}) \quad \text{for } i = 0, \ldots, t-1. \tag{3.3}$$

Let us denote

$$R_t = (R(\mathbf{x}_0), \ldots, R(\mathbf{x}_t))^{\top}, \; V_t = (V(\mathbf{x}_0), \ldots, V(\mathbf{x}_t))^{\top}, \; N_t = (N(\mathbf{x}_0, \mathbf{x}_1), \ldots, N(\mathbf{x}_{t-1}, \mathbf{x}_t))^{\top}. \tag{3.4}$$

Using these definitions we may write our set of model equations (3.3) concisely as

$$R_{t-1} = \mathbf{H}_t V_t + N_t, \tag{3.5}$$

where

$$\mathbf{H}_t = \begin{bmatrix} 1 & -\gamma & 0 & \ldots & 0 \\ 0 & 1 & -\gamma & \ldots & 0 \\ \vdots & & & & \vdots \\ 0 & 0 & \ldots & 1 & -\gamma \end{bmatrix}. \tag{3.6}$$

In order to fully define a complete probabilistic generative model, we also need to specify the distribution of the noise process $N_t$. We do this by modeling the residuals $\Delta V_t = (\Delta V(\mathbf{x}_0), \ldots, \Delta V(\mathbf{x}_t))^{\top}$ as random Gaussian noise[8]. In particular, this means that the distribution of the vector $\Delta V_t$ is completely specified by its mean and covariance. Another assumption we make is that each of the residuals $\Delta V(\mathbf{x}_i)$ is independently distributed. We will discuss the implications of this assumption in Section 3.3. Denoting $\sigma_i^2 = \mathbf{Var}\big[D(\mathbf{x}_i)\big]$, the distribution of $\Delta V_t$ is given by:

$$\Delta V_t \sim \mathcal{N}\big(\mathbf{0}, \mathrm{diag}(\boldsymbol{\sigma}_t)\big),$$

where $\boldsymbol{\sigma}_t = \big(\sigma_0^2, \;\; \sigma_1^2, \;\; \ldots \;\; , \sigma_t^2\big)^{\top}$, and $\mathrm{diag}(\cdot)$ denotes a diagonal matrix whose diagonal elements are the components of the argument vector. Since $N_t = \mathbf{H}_t \Delta V_t$, we have $N_t \sim$

8. This may not be a correct assumption in general; however, in the absence of any prior information concerning the distribution of the residuals, it is the *simplest* assumption we can make, since the Gaussian distribution possesses the highest entropy among all distributions with the same covariance. It is also possible to relax the Gaussianity requirement on both the prior and the noise. The resulting estimator may then be shown to be the *linear minimum mean-squared error* estimator for the value.

$\mathcal{N}(\mathbf{0}, \mathbf{\Sigma}_t)$ with,

$$\mathbf{\Sigma}_t = \mathbf{H}_t \operatorname{diag}(\boldsymbol{\sigma}_t)\mathbf{H}_t^\top \tag{3.7}$$
$$= \begin{bmatrix} \sigma_0^2+\gamma^2\sigma_1^2 & -\gamma\sigma_1^2 & 0 & \dots & 0 & 0 \\ -\gamma\sigma_1^2 & \sigma_1^2+\gamma^2\sigma_2^2 & -\gamma\sigma_2^2 & 0 & \dots & 0 \\ 0 & -\gamma\sigma_2^2 & \sigma_2^2+\gamma^2\sigma_3^2 & \ddots & & \vdots \\ \vdots & 0 & \ddots & \ddots & \ddots & 0 \\ 0 & \vdots & & \ddots & \ddots & -\gamma\sigma_{t-1}^2 \\ 0 & 0 & \dots & 0 & -\gamma\sigma_{t-1}^2 & \sigma_{t-1}^2+\gamma^2\sigma_t^2 \end{bmatrix}.$$

Fig. 2 illustrates the conditional independence relations between the latent value variables $V(\mathbf{x}_i)$, the noise variables $\Delta V(\mathbf{x}_i)$, and the observable rewards $R(\mathbf{x}_i)$. Unlike GP regression, there are vertices connecting variables from different time steps, making the ordering of samples important. Also note that, for the last state in each episode ($\mathbf{x}_t$, in the figure), $R(\mathbf{x}_t)$ depends only on $V(\mathbf{x}_t)$ and $\Delta V(\mathbf{x}_t)$ (as in Eq. 3.11).

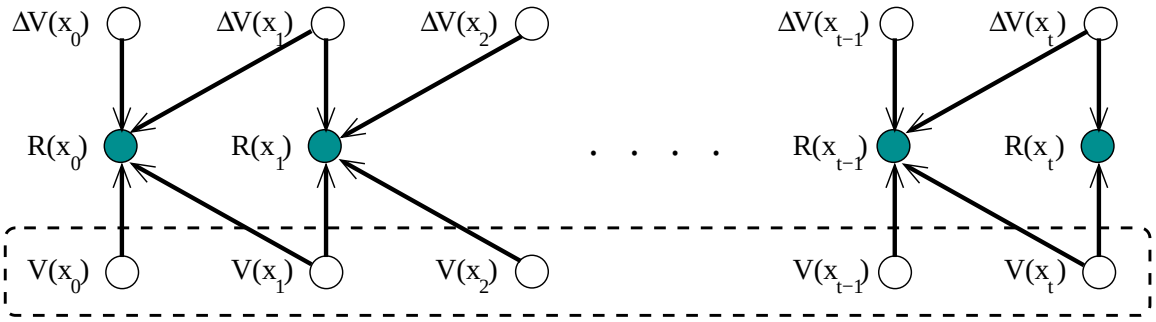


Figure 2: A graph illustrating the conditional independencies between the latent $V(\mathbf{x}_i)$ value variables (bottom row), the noise variables $\Delta V(\mathbf{x}_i)$ (top row), and the observable $R(\mathbf{x}_i)$ reward variables (middle row), in the GPTD model. As in the case of GP regression, all of the $V(\mathbf{x}_i)$ variables should be connected by arrows, due to the dependencies introduced by the prior. To avoid cluttering the diagram, this was marked by the dashed frame surrounding them.

Eq. 3.5, along with a Gaussian prior distribution on $V$, and a Gaussian measurement noise distribution, define a linear statistical model (Scharf, 1991) connecting the value and reward random processes. Bayes' rule may then be used to compute the posterior distribution of $V$ conditioned on the observed sequence of rewards, as was done in Section 2.1 for GP regression.

Let us now compute the posterior moments, conditioned on the state-reward trajectory up to time $t$. Application of Bayes' rule yields the marginal posterior distribution[9] of

9. We refer to the distribution of $V(\mathbf{x})$ as a marginal, since, to obtain this distribution, we effectively integrate out all other variables in $V$, namely $V(\mathbf{x}')$ for all $\mathbf{x}' \in \mathcal{X}\backslash\{\mathbf{x}\}$.

the value at a query point $\mathbf{x}$, conditioned on the observed sequence of rewards $\mathbf{r}_{t-1} = (r_0, \ldots, r_{t-1})^\top$ (see Section B.2 in the appendix):

$$(V(\mathbf{x})|R_{t-1} = \mathbf{r}_{t-1}) \sim \mathcal{N}\left\{\hat{V}_t(\mathbf{x}), P_t(\mathbf{x}, \mathbf{x})\right\} \quad , \tag{3.8}$$

where

$$\begin{aligned} \hat{V}_t(\mathbf{x}) &= \mathbf{k}_t(\mathbf{x})^\top \mathbf{H}_t^\top \mathbf{Q}_t \mathbf{r}_{t-1}, \quad P_t(\mathbf{x}, \mathbf{x}') = k(\mathbf{x}, \mathbf{x}') - \mathbf{k}_t(\mathbf{x})^\top \mathbf{H}_t^\top \mathbf{Q}_t \mathbf{H}_t \mathbf{k}_t(\mathbf{x}'), \\ \mathbf{Q}_t &= (\mathbf{H}_t \mathbf{K}_t \mathbf{H}_t^\top + \mathbf{\Sigma}_t)^{-1}, \quad \mathbf{\Sigma}_t = \mathbf{Cov}[N_t]. \end{aligned} \tag{3.9}$$

The expressions above can be cast in a somewhat more concise form, by separating the input dependent terms from the learned terms:

$$\hat{V}_t(\mathbf{x}) = \boldsymbol{\alpha}_t^\top \mathbf{k}_t(\mathbf{x}), \quad P_t(\mathbf{x}, \mathbf{x}') = k(\mathbf{x}, \mathbf{x}') - \mathbf{k}_t(\mathbf{x})^\top \mathbf{C}_t \mathbf{k}_t(\mathbf{x}'), \tag{3.10}$$

where $\boldsymbol{\alpha}_t = \mathbf{H}_t^\top \mathbf{Q}_t \mathbf{r}_{t-1}$ and $\mathbf{C}_t = \mathbf{H}_t^\top \mathbf{Q}_t \mathbf{H}_t$ are independent of $\mathbf{x}$ and $\mathbf{x}'$.

Due to the special structure of the noise covariance matrix $\mathbf{\Sigma}_t$, it is possible to derive recursive updates for $\boldsymbol{\alpha}_t$ and $\mathbf{C}_t$. The complete derivation may be found in Appendix A.2. The updates are:

$$\boldsymbol{\alpha}_t = \begin{pmatrix} \boldsymbol{\alpha}_{t-1} \\ 0 \end{pmatrix} + \frac{\mathbf{c}_t}{s_t} d_t, \quad \mathbf{C}_t = \begin{bmatrix} \mathbf{C}_{t-1} & \mathbf{0} \\ \mathbf{0}^\top & 0 \end{bmatrix} + \frac{1}{s_t} \mathbf{c}_t \mathbf{c}_t^\top$$

with

$$\begin{aligned} \mathbf{c}_t &= \frac{\gamma \sigma_{t-1}^2}{s_{t-1}} \begin{pmatrix} \mathbf{c}_{t-1} \\ 0 \end{pmatrix} + \mathbf{h}_t - \begin{pmatrix} \mathbf{C}_{t-1} \mathbf{\Delta k}_t \\ 0 \end{pmatrix}, \quad d_t = \frac{\gamma \sigma_{t-1}^2}{s_{t-1}} d_{t-1} + r_{t-1} - \mathbf{\Delta k}_t^\top \boldsymbol{\alpha}_{t-1}, \\ s_t &= \sigma_{t-1}^2 + \gamma^2 \sigma_t^2 - \frac{\gamma^2 \sigma_{t-1}^4}{s_{t-1}} + \Delta k_{tt} - \mathbf{\Delta k}_t^\top \mathbf{C}_{t-1} \mathbf{\Delta k}_t + \frac{2\gamma \sigma_{t-1}^2}{s_{t-1}} \mathbf{c}_{t-1}^\top \mathbf{\Delta k}_t. \end{aligned}$$

Above we made use of the following definitions:

$$\begin{aligned} \mathbf{h}_t &= (0, \ldots, 1, -\gamma)^\top, \quad \mathbf{\Delta k}_t = \mathbf{k}_{t-1}(\mathbf{x}_{t-1}) - \gamma \mathbf{k}_{t-1}(\mathbf{x}_t), \\ \Delta k_{tt} &= k(\mathbf{x}_{t-1}, \mathbf{x}_{t-1}) - 2\gamma k(\mathbf{x}_{t-1}, \mathbf{x}_t) + \gamma^2 k(\mathbf{x}_t, \mathbf{x}_t). \end{aligned}$$

It may be readily verified that these recursions should be initialized as follows:

$$\boldsymbol{\alpha}_0 = 0, \quad \mathbf{C}_0 = 0, \quad \mathbf{c}_0 = 0, \quad d_0 = 0, \quad 1/s_0 = 0.$$

The pseudocode for this algorithm is given in Algorithm 1. Note that the term driving the update of $d_t$, namely $r_{t-1} - \mathbf{\Delta k}_t^\top \boldsymbol{\alpha}_{t-1}$, is the temporal difference at time $t$, since $r_{t-1} - \mathbf{\Delta k}_t^\top \boldsymbol{\alpha}_{t-1} = r_{t-1} + \gamma \hat{V}_{t-1}(\mathbf{x}_t) - \hat{V}_{t-1}(\mathbf{x}_{t-1})$.

### 3.2 Episodic Tasks

Let us consider how the GPTD model described above needs to be modified to handle episodic learning tasks. Whereas in continual tasks, in which the RL agent is placed in some starting state, and is then allowed to wander-off indefinitely, in episodic tasks the

**Algorithm 1** A recursive Monte-Carlo GPTD algorithm

**Initialize** $\boldsymbol{\alpha}_0 = \mathbf{0}$, $\mathbf{C}_0 = 0$, $\mathcal{D}_0 = \{\mathbf{x}_0\}$, $\mathbf{c}_0 = \mathbf{0}$, $d_0 = 0$, $1/s_0 = 0$

**for** $t = 1, 2, \ldots$

**observe** $\mathbf{x}_{t-1}$, $r_{t-1}$, $\mathbf{x}_t$

$\mathbf{h}_t = (0, \ldots, 1, -\gamma)^\top$

$\boldsymbol{\Delta}\mathbf{k}_t = \mathbf{k}_{t-1}(\mathbf{x}_{t-1}) - \gamma \mathbf{k}_{t-1}(\mathbf{x}_t)$

$\Delta k_{tt} = k(\mathbf{x}_{t-1}, \mathbf{x}_{t-1}) - 2\gamma k(\mathbf{x}_{t-1}, \mathbf{x}_t) + \gamma^2 k(\mathbf{x}_t, \mathbf{x}_t)$

$\mathbf{c}_t = \frac{\gamma \sigma_{t-1}^2}{s_{t-1}} \begin{pmatrix} \mathbf{c}_{t-1} \\ 0 \end{pmatrix} + \mathbf{h}_t - \begin{pmatrix} \mathbf{C}_{t-1} \boldsymbol{\Delta}\mathbf{k}_t \\ 0 \end{pmatrix}$

$d_t = \frac{\gamma \sigma_{t-1}^2}{s_{t-1}} d_{t-1} + r_{t-1} - \boldsymbol{\Delta}\mathbf{k}_t^\top \boldsymbol{\alpha}_{t-1}$

$s_t = \sigma_{t-1}^2 + \gamma^2 \sigma_t^2 - \frac{\gamma^2 \sigma_{t-1}^4}{s_{t-1}} + \Delta k_{tt} - \boldsymbol{\Delta}\mathbf{k}_t^\top \mathbf{C}_{t-1} \boldsymbol{\Delta}\mathbf{k}_t + \frac{2\gamma \sigma_{t-1}^2}{s_{t-1}} \mathbf{c}_{t-1}^\top \boldsymbol{\Delta}\mathbf{k}_t$

$\boldsymbol{\alpha}_t = \begin{pmatrix} \boldsymbol{\alpha}_{t-1} \\ 0 \end{pmatrix} + \frac{\mathbf{c}_t}{s_t} d_t$

$\mathbf{C}_t = \begin{bmatrix} \mathbf{C}_{t-1} & \mathbf{0} \\ \mathbf{0}^\top & 0 \end{bmatrix} + \frac{1}{s_t} \mathbf{c}_t \mathbf{c}_t^\top$

$\mathcal{D}_t = \mathcal{D}_{t-1} \cup \{\mathbf{x}_t\}$

**end for**

**return** $\boldsymbol{\alpha}_t$, $\mathbf{C}_t$, $\mathcal{D}_t$

state space is assumed to contain an absorbing, *terminal state* into which the agent is assured to transition after a finite, but possibly random, number of time-steps. A terminal state may be thought of as a state from which there are only zero-reward self-transitions, for all actions. In episodic RL tasks, when such a state is reached, the episode terminates and the agent is placed in a new, usually random state (drawn from $p_0$) to begin a new episode.

As far as value estimation is concerned, the key property of terminal states is that, once a terminal state is reached, all subsequent rewards vanish. Therefore, both the discounted return and the value of a terminal state vanish, implying that the discounted return and the value of the state immediately preceding the terminal state are equal to that state's reward and expected reward, respectively. Specifically, if the last time-step in an episode is $t$ (i.e., $\mathbf{x}_{t+1}$ is a terminal state), the last equation in the system of equations (3.5) reads

$$R(\mathbf{x}_t) = V(\mathbf{x}_t) + N(\mathbf{x}_t). \tag{3.11}$$

Hence, for the first episode, $\mathbf{H}_{t+1}$ becomes the following $(t+1) \times (t+1)$ square *invertible* matrix (its determinant equals 1),

$$\mathbf{H}_{t+1} = \begin{bmatrix} 1 & -\gamma & 0 & \ldots & 0 \\ 0 & 1 & -\gamma & \ldots & 0 \\ \vdots & & & & \vdots \\ 0 & 0 & \ldots & 1 & -\gamma \\ 0 & 0 & \ldots & 0 & 1 \end{bmatrix}. \tag{3.12}$$

The noise covariance matrix for the first episode becomes (compare this to Eq. 3.7)

$$\begin{aligned}\boldsymbol{\Sigma}_{t+1} &= \mathbf{H}_{t+1}\,\mathrm{diag}(\boldsymbol{\sigma}_t)\mathbf{H}_{t+1}^\top \\ &= \begin{bmatrix} \sigma_0^2+\gamma^2\sigma_1^2 & -\gamma\sigma_1^2 & 0 & \dots & 0 & 0 \\ -\gamma\sigma_1^2 & \sigma_1^2+\gamma^2\sigma_2^2 & -\gamma\sigma_2^2 & 0 & \dots & 0 \\ 0 & -\gamma\sigma_2^2 & \sigma_2^2+\gamma^2\sigma_3^2 & \ddots & & \vdots \\ \vdots & 0 & \ddots & \ddots & \ddots & 0 \\ 0 & \vdots & & \ddots & \ddots & -\gamma\sigma_t^2 \\ 0 & 0 & \dots & 0 & -\gamma\sigma_t^2 & \sigma_t^2 \end{bmatrix}.\end{aligned} \tag{3.13}$$

Finally, our set of model equations is now (compare to Eq. 3.5)

$$R_t = \mathbf{H}_{t+1}V_t + N_{t+1}. \tag{3.14}$$

After a *sequence* of such learning episodes, each ending in a terminal state, $\mathbf{H}_{t+1}$ is a square block-diagonal matrix with each block being of the form of the r.h.s. of (3.12). The noise covariance matrix maintains a corresponding block-diagonal structure, in which each block of $\boldsymbol{\Sigma}_t$ is a tridiagonal matrix of the form (3.13).

In order to derive the updates corresponding to the last transition in an episode, it is useful to make the observation that Eq. 3.11, 3.12 and 3.14 could be obtained simply by temporarily setting the discount factor $\gamma$ to zero, only for the transition from $\mathbf{x}_t$ to the terminal state. However, since the update equations contain discount factors from two consecutive time steps, some care must be exercised. By tagging each $\gamma$ with its respective time-step, we obtain the following set of equations for the update corresponding to the last transition in an episode (we denote $\mathbf{e} = (0,\dots,0,1)^\top$):

$$\boldsymbol{\alpha}_{t+1} = \boldsymbol{\alpha}_t + \frac{\mathbf{c}_{t+1}}{s_{t+1}}d_{t+1}\,, \quad \mathbf{C}_{t+1} = \mathbf{C}_t + \frac{1}{s_{t+1}}\mathbf{c}_{t+1}\mathbf{c}_{t+1}^\top, \tag{3.15}$$

where

$$\begin{aligned}\mathbf{c}_{t+1} &= \frac{\gamma\sigma_t^2}{s_t}\mathbf{c}_t + \mathbf{e} - \mathbf{C}_t\mathbf{k}_t(\mathbf{x}_t), \quad d_{t+1} = \frac{\gamma\sigma_t^2}{s_t}d_t + r_t - \mathbf{k}_t(\mathbf{x}_t)^\top\boldsymbol{\alpha}_t, \\ s_{t+1} &= \sigma_t^2 + \mathbf{k}_t(\mathbf{x}_t)^\top\left(\mathbf{c}_{t+1} + \frac{\gamma\sigma_t^2}{s_t}\mathbf{c}_t\right) - \frac{\gamma^2\sigma_t^4}{s_t}.\end{aligned} \tag{3.16}$$

Note that we avoid performing the matrix-vector multiplication $\mathbf{C}_t\mathbf{k}_t(\mathbf{x}_t)$ in computing $s_{t+1}$, by using $\mathbf{c}_{t+1}$ computed in the $\mathbf{c}$-update. Further simplification results from the fact that $k_{tt} - \mathbf{k}_t(\mathbf{x}_t)^\top\mathbf{e} = 0$. This simple trick is applicable in all updates, not just the final one.

### 3.3 Relation to Monte-Carlo Estimation

The assumption on the independence of the residuals made in Section 3.1 can be related to a well known *Monte-Carlo* method for value estimation, see Bertsekas and Tsitsiklis (1996, Chapters 5, 6) and Sutton and Barto (1998). Monte-Carlo policy evaluation reduces the problem of estimating the value function into a supervised regression problem, in which the target values in the regression are samples of the discounted return. In the episodic setting

discussed above, we may obtain such samples by waiting until the end of the episode, while noting for each state encountered in the sample path, the sum of the discounted rewards from that point until the episode's last time-step. In the non-episodic setting, if discounted rewards are summed over a period of time $\tau \gg \frac{1}{1-\gamma}$ (the horizon time) then the error introduced by truncating the sum after $\tau$ time-steps will be small. However, in the interest of clarity, let us limit our discussion in this section to the episodic setting. Suppose that the last non-terminal state in the current episode is $\mathbf{x}_t$, then the Monte-Carlo training set is $(\mathbf{x}_i, y_i)_{i=0}^t$ with

$$y_i = \sum_{j=i}^{t} \gamma^{j-i} r_j, \tag{3.17}$$

where $r_j$ is the reward observed at the $j$'th time step.

It is interesting to note that, in the episodic setting, we can establish a connection between our GPTD models and GP regression models. This is done by performing a *whitening* transformation on Eq. 3.14. Since the noise covariance matrix $\mathbf{\Sigma}_{t+1}$ is positive definite, there exists a square matrix $\mathbf{\Sigma}_{t+1}^{-1/2}$ satisfying $\mathbf{\Sigma}_{t+1}^{-1/2^\top}\mathbf{\Sigma}_{t+1}^{-1/2} = \mathbf{\Sigma}_{t+1}^{-1}$. Multiplying Eq. (3.14) by $\mathbf{\Sigma}_{t+1}^{-1/2}$ we then get $\mathbf{\Sigma}_{t+1}^{-1/2} R_t = \mathbf{\Sigma}_{t+1}^{-1/2}\mathbf{H}_{t+1} V_t + \mathbf{\Sigma}_{t+1}^{-1/2} N_{t+1}$. The transformed noise term $\mathbf{\Sigma}_{t+1}^{-1/2} N_{t+1}$ has a covariance matrix given by $\mathbf{\Sigma}_{t+1}^{-1/2}\mathbf{\Sigma}_{t+1}\mathbf{\Sigma}_{t+1}^{-1/2^\top} = \mathbf{\Sigma}_{t+1}^{-1/2}(\mathbf{\Sigma}_{t+1}^{-1/2^\top}\mathbf{\Sigma}_{t+1}^{-1/2})^{-1}\mathbf{\Sigma}_{t+1}^{-1/2^\top} = \mathbf{I}$. Thus, the transformation $\mathbf{\Sigma}_{t+1}^{-1/2}$ whitens the noise. In our case, a whitening matrix is given by

$$\mathbf{H}_{t+1}^{-1} = \begin{bmatrix} 1 & \gamma & \gamma^2 & \dots & \gamma^t \\ 0 & 1 & \gamma & \dots & \gamma^{t-1} \\ \vdots & & & & \vdots \\ 0 & 0 & 0 & \dots & 1 \end{bmatrix} \tag{3.18}$$

(showing that $\mathbf{H}_{t+1}^{-1}$ is indeed given by the matrix above is an easy exercise). The transformed model is $\mathbf{H}_{t+1}^{-1} R_t = V_t + N'_t$ with *white* Gaussian noise, since $N'_t = \mathbf{H}_{t+1}^{-1} N_{t+1} \sim \mathcal{N}\{\mathbf{0}, \mathrm{diag}(\boldsymbol{\sigma}_t)\}$. Let us look at the $i$'th equation (i.e., row) of this transformed model:

$$R(\mathbf{x}_i) + \gamma R(\mathbf{x}_{i+1}) + \ldots + \gamma^{t-i} R(\mathbf{x}_t) = V(\mathbf{x}_i) + N'(\mathbf{x}_i), \tag{3.19}$$

where $N'(\mathbf{x}_i) \sim \mathcal{N}\{0, \sigma_i^2\}$. The l.h.s. of Eq. 3.19 is the discounted return of the path $(\mathbf{x}_i, \ldots, \mathbf{x}_t)$. Let us denote by $Y_t$ the random vector, whose $i$-th component is $(Y_t)_i = R(\mathbf{x}_i) + \gamma R(\mathbf{x}_{i+1}) + \ldots + \gamma^{t-i} R(\mathbf{x}_t)$, and by $(\mathbf{y}_t)_i = \sum_{j=i}^t \gamma^{j-i} r_j$ a sampled instance of $(Y_t)_i$. Eq. 3.19 becomes $Y_t = V_t + N'_t$, which is readily recognized as a (white noise) GP regression model (see Eq. 2.2), in which the targets are Monte-Carlo samples of the discounted return. Finally, the parameters $\boldsymbol{\alpha}_{t+1}$ and $\mathbf{C}_{t+1}$ defining the posterior moments are given by (see Eq. 2.4)

$$\boldsymbol{\alpha}_{t+1} = \left(\mathbf{K}_t + \sigma^2 \mathbf{I}\right)^{-1} \mathbf{y}_t, \quad \text{and } \mathbf{C}_{t+1} = \left(\mathbf{K}_t + \sigma^2 \mathbf{I}\right)^{-1}.$$

This equivalence between the MC-GPTD model and a GP regression model uncovers the implicit assumption underlying MC value estimation; namely, that the samples of the discounted return used for regression are statistically independent. In a typical online RL scenario, this assumption is clearly incorrect, as the samples of the discounted return are

based on trajectories that partially overlap (for instance, for two consecutive states, $\mathbf{x}_i$ and $\mathbf{x}_{i+1}$, the respective trajectories only differ by a single state – $\mathbf{x}_i$). This may help explain the frequently observed advantage of TD methods using $\lambda < 1$ over the corresponding Monte-Carlo (i.e., $\lambda = 1$) methods. The major benefit in using the GPTD formulation is that, in episodic RL tasks, it allows us to derive exact updates of the parameters of the posterior value mean and covariance that may be computed online, rather than having to wait until the end of the episode. This also justifies referring to the GP model described in this section, as well as to Algorithm 1, as the *Monte-Carlo GPTD (MC-GPTD)* algorithm. Note that in non-episodic tasks we do not have such an equivalence result, except in the limit of infinitely long trajectories.

### 3.4 Sparse Online Algorithms

Algorithm 1 described above, though recursive, is not amenable to the online setting[10]. This is due to the fact that the cost of computing the update at time-step $t$ is $O(t^2)$ in both time and memory. In the online, or real-time setting we require the computational costs at time-step $t$ to be independent of $t$, at least asymptotically. There are two general approaches for adapting the algorithms described above to this setting. One amounts to deriving and using parametric counterparts of these GPTD models and algorithms. We defer discussion of this approach to Section 3.5. In the current section we describe another approach, which is based on an efficient sequential kernel sparsification method, resulting in semi-parametric expressions for the posterior moments.

In Engel et al. (2002) an online sparsification method for kernel algorithms was proposed in the context of Support Vector Regression. This method is based on the following observation: Due to Mercer's Theorem the covariance kernel function $k(\cdot,\cdot)$ may be viewed as an inner product in a generally infinite-dimensional Hilbert space $\mathcal{H}$ (see Schölkopf and Smola (2002); Shawe-Taylor and Cristianini (2004) for details). This means that there exists a generally non-linear mapping $\boldsymbol{\phi} : \mathcal{X} \rightarrow \mathcal{H}$ for which $\left\langle \boldsymbol{\phi}(\mathbf{x}), \boldsymbol{\phi}(\mathbf{x}') \right\rangle_{\mathcal{H}} = k(\mathbf{x}, \mathbf{x}')$. Although the dimension of $\mathcal{H}$ may be exceedingly high, the dimensionality of the manifold spanned by the set of vectors $\{\boldsymbol{\phi}(\mathbf{x}_i)\}_{i=0}^{t}$ is at most $t$, and may be lower if linear dependencies between the different $\boldsymbol{\phi}(\mathbf{x}_i)$ vectors occur. Consequently, any expression describable as a linear combination of these vectors may be expressed in terms of an arbitrary set of linearly independent feature vectors which span this manifold (i.e., a basis). When such a basis consists of a subset of $\{\boldsymbol{\phi}(\mathbf{x}_i)\}_{i=0}^{t}$ we refer to it, as well as to the corresponding set of input points as a *dictionary*; which of the two we refer to should be clear from the context. Furthermore, rather than insisting on dictionaries that exactly span the manifold, we will content ourselves with dictionaries that approximately span it. Specifically, we are interested in dictionaries for which the distance (as measured by the norm in $\mathcal{H}$) between any of the feature vectors in $\{\boldsymbol{\phi}(\mathbf{x}_i)\}_{i=0}^{t}$ and the manifold spanned by the dictionary is bounded by a small positive constant.[11]

10. In learning theory the term "online" has a different meaning from the one we use here. By "online" we qualify merely the computational aspect of the algorithm – see below.
11. Based on similar ideas, several other sparsification algorithms have been previously proposed, e.g. Burges (1996); Smola and Bartlett (2001); Williams and Seeger (2001); Csató and Opper (2002), to mention a few. However, all but the latter are inapplicable to the online setting.

Taking advantage of this insight our sparsification method starts with an empty dictionary $\mathcal{D}_0 = \{\}$. It observes the sequence of states $\mathbf{x}_0, \mathbf{x}_1, \ldots$ one state at a time. $\mathbf{x}_t$ is admitted into the dictionary only if its feature space image $\boldsymbol{\phi}(\mathbf{x}_t)$ cannot be approximated sufficiently well by combining the images of states already in the dictionary – $\mathcal{D}_{t-1} = \{\tilde{\mathbf{x}}_1, \ldots, \tilde{\mathbf{x}}_{m_{t-1}}\}$. Given the set of $m_{t-1}$ linearly independent dictionary feature vectors at time $t-1$, $\{\boldsymbol{\phi}(\tilde{\mathbf{x}}_j)\}_{j=1}^{m_{t-1}}$, we seek the least squares approximation of $\boldsymbol{\phi}(\mathbf{x}_t)$ in terms of this set. It can be easily verified that this approximation is given by the solution to the following quadratic minimization problem (Engel, 2005, Section 2.2):

$$\min_{\boldsymbol{a}} \left\{ \boldsymbol{a}^\top \tilde{\mathbf{K}}_{t-1} \boldsymbol{a} - 2\boldsymbol{a}^\top \tilde{\mathbf{k}}_{t-1}(\mathbf{x}_t) + k_{tt} \right\}, \tag{3.20}$$

where $\tilde{\mathbf{K}}_{t-1}$ is the kernel matrix of the dictionary states a time $t-1$ (i.e., $[\tilde{\mathbf{K}}_{t-1}]_{i,j} = k(\tilde{\mathbf{x}}_i, \tilde{\mathbf{x}}_j)$ with $i, j = 1, \ldots, m_{t-1}$), $\tilde{\mathbf{k}}_{t-1}(\mathbf{x}) = (k(\tilde{\mathbf{x}}_1, \mathbf{x}), \ldots, k(\tilde{\mathbf{x}}_{m_{t-1}}, \mathbf{x}))^\top$ and $k_{tt} = k(\mathbf{x}_t, \mathbf{x}_t)$. The solution to (3.20) is $\boldsymbol{a}_t = \tilde{\mathbf{K}}_{t-1}^{-1} \tilde{\mathbf{k}}_{t-1}(\mathbf{x}_t)$, and substituting it back into (3.20) we obtain the squared error (or distance in $\mathcal{H}$) incurred by the approximation,

$$\delta_t = k_{tt} - \tilde{\mathbf{k}}_{t-1}(\mathbf{x}_t)^\top \boldsymbol{a}_t = k_{tt} - \tilde{\mathbf{k}}_{t-1}(\mathbf{x}_t)^\top \tilde{\mathbf{K}}_{t-1}^{-1} \tilde{\mathbf{k}}_{t-1}(\mathbf{x}_t). \tag{3.21}$$

If $\delta_t > \nu$, where $\nu$ is an accuracy threshold parameter, we add $\mathbf{x}_t$ to the dictionary, set $\boldsymbol{a}_t = (0, \ldots, 1)^\top$ and $\delta_t = 0$, since $\boldsymbol{\phi}(\mathbf{x}_t)$ is exactly represented by a term in the dictionary, namely itself. If $\delta_t \leq \nu$ the dictionary remains unchanged. Either way we are assured that all the feature vectors corresponding to the states seen until time $t$ can be approximated by the dictionary at time $t$ with a maximum squared error $\nu$, namely

$$\boldsymbol{\phi}(\mathbf{x}_i) = \sum_{j=1}^{m_i} a_{i,j} \boldsymbol{\phi}(\tilde{\mathbf{x}}_j) + \boldsymbol{\phi}_i^{res} \quad \text{where } \|\boldsymbol{\phi}_i^{res}\|^2 \leq \nu. \tag{3.22}$$

It is easy to show that $\delta_t$ may also be interpreted as the variance of the value $V(\mathbf{x}_t)$ of the current state $\mathbf{x}_t$, given the values of the current dictionary points $V(\tilde{\mathbf{x}}_1), \ldots, V(\tilde{\mathbf{x}}_{|\mathcal{D}_{t-1}|})$. This provides a complementary probabilistic viewpoint on our choice of sparsification criterion. Put explicitly, $\mathbf{x}_t$ will be added to the dictionary if, assuming that the values of the current dictionary points were *exactly* known, the remaining uncertainty in the value at $\mathbf{x}_t$ (quantified by the conditional variance) is still larger than $\nu$. Again, if $\mathbf{x}_t$ is added to the dictionary this conditional variance vanishes, and so does $\delta_t$.

In order to be able to efficiently compute $\boldsymbol{a}_t$ at each time step, we need to update $\tilde{\mathbf{K}}_t^{-1}$ whenever the dictionary is appended with a new state. This is done via the partitioned matrix inversion formula (see Appendix B.4):

$$\tilde{\mathbf{K}}_t = \begin{bmatrix} \tilde{\mathbf{K}}_{t-1} & \tilde{\mathbf{k}}_{t-1}(\mathbf{x}_t) \\ \tilde{\mathbf{k}}_{t-1}(\mathbf{x}_t)^\top & k_{tt} \end{bmatrix} \quad \Rightarrow \quad \tilde{\mathbf{K}}_t^{-1} = \frac{1}{\delta_t} \begin{bmatrix} \delta_t \tilde{\mathbf{K}}_{t-1}^{-1} + \hat{\boldsymbol{a}}_t \hat{\boldsymbol{a}}_t^\top & -\hat{\boldsymbol{a}}_t \\ -\hat{\boldsymbol{a}}_t^\top & 1 \end{bmatrix}, \tag{3.23}$$

where $\hat{\boldsymbol{a}}_t = \tilde{\mathbf{K}}_{t-1}^{-1} \tilde{\mathbf{k}}_{t-1}(\mathbf{x}_t)$. Note that $\hat{\boldsymbol{a}}_t$ is identical to the vector $\boldsymbol{a}_t$ computed in solving (3.20) (i.e., before it was reset to $(0, \ldots, 0, 1)^\top$), so there is no need to recompute it.

Defining the matrices[12] $[\mathbf{A}_t]_{i,j} = a_{i,j}$, $\boldsymbol{\Phi}_t = [\boldsymbol{\phi}(\mathbf{x}_1), \ldots, \boldsymbol{\phi}(\mathbf{x}_t)]$, and $\boldsymbol{\Phi}_t^{res} = [\boldsymbol{\phi}_1^{res}, \ldots, \boldsymbol{\phi}_t^{res}]$ we may write Eq. (3.22) for all time-steps up to $t$, concisely as

$$\boldsymbol{\Phi}_t = \tilde{\boldsymbol{\Phi}}_t \mathbf{A}_t^\top + \boldsymbol{\Phi}_t^{res}. \tag{3.24}$$

12. Due to the sequential nature of the algorithm, for $j > m_i$, $[\mathbf{A}_t]_{i,j} = 0$.

By premultiplying (3.24) with its transpose we obtain a decomposition of the full $t \times t$ kernel matrix $\mathbf{K}_t = \mathbf{\Phi}_t^\top \mathbf{\Phi}_t$ into two matrices:

$$\mathbf{K}_t = \mathbf{A}_t \tilde{\mathbf{K}}_t \mathbf{A}_t^\top + \mathbf{K}_t^{res} \tag{3.25}$$

where $\tilde{\mathbf{K}}_t = \tilde{\mathbf{\Phi}}_t^\top \tilde{\mathbf{\Phi}}_t$. The matrix $\mathbf{A}_t \tilde{\mathbf{K}}_t \mathbf{A}_t^\top$ is a rank $m_t$ approximation of $\mathbf{K}_t$. It can be shown that the norm of the residual matrix $\mathbf{K}_t^{res}$ is bounded from above by a factor linear in $\sqrt{\nu}$. Consequently we make the following approximations (The notation $\overset{\sqrt{\nu}}{\approx}$ means that the norm of the difference between the terms on either side of this sign is bounded by a constant linear in $\sqrt{\nu}$):

$$\mathbf{K}_t \overset{\sqrt{\nu}}{\approx} \mathbf{A}_t \tilde{\mathbf{K}}_t \mathbf{A}_t^\top \quad , \quad \mathbf{k}_t(\mathbf{x}) \overset{\sqrt{\nu}}{\approx} \mathbf{A}_t \tilde{\mathbf{k}}_t(\mathbf{x}). \tag{3.26}$$

We note that the computational cost per time step of this sparsification algorithm is $O(m_t^2)$ which, assuming $m_t$ does not depend asymptotically on $t$, is independent of time[13]. Our online semi-parametric version of the GPTD algorithm draws its computational leverage from the low rank approximation provided by this sparsification algorithm. For further details on properties of this algorithm, as well as on its connection with kernel principal components analysis (kernel PCA), see Engel (2005).

We are now ready to incorporate this sparsification method into the recursive updates of the GPTD posterior moments derived in the preceding sections (As long as $\mathbf{\Sigma}_t$ is unspecified, the discussion here pertains to both the deterministic and MC-GPTD models). Substituting the approximations (3.26) into the exact GP solution (3.10) we obtain

$$\begin{aligned} \hat{V}_t(\mathbf{x}) &\overset{\sqrt{\nu}}{\approx} \boldsymbol{\alpha}_t^\top \mathbf{A}_t \tilde{\mathbf{k}}_t(\mathbf{x}) = \tilde{\boldsymbol{\alpha}}_t^\top \tilde{\mathbf{k}}_t(\mathbf{x}), \\ P_t(\mathbf{x}, \mathbf{x}') &\overset{\sqrt{\nu}}{\approx} k(\mathbf{x}, \mathbf{x}') - \mathbf{k}_t(\mathbf{x})^\top \mathbf{A}_t^\top \mathbf{C}_t \mathbf{A}_t \mathbf{k}_t(\mathbf{x}') = k(\mathbf{x}, \mathbf{x}') - \tilde{\mathbf{k}}_t(\mathbf{x})^\top \tilde{\mathbf{C}}_t \tilde{\mathbf{k}}_t(\mathbf{x}'), \end{aligned} \tag{3.27}$$

where we used the definitions

$$\begin{aligned} \tilde{\mathbf{H}}_t &= \mathbf{H}_t \mathbf{A}_t, & \tilde{\mathbf{Q}}_t &= \left( \tilde{\mathbf{H}}_t \tilde{\mathbf{K}}_t \tilde{\mathbf{H}}_t^\top + \mathbf{\Sigma}_t \right)^{-1}, \\ \tilde{\boldsymbol{\alpha}}_t &= \tilde{\mathbf{H}}_t^\top \tilde{\mathbf{Q}}_t \mathbf{r}_{t-1}, & \tilde{\mathbf{C}}_t &= \tilde{\mathbf{H}}_t^\top \tilde{\mathbf{Q}}_t \tilde{\mathbf{H}}_t. \end{aligned} \tag{3.28}$$

Note that the parameters we are required to store and update in order to evaluate the posterior mean and covariance are now $\tilde{\boldsymbol{\alpha}}_t$ and $\tilde{\mathbf{C}}_t$, the dimensions of which are $m_t \times 1$ and $m_t \times m_t$, respectively. Here we quote the updates for the MC-GPTD model. The complete derivation may be found in Appendix A.3.

At each time step, the current sampled state $\mathbf{x}_t$ may either be left out of the dictionary ($\mathcal{D}_t = \mathcal{D}_{t-1}$), or added to it ($\mathcal{D}_t = \mathcal{D}_{t-1} \cup \{\mathbf{x}_t\}$). In the latter case the dimensions of $\tilde{\boldsymbol{\alpha}}$ and $\tilde{\mathbf{C}}$ increase by 1. The updates in each case are a little different. In either one of the two cases we use the definition

$$\Delta \tilde{\mathbf{k}}_t = \tilde{\mathbf{k}}_{t-1}(\mathbf{x}_{t-1}) - \gamma \tilde{\mathbf{k}}_{t-1}(\mathbf{x}_t).$$

13. We can bound $m_t$ under mild assumptions on the kernel and the space $\mathcal{X}$, see Engel (2005) for details.

**Algorithm 2** A recursive sparse Monte-Carlo GPTD algorithm

**Parameters:** $\nu$

**Initialize** $\mathcal{D}_0 = \{\mathbf{x}_0\}$, $\tilde{\mathbf{K}}_0^{-1} = 1/k(\mathbf{x}_0, \mathbf{x}_0)$, $\boldsymbol{a}_0 = (1)$, $\tilde{\boldsymbol{\alpha}}_0 = 0$, $\tilde{\mathbf{C}}_0 = 0$, $\tilde{\mathbf{c}}_0 = 0$, $d_0 = 0$, $1/s_0 = 0$

**for** $t = 1, 2, \ldots$

**observe** $\mathbf{x}_{t-1}$, $r_{t-1}$, $\mathbf{x}_t$

$\boldsymbol{a}_t = \tilde{\mathbf{K}}_{t-1}^{-1} \tilde{\mathbf{k}}_{t-1}(\mathbf{x}_t)$

$\delta_t = k(\mathbf{x}_t, \mathbf{x}_t) - \tilde{\mathbf{k}}_{t-1}(\mathbf{x}_t)^\top \boldsymbol{a}_t$

$\boldsymbol{\Delta}\tilde{\mathbf{k}}_t = \tilde{\mathbf{k}}_{t-1}(\mathbf{x}_{t-1}) - \gamma \tilde{\mathbf{k}}_{t-1}(\mathbf{x}_t)$

$d_t = \frac{\gamma \sigma_{t-1}^2}{s_{t-1}} d_{t-1} + r_{t-1} - \boldsymbol{\Delta}\tilde{\mathbf{k}}_t^\top \tilde{\boldsymbol{\alpha}}_{t-1}$

**if** $\delta_t > \nu$

compute $\tilde{\mathbf{K}}_t^{-1}$ (3.23)

$\boldsymbol{a}_t = (0, \ldots, 1)^\top$

$\tilde{\mathbf{h}}_t = (\boldsymbol{a}_{t-1}, -\gamma)^\top$

$\Delta k_{tt} = \boldsymbol{a}_{t-1}^\top \left( \tilde{\mathbf{k}}_{t-1}(\mathbf{x}_{t-1}) - 2\gamma \tilde{\mathbf{k}}_{t-1}(\mathbf{x}_t) \right) + \gamma^2 k_{tt}$

$\tilde{\mathbf{c}}_t = \frac{\gamma \sigma_{t-1}^2}{s_{t-1}} \begin{pmatrix} \tilde{\mathbf{c}}_{t-1} \\ 0 \end{pmatrix} + \tilde{\mathbf{h}}_t - \begin{pmatrix} \tilde{\mathbf{C}}_{t-1} \boldsymbol{\Delta}\tilde{\mathbf{k}}_t \\ 0 \end{pmatrix}$

$s_t = \sigma_{t-1}^2 + \gamma^2 \sigma_t^2 + \Delta k_{tt} - \boldsymbol{\Delta}\tilde{\mathbf{k}}_t^\top \tilde{\mathbf{C}}_{t-1} \boldsymbol{\Delta}\tilde{\mathbf{k}}_t + \frac{2\gamma \sigma_{t-1}^2}{s_{t-1}} \tilde{\mathbf{c}}_{t-1}^\top \boldsymbol{\Delta}\tilde{\mathbf{k}}_t - \frac{\gamma^2 \sigma_{t-1}^4}{s_{t-1}}$

$\tilde{\boldsymbol{\alpha}}_{t-1} = \begin{pmatrix} \tilde{\boldsymbol{\alpha}}_{t-1} \\ 0 \end{pmatrix}$

$\tilde{\mathbf{C}}_{t-1} = \begin{bmatrix} \tilde{\mathbf{C}}_{t-1} & \mathbf{0} \\ \mathbf{0}^\top & 0 \end{bmatrix}$

**else**

$\tilde{\mathbf{h}}_t = \boldsymbol{a}_{t-1} - \gamma \boldsymbol{a}_t$

$\Delta k_{tt} = \tilde{\mathbf{h}}_t^\top \boldsymbol{\Delta}\tilde{\mathbf{k}}_t$

$\tilde{\mathbf{c}}_t = \frac{\gamma \sigma_{t-1}^2}{s_{t-1}} \tilde{\mathbf{c}}_t + \tilde{\mathbf{h}}_t - \tilde{\mathbf{C}}_{t-1} \boldsymbol{\Delta}\tilde{\mathbf{k}}_t$

$s_t = \sigma_{t-1}^2 + \gamma^2 \sigma_t^2 + \boldsymbol{\Delta}\tilde{\mathbf{k}}_t^\top \left( \tilde{\mathbf{c}}_t + \frac{\gamma \sigma^2}{s_{t-1}} \tilde{\mathbf{c}}_{t-1} \right) - \frac{\gamma^2 \sigma_{t-1}^4}{s_{t-1}}$

**end if**

$\tilde{\boldsymbol{\alpha}}_t = \tilde{\boldsymbol{\alpha}}_{t-1} + \frac{\tilde{\mathbf{c}}_t}{s_t} d_t$

$\tilde{\mathbf{C}}_t = \tilde{\mathbf{C}}_{t-1} + \frac{1}{s_t} \tilde{\mathbf{c}}_t \tilde{\mathbf{c}}_t^\top$

**end for**

**return** $\mathcal{D}_t$, $\tilde{\boldsymbol{\alpha}}_t$, $\tilde{\mathbf{C}}_t$

**Case 1.** Dictionary remains unchanged: $\mathcal{D}_t = \mathcal{D}_{t-1}$:

$$\tilde{\boldsymbol{\alpha}}_t = \tilde{\boldsymbol{\alpha}}_{t-1} + \frac{\tilde{\mathbf{c}}_t}{s_t} d_t \,, \quad \tilde{\mathbf{C}}_t = \tilde{\mathbf{C}}_{t-1} + \frac{1}{s_t} \tilde{\mathbf{c}}_t \tilde{\mathbf{c}}_t^\top, \tag{3.29}$$

where

$$\begin{aligned} \tilde{\mathbf{c}}_t &= \frac{\gamma \sigma_{t-1}^2}{s_{t-1}} \tilde{\mathbf{c}}_{t-1} + \tilde{\mathbf{h}}_t - \tilde{\mathbf{C}}_{t-1} \Delta\tilde{\mathbf{k}}_t \\ d_t &= \frac{\gamma \sigma_{t-1}^2}{s_{t-1}} d_{t-1} + r_{t-1} - \Delta\tilde{\mathbf{k}}_t^\top \tilde{\boldsymbol{\alpha}}_{t-1} \\ s_t &= \sigma_{t-1}^2 + \gamma^2 \sigma_t^2 + \Delta\tilde{\mathbf{k}}_t^\top \left( \tilde{\mathbf{c}}_t + \frac{\gamma \sigma^2}{s_{t-1}} \tilde{\mathbf{c}}_{t-1} \right) - \frac{\gamma^2 \sigma_{t-1}^4}{s_{t-1}}, \end{aligned} \tag{3.30}$$

with the definitions $\tilde{\mathbf{h}}_t = \boldsymbol{a}_{t-1} - \gamma \boldsymbol{a}_t$ and $\Delta k_{tt} = \tilde{\mathbf{h}}_t^\top \Delta\tilde{\mathbf{k}}_t$.

**Case 2.** Dictionary is appended with $\mathbf{x}_t$: $\mathcal{D}_t = \mathcal{D}_{t-1} \cup \{\mathbf{x}_t\}$:

$$\tilde{\boldsymbol{\alpha}}_t = \begin{pmatrix} \tilde{\boldsymbol{\alpha}}_{t-1} \\ 0 \end{pmatrix} + \frac{\tilde{\mathbf{c}}_t}{s_t} d_t \,, \quad \tilde{\mathbf{C}}_t = \begin{bmatrix} \tilde{\mathbf{C}}_{t-1} & ,\mathbf{0} \\ \mathbf{0}^\top & ,0 \end{bmatrix} + \frac{1}{s_t} \tilde{\mathbf{c}}_t \tilde{\mathbf{c}}_t^\top, \tag{3.31}$$

where

$$\begin{aligned} \tilde{\mathbf{c}}_t &= \frac{\gamma \sigma_{t-1}^2}{s_{t-1}} \begin{pmatrix} \tilde{\mathbf{c}}_{t-1} \\ 0 \end{pmatrix} + \tilde{\mathbf{h}}_t - \begin{pmatrix} \tilde{\mathbf{C}}_{t-1} \Delta\tilde{\mathbf{k}}_t \\ 0 \end{pmatrix}, \\ d_t &= \frac{\gamma \sigma_{t-1}^2}{s_{t-1}} d_{t-1} + r_{t-1} - \Delta\tilde{\mathbf{k}}_t^\top \tilde{\boldsymbol{\alpha}}_{t-1}, \\ s_t &= \sigma_{t-1}^2 + \gamma^2 \sigma_t^2 + \Delta k_{tt} - \Delta\tilde{\mathbf{k}}_t^\top \tilde{\mathbf{C}}_{t-1} \Delta\tilde{\mathbf{k}}_t + \frac{2\gamma \sigma_{t-1}^2}{s_{t-1}} \tilde{\mathbf{c}}_{t-1}^\top \Delta\tilde{\mathbf{k}}_t - \frac{\gamma^2 \sigma_{t-1}^4}{s_{t-1}}, \end{aligned} \tag{3.32}$$

with the definitions

$$\tilde{\mathbf{h}}_t = \begin{pmatrix} \boldsymbol{a}_{t-1} \\ 0 \end{pmatrix} - \gamma \boldsymbol{a}_t = \begin{pmatrix} \boldsymbol{a}_{t-1} \\ -\gamma \end{pmatrix}, \quad \Delta k_{tt} = \boldsymbol{a}_{t-1}^\top \left( \tilde{\mathbf{k}}_{t-1}(\mathbf{x}_{t-1}) - 2\gamma \tilde{\mathbf{k}}_{t-1}(\mathbf{x}_t) \right) + \gamma^2 k_{tt}.$$

The pseudocode for this algorithm is provided in Algorithm 2.

### 3.5 Parametric GPTD Models

Another approach that allows us to bound the computational cost per time-step is to parameterize the value function by a finite set of random variables $W = (w_1, \ldots, w_n)^\top$ in such a way as to preserve the linearity of the generative model. This leads to the following parameterization

$$V(\mathbf{x}) = \sum_{j=1}^{n} \phi_j(\mathbf{x}) w_j = \boldsymbol{\phi}(\mathbf{x})^\top W, \tag{3.33}$$

where $\boldsymbol{\phi}(\mathbf{x}) = (\phi_1(\mathbf{x}), \ldots, \phi_n(\mathbf{x}))^\top$ is a vector of $n$ fixed basis functions. In this parametric model, the randomness in the value function is due to $W$. In order to use Bayesian reasoning, a prior must be placed on $W$, which, with no loss of generality[14], we take to be distributed

14. For any other prior covariance matrix, the corresponding set of basis functions may be linearly transformed into a new set of basis functions for which the parameter covariance matrix is $\mathbf{I}$, and the value covariance is unchanged.

as $\mathcal{N}(\mathbf{0}, \mathbf{I})$, where $\mathbf{0}$ is a null vector (of zeros), and $\mathbf{I}$ is the identity matrix. This induces a Gaussian prior distribution over the value random process, which makes it a GP, with easily computed moments:

$$\begin{aligned}\mathbf{E}[V(\mathbf{x})] &= \boldsymbol{\phi}(\mathbf{x})^\top \mathbf{E}(W) = 0\\ \mathbf{Cov}[V(\mathbf{x}), V(\mathbf{x}')] &= \boldsymbol{\phi}(\mathbf{x})^\top \mathbf{E}[WW^\top]\boldsymbol{\phi}(\mathbf{x}') = \boldsymbol{\phi}(\mathbf{x})^\top \boldsymbol{\phi}(\mathbf{x}').\end{aligned}$$

The parametric approach is most suitable in cases where domain knowledge may be used to construct a set of basis functions the span of which is known to include good approximations of the true value function. However, if in the parametric approach the set of basis functions employed satisfies $\boldsymbol{\phi}(\mathbf{x})^\top \boldsymbol{\phi}(\mathbf{x}') = k(\mathbf{x}, \mathbf{x}')$ for almost all $\mathbf{x}, \mathbf{x}' \in \mathcal{X}$, then the two approaches – parametric and nonparametric (kernel based) – become equivalent.

The values of the states seen until time-step $t$ satisfy

$$V_t = \boldsymbol{\Phi}_t^\top W, \quad \text{where } \boldsymbol{\Phi}_t = [\boldsymbol{\phi}(\mathbf{x}_0), \ldots, \boldsymbol{\phi}(\mathbf{x}_t)]\,. \tag{3.34}$$

The posterior moments, conditioned on an observed sequence of rewards $\mathbf{r}_{t-1} = (r_0, \ldots, r_{t-1})^\top$, are now easily derived:

$$\hat{\mathbf{w}}_t \stackrel{\text{def}}{=} \mathbf{E}\left[W | R_{t-1} = \mathbf{r}_{t-1}\right] = \boldsymbol{\Phi}_t \mathbf{H}_t^\top \mathbf{Q}_t \mathbf{r}_{t-1}, \tag{3.35}$$

$$\mathbf{P}_t \stackrel{\text{def}}{=} \mathbf{Cov}\left[W | R_{t-1} = \mathbf{r}_{t-1}\right] = \mathbf{I} - \boldsymbol{\Phi}_t \mathbf{H}_t^\top \mathbf{Q}_t \mathbf{H}_t \boldsymbol{\Phi}_t^\top, \tag{3.36}$$

$$\text{where } \mathbf{Q}_t = \left(\mathbf{H}_t \boldsymbol{\Phi}_t^\top \boldsymbol{\Phi}_t \mathbf{H}_t^\top + \boldsymbol{\Sigma}_t\right)^{-1}.$$

For convenience, let us define the $d \times 1$ vector $\boldsymbol{\Delta\phi}_t$ and the $d \times t$ matrix $\boldsymbol{\Delta\Phi}_t$, as

$$\boldsymbol{\Delta\phi}_t \stackrel{\text{def}}{=} \boldsymbol{\phi}(\mathbf{x}_{t-1}) - \gamma \boldsymbol{\phi}(\mathbf{x}_t), \quad \text{and } \boldsymbol{\Delta\Phi}_t \stackrel{\text{def}}{=} \boldsymbol{\Phi}_t \mathbf{H}_t^\top = \left[\boldsymbol{\Delta\phi}_1, \quad \ldots \quad, \boldsymbol{\Delta\phi}_t\right], \tag{3.37}$$

respectively[15]. Using the Transparency Lemma (Appendix B.5), it is easy to verify that alternative, more computationally efficient expressions for the posterior moments may be obtained (assuming $t > n$, where $n$ is number of basis functions):

$$\mathbf{P}_t = \left(\boldsymbol{\Phi}_t \mathbf{H}_t^\top \boldsymbol{\Sigma}_t^{-1} \mathbf{H}_t \boldsymbol{\Phi}_t^\top + \mathbf{I}\right)^{-1} = \left(\boldsymbol{\Delta\Phi}_t \boldsymbol{\Sigma}_t^{-1} \boldsymbol{\Delta\Phi}_t^\top + \mathbf{I}\right)^{-1}, \tag{3.38}$$

$$\hat{\mathbf{w}}_t = \mathbf{P}_t \boldsymbol{\Phi}_t \mathbf{H}_t^\top \boldsymbol{\Sigma}_t^{-1} \mathbf{r}_{t-1} = \mathbf{P}_t \boldsymbol{\Delta\Phi}_t \boldsymbol{\Sigma}_t^{-1} \mathbf{r}_{t-1}. \tag{3.39}$$

Whereas the expressions in Eq. 3.35, 3.36 involve inverting a $t \times t$ matrix, here the inversion is of an $n \times n$ matrix.

It is instructive to note that the same kind of analysis also applies to the nonparametric case. Here again, it is easy to derive alternative expressions for $\boldsymbol{\alpha}_t$ and $\mathbf{C}_t$:

$$\boldsymbol{\alpha}_t = \left(\mathbf{H}_t^\top \boldsymbol{\Sigma}_t^{-1} \mathbf{H}_t \mathbf{K}_t + \mathbf{I}\right)^{-1} \mathbf{H}_t^\top \boldsymbol{\Sigma}_t^{-1} \mathbf{r}_{t-1}, \tag{3.40}$$

$$\mathbf{C}_t = \left(\mathbf{H}_t^\top \boldsymbol{\Sigma}_t^{-1} \mathbf{H}_t \mathbf{K}_t + \mathbf{I}\right)^{-1} \mathbf{H}_t^\top \boldsymbol{\Sigma}_t^{-1} \mathbf{H}_t. \tag{3.41}$$

15. If $\mathbf{x}_t$ is the last state of an episode, then $\boldsymbol{\Delta\phi}_t = \boldsymbol{\phi}(\mathbf{x}_t)$.

In contrast to the parametric case, these alternative expressions do not confer any computational advantage over the original ones. However, this alternative representation of the Bayesian solution serves as a link to the nonparametric MAP and ML solutions, as we shall see below.

Our next goal is to obtain the parametric updates for the MC-GPTD model. For brevity, we forgo the derivation of a batch algorithm, which may be found in Engel (2005), and proceed with the derivation of a recursive algorithm.

Rather than starting from the alternative expressions (3.38, 3.39), we base the derivation of the updates on the original expressions of (3.35, 3.36). The derivation is given in full in Appendix A.1. The resulting recursive updates are:[16]

$$\hat{\mathbf{w}}_t = \hat{\mathbf{w}}_{t-1} + \frac{1}{s_t}\mathbf{p}_t d_t, \quad \mathbf{P}_t = \mathbf{P}_{t-1} - \frac{1}{s_t}\mathbf{p}_t\mathbf{p}_t^\top, \tag{3.42}$$

where

$$\begin{aligned}
\mathbf{p}_t &= \frac{\gamma\sigma_{t-1}^2}{s_{t-1}}\mathbf{p}_{t-1} + \mathbf{P}_{t-1}\left(\boldsymbol{\phi}(\mathbf{x}_{t-1}) - \gamma\boldsymbol{\phi}(\mathbf{x}_t)\right) \\
d_t &= \frac{\gamma\sigma_{t-1}^2}{s_{t-1}}d_{t-1} + r_{t-1} - \left(\boldsymbol{\phi}(\mathbf{x}_{t-1}) - \gamma\boldsymbol{\phi}(\mathbf{x}_t)\right)^\top \hat{\mathbf{w}}_{t-1} \\
s_t &= \sigma_{t-1}^2 + \gamma^2\sigma_t^2 - \frac{\gamma^2\sigma_{t-1}^4}{s_{t-1}} + \left(\mathbf{p}_t + \frac{\gamma\sigma_{t-1}^2}{s_{t-1}}\mathbf{p}_{t-1}\right)^\top \left(\boldsymbol{\phi}(\mathbf{x}_{t-1}) - \gamma\boldsymbol{\phi}(\mathbf{x}_t)\right)
\end{aligned} \tag{3.43}$$

It can be easily verified that these recursions should be initialized as follows:

$$\hat{\mathbf{w}}_0 = \mathbf{0}, \quad \mathbf{P}_0 = \mathbf{I}, \quad \mathbf{p}_0 = \mathbf{0}, \quad d_0 = 0, \quad 1/s_0 = 0.$$

Algorithm 3 provides the pseudocode for this algorithm.

The analysis carried out in Section 3.3 involving a noise whitening transformation applies equally well here, with the conclusion being that at the end of an episode the solution arrived at by MC-GPTD is the same as the solution attained by performing GP regression (parametric, this time) on MC samples of the discounted return. Therefore, at the end of an episode, the parametric solution for the MC-GPTD model, assuming a constant noise variance $\sigma^2$, is given by

$$\begin{aligned}
\hat{\mathbf{w}}_{t+1} &= \mathbf{E}\left(W|Y_t = \mathbf{y}_t\right) = \left(\boldsymbol{\Phi}_t\boldsymbol{\Phi}_t^\top + \sigma^2\mathbf{I}\right)^{-1}\boldsymbol{\Phi}_t\mathbf{y}_t, \\
\mathbf{P}_{t+1} &= \mathbf{Cov}\left(W|Y_t = \mathbf{y}_t\right) = \sigma^2\left(\boldsymbol{\Phi}_t\boldsymbol{\Phi}_t^\top + \sigma^2\mathbf{I}\right)^{-1}.
\end{aligned} \tag{3.44}$$

## 4. Connections with Other TD Methods

In Engel et al. (2003) we derived a generative model, similar to the one derived here, which is applicable only to MDPs with deterministic transitions. Surprisingly, the only

16. Since our model is essentially the Kalman Filter (KF) model, less the state dynamics, we could have used the measurement updates of the KF (Scharf, 1991, Chapters 7, 8) to derive the GPTD updates. This, however, is complicated by our correlated noise model, which would require us to introduce auxiliary state variables. Instead, we take the possibly longer, but more instructive route, and derive the updates directly.

**Algorithm 3** A recursive parametric Monte-Carlo GPTD algorithm

**Initialize** $\hat{\mathbf{w}}_0 = \mathbf{0}$, $\mathbf{P}_0 = \mathbf{I}$, $\mathbf{p}_0 = \mathbf{0}$, $d_0 = 0$, $1/s_0 = 0$
**for** $t = 1, 2, \ldots$
  **observe** $\mathbf{x}_{t-1}$, $r_{t-1}$, $\mathbf{x}_t$
  $\boldsymbol{\Delta\phi}_t = \boldsymbol{\phi}(\mathbf{x}_{t-1}) - \gamma\boldsymbol{\phi}(\mathbf{x}_t)$
  $\mathbf{p}_t = \frac{\gamma\sigma_{t-1}^2}{s_{t-1}}\mathbf{p}_{t-1} + \mathbf{P}_{t-1}\boldsymbol{\Delta\phi}_t$
  $d_t = \frac{\gamma\sigma_{t-1}^2}{s_{t-1}}d_{t-1} + r_{t-1} - \boldsymbol{\Delta\phi}_t^\top \hat{\mathbf{w}}_{t-1}$
  $s_t = \sigma_{t-1}^2 + \gamma^2\sigma_t^2 - \frac{\gamma^2\sigma_{t-1}^4}{s_{t-1}} + \left(\mathbf{p}_t + \frac{\gamma\sigma_{t-1}^2}{s_{t-1}}\mathbf{p}_{t-1}\right)^\top \boldsymbol{\Delta\phi}_t$
  $\hat{\mathbf{w}}_t = \hat{\mathbf{w}}_{t-1} + \frac{1}{s_t}\mathbf{p}_t d_t$
  $\mathbf{P}_t = \mathbf{P}_{t-1} - \frac{1}{s_t}\mathbf{p}_t\mathbf{p}_t^\top$
**end for**
**return** $\hat{\mathbf{w}}_t$, $\mathbf{P}_t$

difference between these two models is in the measurement noise statistics. Specifically, in the model presented in Engel et al. (2003) the noise covariance matrix is diagonal, whereas in the MC-GPTD model presented here, it is tridiagonal. We are therefore inclined to adopt a broader view of GPTD as a general GP-based framework for Bayesian modeling of value functions. This, more general view, encompasses linear statistical models of the form $R_{t-1} = \mathbf{H}_t V_t + N_t$, with $\mathbf{H}_t$ given by (3.6) (or (3.12) at the end of an episode), a Gaussian prior placed on $V$, and an arbitrary zero-mean Gaussian noise process $N$. No doubt, most such models will be meaningless from a value estimation point of view, while others may not admit efficient recursive algorithms for computing the posterior value moments. However, if the noise covariance $\boldsymbol{\Sigma}_t$ is suitably chosen, and if it is additionally *simple* in some way, then we may be able to derive such a recursive algorithm to compute complete posterior value distributions, online. In this section we show that by employing alternative forms of noise covariance, we are able to obtain GP-based variants of LSTD($\lambda$) (Bradtke and Barto, 1996; Boyan, 1999). In order to demonstrate this, we need to consider first the Maximum Likelihood (ML) solution for our MC-GPTD model.

### 4.1 A Maximum Likelihood Variant of MC-GPTD

In certain cases, one may prefer to forgo specifying a prior over the weight vector $W$. In such cases, one can no longer perform Bayesian analysis, but it is nevertheless still possible to perform classical ML inference to find $\hat{\mathbf{w}}^{ML}$ – the value of $W$ for which the observed data is most likely, according to the statistical model of Eq. 3.5. Due to the normality assumption, ignoring terms independent of $W$, the log-likelihood of $W$, based on the sequence of observed rewards is

$$\log \mathbb{P}(R_{t-1} = \mathbf{r}_{t-1}|W) \propto -\left(\mathbf{r}_{t-1} - \boldsymbol{\Delta\Phi}_t^\top W\right)^\top \boldsymbol{\Sigma}_t^{-1}\left(\mathbf{r}_{t-1} - \boldsymbol{\Delta\Phi}_t^\top W\right). \tag{4.1}$$

The maximum-likelihood (ML) problem therefore reduces to the minimization of a simple quadratic form:

$$\min_{\mathbf{w}}\left\{\left(\mathbf{r}_{t-1} - \boldsymbol{\Delta\Phi}_t^\top \mathbf{w}\right)^\top \boldsymbol{\Sigma}_t^{-1}\left(\mathbf{r}_{t-1} - \boldsymbol{\Delta\Phi}_t^\top \mathbf{w}\right)\right\}, \tag{4.2}$$

the solution of which is given by (recall that $\mathbf{\Delta\Phi}_t = \mathbf{\Phi}_t\mathbf{H}_t^\top$)

$$\hat{\mathbf{w}}_t^{ML} = \left(\mathbf{\Phi}_t\mathbf{H}_t^\top\mathbf{\Sigma}_t^{-1}\mathbf{H}_t\mathbf{\Phi}_t^\top\right)^{-1}\mathbf{\Phi}_t\mathbf{H}_t^\top\mathbf{\Sigma}_t^{-1}\mathbf{r}_{t-1}. \tag{4.3}$$

As we saw in Section 3.3, at the end of an episode $\mathbf{H}_t$ is invertible and is given by Eq. 3.18. The inverse noise covariance, assuming for simplicity a constant residual variance $\mathbf{Var}\left[\Delta V\right] = \sigma^2\mathbf{I}$, is given by

$$\mathbf{\Sigma}_{t-1}^{-1} = \frac{1}{\sigma^2}{\mathbf{H}_{t+1}^\top}^{-1}\mathbf{H}_{t+1}{}^{-1}.$$

Therefore, Eq. 4.3 becomes

$$\hat{\mathbf{w}}_{t+1}^{ML} = \left(\mathbf{\Phi}_t\mathbf{\Phi}_t^\top\right)^{-1}\mathbf{\Phi}_t\mathbf{H}_{t+1}^{-1}\mathbf{r}_t \tag{4.4}$$

From the formula for $\mathbf{H}_{t+1}^{-1}$ we infer that $\mathbf{H}_{t+1}^{-1}\mathbf{r}_t = \mathbf{y}_t$, where the $i$'th component of $\mathbf{y}_t$ is $y_i = \sum_{j=i}^{t}\gamma^{j-i}r_j$ (see Eq. 3.17). We therefore conclude that our parametric ML solution is precisely the LSTD(1) solution. It is equally straightforward to show that TD(1) may be derived as a gradient ascent method for maximizing the log-likelihood (4.1), with a fixed residual variance $\sigma^2$. The derivation may be found in Engel (2005).

As with any other ML estimator, it should be cautioned that the ML variant will tend to overfit the data, until the number of samples considerably exceeds the dimensionality of the feature space $\boldsymbol{\phi}(\mathcal{X})$ (i.e., the number of independent adjustable parameters). GPTD solutions avoid overfitting by virtue of the regularizing influence of the prior. For this reason, for all practical purposes, we consider MC-GPTD, in its original form, to be preferable to its ML variant, namely, LSTD(1).

A natural question to ask at this point is whether LSTD($\lambda$) for $\lambda < 1$ may also be derived as a ML solution arising from some other GPTD generative model. We address this issue next.

### 4.2 LSTD($\lambda$) as a Maximum Likelihood Algorithm

LSTD($\lambda$) with linear function approximation solves the following set of linear equations (Bradtke and Barto, 1996; Boyan, 1999):

$$\mathbf{B}_t\hat{\mathbf{w}}_t = \mathbf{b}_t, \quad \text{where } \mathbf{B}_t = \sum_{i=0}^{t-1}\mathbf{z}_i\left(\boldsymbol{\phi}(\mathbf{x}_i) - \gamma\boldsymbol{\phi}(\mathbf{x}_{i+1})\right)^\top \quad \text{and } \mathbf{b}_t = \sum_{i=0}^{t-1}\mathbf{z}_i r_i. \tag{4.5}$$

In both TD($\lambda$) and LSTD($\lambda$) a vector $\mathbf{z}_t$ of eligibilities is maintained, using the recursion

$$\mathbf{z}_t = \gamma\lambda\mathbf{z}_{t-1} + \boldsymbol{\phi}(\mathbf{x}_t), \quad \text{with } \mathbf{z}_0 = \boldsymbol{\phi}(\mathbf{x}_0).$$

These eligibility vectors may be arranged in an $n \times t$ *eligibility matrix* $\mathbf{Z}_t^{(\lambda)}$, defined by

$$\mathbf{Z}_t^{(\lambda)} = \left[\mathbf{z}_0, \ldots, \mathbf{z}_{t-1}\right].$$

Using $\mathbf{Z}_t^{(\lambda)}$ and the definition of $\mathbf{\Delta\Phi}_t$ (3.37), we may write $\mathbf{B}_t$ and $\mathbf{b}_t$ from Eq. 4.5, as

$$\mathbf{B}_t = \mathbf{Z}_t^{(\lambda)}\mathbf{\Delta\Phi}_t^\top, \quad \text{and} \quad \mathbf{b}_t = \mathbf{Z}_t^{(\lambda)}\mathbf{r}_{t-1}.$$

$\mathbf{B}_t$ is an $n \times n$ square matrix, and for $t \geq n$, assuming no state is visited twice, it is of full rank (i.e., its null-space is $\{\mathbf{0}\}$ and $\mathbf{B}_t^{-1}$ exists). We may therefore premultiply Eq. 4.5 by $\mathbf{B}_t^\top \mathbf{G}_t$, where $\mathbf{G}_t$ is an arbitrary $d \times d$ symmetric positive-definite matrix, with no change in the solution whatsoever. This results in

$$\boldsymbol{\Delta\Phi}_t {\mathbf{Z}_t^{(\lambda)}}^\top \mathbf{G}_t \mathbf{Z}_t^{(\lambda)} \boldsymbol{\Delta\Phi}_t^\top \hat{\mathbf{w}}_t = \boldsymbol{\Delta\Phi}_t {\mathbf{Z}_t^{(\lambda)}}^\top \mathbf{G}_t \mathbf{Z}_t^{(\lambda)} \mathbf{r}_{t-1}. \tag{4.6}$$

Eq. 4.6 is the set of normal equations resulting from the least squares problem

$$\min_{\mathbf{w}} \left\{ \left( \mathbf{r}_{t-1} - \boldsymbol{\Delta\Phi}_t^\top \mathbf{w} \right)^\top {\mathbf{Z}_t^{(\lambda)}}^\top \mathbf{G}_t \mathbf{Z}_t^{(\lambda)} \left( \mathbf{r}_{t-1} - \boldsymbol{\Delta\Phi}_t^\top \mathbf{w} \right) \right\}, \tag{4.7}$$

which, in turn, may be interpreted as arising from a maximum likelihood problem over jointly Gaussian random variables:

$$\max_{\mathbf{w}} \left\{ \log \mathbb{P}(\mathbf{r}_{t-1} | \mathbf{w}) \right\}, \quad \text{where}$$

$$\mathbb{P}(\mathbf{r}_{t-1} | \mathbf{w}) \propto \exp\left( -\frac{1}{2} \left( \mathbf{r}_{t-1} - \boldsymbol{\Delta\Phi}_t^\top \mathbf{w} \right)^\top {\mathbf{Z}_t^{(\lambda)}}^\top \mathbf{G}_t \mathbf{Z}_t^{(\lambda)} \left( \mathbf{r}_{t-1} - \boldsymbol{\Delta\Phi}_t^\top \mathbf{w} \right) \right).$$

Note that, regardless of the choice of $\mathbf{G}_t$, the minimum in (4.7) is zero, since $\mathbf{Z}_t^{(\lambda)} \left( \mathbf{r}_{t-1} - \boldsymbol{\Delta\Phi}_t^\top \mathbf{w} \right) = 0$ is a set of $n$ equations in $n$ variables, and therefore $\mathbf{Z}_t^{(\lambda)} \left( \mathbf{r}_{t-1} - \boldsymbol{\Delta\Phi}_t^\top \mathbf{w} \right)$ can be made to vanish.

Comparing Eq. 4.7 with Eq. 4.2, we conclude that LSTD($\lambda$) implicitly assumes that rewards and values are connected by a Gaussian process model of the by now familiar (parametric) GPTD form, namely:

$$R_{t-1} = \mathbf{H}_t V_t + N_t = \mathbf{H}_t \boldsymbol{\Phi}_t^\top W + N_t,$$

in which the inverse covariance matrix of the the noise process $N_t$ satisfies

$$\boldsymbol{\Sigma}_t^{-1} \propto {\mathbf{Z}_t^{(\lambda)}}^\top \mathbf{G}_t \mathbf{Z}_t^{(\lambda)}.$$

The noise covariance matrix $\boldsymbol{\Sigma}_t$ itself does not generally exist, since, for $t > n$, ${\mathbf{Z}_t^{(\lambda)}}^\top \mathbf{G}_t \mathbf{Z}_t^{(\lambda)}$ is rank deficient (with rank $n$). This means that there are $t - n$ orthogonal directions in $\mathbb{R}^t$ (defined by an orthogonal basis for the null space of ${\mathbf{Z}_t^{(\lambda)}}^\top \mathbf{G}_t \mathbf{Z}_t^{(\lambda)}$) in which the variance is infinite, which means that deviations in these directions have no effect on the likelihood. The nonexistence of $\boldsymbol{\Sigma}_t$ is less of a problem than it might seem, since the ML, MAP and Bayesian solutions may all be written entirely in terms of $\boldsymbol{\Sigma}_t^{-1}$ (see Eq. 4.2, 3.38, 3.39), and it is therefore only important that $\boldsymbol{\Sigma}_t^{-1}$ exists. In any event, the inverse of $\boldsymbol{\Sigma}_t^{-1} + \varepsilon \mathbf{I}$ for any $\varepsilon > 0$ exists and may be used as a substitute for $\boldsymbol{\Sigma}_t$.

It now becomes a simple exercise to derive a new set of GPTD algorithms, which are based on the LSTD($\lambda$) noise model, i.e., using $\boldsymbol{\Sigma}_t^{-1} = \frac{1}{\sigma^2} {\mathbf{Z}_t^{(\lambda)}}^\top \mathbf{G}_t \mathbf{Z}_t^{(\lambda)}$. For the parametric GPTD($\lambda$) model, the posterior moments, given by (Eq. 3.38, 3.39), are

$$\mathbf{P}_t^{(\lambda)} = \sigma^2 \left( \boldsymbol{\Delta\Phi}_t {\mathbf{Z}_t^{(\lambda)}}^\top \mathbf{G}_t \mathbf{Z}_t^{(\lambda)} \boldsymbol{\Delta\Phi}_t^\top + \sigma^2 \mathbf{I} \right)^{-1}, \tag{4.8}$$

$$\hat{\mathbf{w}}_t^{(\lambda)} = \frac{1}{\sigma^2} \mathbf{P}_t^{(\lambda)} \boldsymbol{\Delta\Phi}_t {\mathbf{Z}_t^{(\lambda)}}^\top \mathbf{G}_t \mathbf{Z}_t^{(\lambda)} \mathbf{r}_{t-1}.$$

Whereas for the nonparametric GPTD($\lambda$), we have (see Eq. 3.40, 3.41)

$$\boldsymbol{\alpha}_t^{(\lambda)} = \left(\mathbf{H}_t^\top {\mathbf{Z}_t^{(\lambda)}}^\top \mathbf{G}_t \mathbf{Z}_t^{(\lambda)} \mathbf{H}_t \mathbf{K}_t + \sigma^2 \mathbf{I}\right)^{-1} \mathbf{H}_t^\top {\mathbf{Z}_t^{(\lambda)}}^\top \mathbf{G}_t \mathbf{Z}_t^{(\lambda)} \mathbf{r}_{t-1}, \tag{4.9}$$

$$\mathbf{C}_t^{(\lambda)} = \left(\mathbf{H}_t^\top {\mathbf{Z}_t^{(\lambda)}}^\top \mathbf{G}_t \mathbf{Z}_t^{(\lambda)} \mathbf{H}_t \mathbf{K}_t + \sigma^2 \mathbf{I}\right)^{-1} \mathbf{H}_t^\top {\mathbf{Z}_t^{(\lambda)}}^\top \mathbf{G}_t \mathbf{Z}_t^{(\lambda)} \mathbf{H}_t.$$

In the parametric case, for an arbitrary $\mathbf{G}_t$, taking the limit $\sigma \to 0$ brings us back to the familiar LSTD($\lambda$). Moreover, for any $\sigma > 0$, and for $\lambda = 1$ there exists a particular choice of $\mathbf{G}_t$ for which we are returned to the MC-GPTD model of Sections 3.1 and 3.5, as we show next for the parametric model.

### 4.3 GPTD($\lambda$)

In this section we consider a possible choice for the matrix $\mathbf{G}_t$ in Eq. 4.7. First, however, let us recall that the eligibility vectors satisfy $\mathbf{z}_t = \sum_{i=0}^{t} (\gamma\lambda)^{t-i} \boldsymbol{\phi}(\mathbf{x}_i)$. We may therefore write $\mathbf{Z}_{t+1}^{(\lambda)}$ as

$$\mathbf{Z}_{t+1}^{(\lambda)} = \boldsymbol{\Phi}_t \begin{bmatrix} 1 & \gamma\lambda & (\gamma\lambda)^2 & \dots & (\gamma\lambda)^t \\ 0 & 1 & \gamma\lambda & \dots & (\gamma\lambda)^{t-1} \\ \vdots & & & & \vdots \\ 0 & 0 & 0 & \dots & 1 \end{bmatrix}. \tag{4.10}$$

Suppose that $\mathbf{x}_t$ is the final state in an episode. Recalling $\mathbf{H}_{t+1}^{-1}$ from Eq. 3.18, for $\lambda = 1$ we may write

$$\mathbf{Z}_{t+1}^{(1)} = \boldsymbol{\Phi}_t \mathbf{H}_{t+1}^{-1}$$

$\mathbf{P}_{t+1}^{(1)}$ of Eq. 4.8 then becomes

$$\mathbf{P}_{t+1}^{(1)} = \sigma^2 \left(\boldsymbol{\Phi}_t \boldsymbol{\Phi}_t^\top \mathbf{G}_t \boldsymbol{\Phi}_t \boldsymbol{\Phi}_t^\top + \sigma^2 \mathbf{I}\right)^{-1},$$

while $\hat{\mathbf{w}}_{t+1}^{(1)}$ becomes

$$\hat{\mathbf{w}}_{t+1}^{(1)} = \left(\boldsymbol{\Phi}_t \boldsymbol{\Phi}_t^\top \mathbf{G}_t \boldsymbol{\Phi}_t \boldsymbol{\Phi}_t^\top + \sigma^2 \mathbf{I}\right)^{-1} \boldsymbol{\Phi}_t \boldsymbol{\Phi}_t^\top \mathbf{G}_t \boldsymbol{\Phi}_t \mathbf{H}_{t+1}^{-1} \mathbf{r}_t.$$

These expressions for the posterior moments suggest that a reasonable choice for $\mathbf{G}_t$ should satisfy $\lim_{\lambda \to 1} \mathbf{G}_t(\lambda) = \left(\boldsymbol{\Phi}_t \boldsymbol{\Phi}_t^\top\right)^{-1}$, since then, for $\lambda = 1$, we are returned to the MC-GPTD solution,

$$\mathbf{P}_{t+1}^{(1)} = \sigma^2 \left(\boldsymbol{\Phi}_t \boldsymbol{\Phi}_t^\top + \sigma^2 \mathbf{I}\right)^{-1} \quad \text{and } \hat{\mathbf{w}}_{t+1}^{(1)} = \left(\boldsymbol{\Phi}_t \boldsymbol{\Phi}_t^\top + \sigma^2 \mathbf{I}\right)^{-1} \boldsymbol{\Phi}_t \mathbf{H}_{t+1}^{-1} \mathbf{r}_t.$$

These solutions are recognized as the parametric MC-GPTD posterior moments (see Eq. 3.44). If additionally we require that $\mathbf{G}_t$ be independent of $\lambda$, we are left with the unique choice: $\mathbf{G}_t = \left(\boldsymbol{\Phi}_t \boldsymbol{\Phi}_t^\top\right)^{-1}$.

Our argument in favor of this particular choice of the inverse noise covariance (through the choice of $\mathbf{G}_t$) is based on the limiting behavior of the resulting solutions when $\lambda$ approaches 1 (in which case it coincides with the MC-GPTD solution), and when $\sigma$ approaches

0 (in which case it coincides with LSTD($\lambda$)). Apart from these continuity arguments, the question of how a noise model such as the one suggested above may be theoretically justified, remains unanswered. In particular, this means that, even with complete knowledge of the underlying MDP, the optimal choice of the parameter $\lambda$ is currently an issue that has to be resolved empirically. However, the same could be said about LSTD($\lambda$). As far as we are aware, the only way LSTD($\lambda$) with linear function approximation may be justified, or indeed derived (apart from the post hoc asymptotic bounds of Tsitsiklis and Van Roy (1996)), is either by analogy with the lookup-table based LSTD($\lambda$), or by a continuity argument based on the fact that as $\lambda$ approaches 1, the resulting algorithm approaches LSTD(1), which is known to perform least-squares regression on Monte-Carlo samples of the discounted return (Bertsekas and Tsitsiklis, 1996; Sutton and Barto, 1998; Boyan, 1999).

## 5. Policy Improvement

So far, we have restricted our attention to the problem of policy evaluation. In this section we build upon our earlier algorithms to construct algorithms that improve policies rather than evaluate them. This would allow us to solve the complete RL problem, namely, that of finding a near-optimal policy for the MDP in question. Here we will propose two types of algorithms, one based on the policy-iteration algorithm (Bertsekas and Tsitsiklis, 1996), and the other on the SARSA algorithm Sutton and Barto (1998). In either case, a GPTD policy evaluation algorithms will be used as a sub-component of the complete algorithm. However, in order to avoid having to learn and use the MDP's transition-model $p$ in the policy improvement steps, we will need to make a simple modification in our GPTD algorithms that would allow them to learn state-action values.

The algorithms presented so far were based on viewing a MDP controlled by a fixed stationary policy $\mu$ as a Markov reward process (MRP) (Puterman, 1994), with the actions ignored, except through their effect on the state transition probabilities. The state space of this MRP is $\mathcal{X}$ (the same as the MDP's state space) and the state transition probability density is $p^\mu(\mathbf{x}'|\mathbf{x}) = \int_{\mathcal{U}} d\mathbf{u}\mu(\mathbf{u}|\mathbf{x})p(\mathbf{x}'|\mathbf{u},\mathbf{x})$. However, it is possible to define another MRP $\mathcal{M}'$, for which the state space is $\mathcal{X}' = \mathcal{X} \times \mathcal{U}$ (i.e., the MDP's state space augmented by its action space), a transition probability density $p'(\mathbf{x}', \mathbf{u}'|\mathbf{x}, \mathbf{u}) = p(\mathbf{x}'|\mathbf{x}, \mathbf{u})\mu(\mathbf{u}'|\mathbf{x}')$, an initial state probability density $p'_0(\mathbf{x}, \mathbf{u}) = p_0(\mathbf{x})\mu(\mathbf{u}|\mathbf{x})$, and a reward probability density $q'(r|\mathbf{x}, \mathbf{u}) = q(r|\mathbf{x})$. When a policy evaluation algorithm is applied to the MRP $\mathcal{M}'$, the result is an algorithm that estimates state-action values. The main benefit in estimating state-action values $Q(\mathbf{x}, \mathbf{u})$ is that instead of having to evaluate $\max_{\mathbf{u}} \mathbf{E}_{\mathbf{x}'|\mathbf{u},\mathbf{x}} \hat{V}(\mathbf{x}')$ all we need to do in order to improve the policy at $\mathbf{x}$ is to evaluate $\max_{\mathbf{u}} \hat{Q}(\mathbf{x}, \mathbf{u})$, which does not involve an expectation (and hence does not require a model). In the parametric GPTD models, all that is required is that the user define a set of state-action basis functions $\{\phi_i : \mathcal{X} \times \mathcal{U} \to \mathbb{R}\}_{i=1}^n$, and proceed as usual. Here we will treat the more interesting non-parametric case. In the nonparametric kernel-based models, one needs to define a covariance kernel function over state-action pairs, i.e., $k : (\mathcal{X} \times \mathcal{U}) \times (\mathcal{X} \times \mathcal{U}) \to \mathbb{R}$. Since states and actions are quite different entities it makes sense to decompose $k$ into a state-kernel $k_x$ and an action-kernel $k_u$. It is well known that the set of positive-definite kernels is closed under several unary and binary operations, such as scaling by a positive constant, addition and multiplication, to name a few (Schölkopf and Smola, 2002; Shawe-Taylor and Cristianini,

2004). As an illustration let us set[17]

$$k(\mathbf{x}, \mathbf{u}, \mathbf{x}', \mathbf{u}') = k_x(\mathbf{x}, \mathbf{x}')k_u(\mathbf{u}, \mathbf{u}'). \tag{5.1}$$

Just as the state-kernel codes our prior beliefs concerning correlations between the values of different states, so should the action-kernel code our prior beliefs on value correlations between different actions.

The first algorithm we propose is a simple adaptation of approximate policy iteration (Bertsekas and Tsitsiklis, 1996). The algorithm works by maintaining two GPTD policy evaluators, $G_0$ and $G_1$. The state-action value function maintained by $G_0$ is used to determine a policy according to which actions are selected, while $G_1$ is used to evaluate this policy. Once $G_1$ has acquired a sufficiently precise estimate of the state-action value function, the roles are switched, and now $G_0$ is used to evaluate the policy determined by the state-action values of $G_1$. The policies used in these iterations should approach greedy policies with respect to the value estimates maintained by their respective policy evaluators. It is therefore useful if the state-action kernel function is chosen in such a way as to allow an easy computation of $\max_{\mathbf{u}} \hat{Q}(\mathbf{x}, \mathbf{u})$. If the set of actions $\mathcal{U}$ is finite this can be done in time linear in the size of $\mathcal{U}$. Later we will see an example where $\mathcal{U}$ is continuous and the maximization may be performed analytically. Pseudocode of a GPTD-based approximate policy iteration (GPTD-API) algorithm is shown in Algorithm 4.

**Algorithm 4** A GPTD-based approximate policy iteration algorithm (GPTD-API)

**Input:** MDP $\mathcal{M}$ , convergence threshold $\eta$
**Initialize:** GPTD policy evaluators $G_0$, $G_1$, greed parameter $\varepsilon$, $iter = 0$
$done = false$
**while** not *done*
  $iter = iter + 1$, $i =$ mod $(iter, 2)$, $j = 1 - i$
  $\mu(G_j, \varepsilon) =$ semi-greedy policy w.r.t. to $G_j$
  $G_i = \mathbf{GPTD}(\mathcal{M}, \mu(G_j, \varepsilon))$
  $done = \|G_i - G_j\| \leq \eta$
**end while**
**return** $G_i$

Provided that the error in the approximate value-function returned by GPTD is small and improper policies are avoided (for $\gamma = 1$), it is possible to show that API produces a sequence of policies that eventually perform almost as well as the optimal policy; Bertsekas and Tsitsiklis (see 1996, Chapter 6).

Another approach to policy improvement is based on the SARSA algorithm (Rummery and Niranjan, 1994; Sutton and Barto, 1998); see also Bertsekas and Tsitsiklis (1996), where such algorithms are referred to as optimistic policy iteration. SARSA is a fairly straightforward extension of the TD algorithm in which state-action values are estimated, while at the same time actions are selected semi-greedily based on the current estimate of the state-action values. The reason this is referred to as optimistic policy iteration is the fact

17. A simpler choice would be a sum of the two types of kernel functions. However, this would result in a greedy policy that ignores the state in its choice of action.

that the policy is being updated constantly, without waiting for an adequate approximation of its state-action value function to be learned. Some degree of optimism is therefore in order. SARSA was originally formulated with TD updates, however the same idea may be applied to other online policy evaluation algorithms. Here we propose to use GPTD as our online policy evaluator, resulting in the GPSARSA algorithm. Algorithm 5 is the pseudocode for a GPSARSA algorithm that is based on the non-parametric MC-GPTD algorithm (Algorithm 1). Sparsified and parametric versions are easily derived by using Algorithm 2 and 3, respectively, as one's starting point[18].

**Algorithm 5** Nonparametric GPSARSA

**Initialize** $\boldsymbol{\alpha}_0 = \mathbf{0}$, $\mathbf{C}_0 = 0$, $\mathcal{D}_0 = \{\mathbf{x}_0, \mathbf{u}_0\}$, $\mathbf{c}_0 = \mathbf{0}$, $d_0 = 0$, $1/s_0 = 0$

**for** $t = 1, 2, \ldots$

**observe** $\mathbf{x}_{t-1}$, $\mathbf{u}_{t-1}$, $r_{t-1}$, $\mathbf{x}_t$

$\mathbf{u}_t = \mathbf{SemiGreedyAction}(\mathbf{x}_t, \mathcal{D}_{t-1}, \boldsymbol{\alpha}_{t-1}, \mathbf{C}_{t-1})$

$\mathbf{h}_t = (0, \ldots, 1, -\gamma)^\top$

$\boldsymbol{\Delta}\mathbf{k}_t = \mathbf{k}_{t-1}(\mathbf{x}_{t-1}, \mathbf{u}_{t-1}) - \gamma \mathbf{k}_{t-1}(\mathbf{x}_t, \mathbf{u}_t)$

$\Delta k_{tt} = k((\mathbf{x}_{t-1}, \mathbf{u}_{t-1}), (\mathbf{x}_{t-1}, \mathbf{u}_{t-1})) - 2\gamma k((\mathbf{x}_{t-1}, \mathbf{u}_{t-1}), (\mathbf{x}_t, \mathbf{u}_t) + \gamma^2 k((\mathbf{x}_t, \mathbf{u}_t), (\mathbf{x}_t, \mathbf{u}_t))$

$d_t = \frac{\gamma \sigma_{t-1}^2}{s_{t-1}} d_{t-1} + r_{t-1} - \boldsymbol{\Delta}\mathbf{k}_t^\top \boldsymbol{\alpha}_{t-1}$

$\mathbf{c}_t = \frac{\gamma \sigma_{t-1}^2}{s_{t-1}} \begin{pmatrix} \mathbf{c}_{t-1} \\ 0 \end{pmatrix} + \mathbf{h}_t - \begin{pmatrix} \mathbf{C}_{t-1} \boldsymbol{\Delta}\mathbf{k}_t \\ 0 \end{pmatrix}$

$s_t = \sigma_{t-1}^2 + \gamma^2 \sigma_t^2 - \frac{\gamma^2 \sigma_{t-1}^4}{s_{t-1}} + \Delta k_{tt} - \boldsymbol{\Delta}\mathbf{k}_t^\top \mathbf{C}_{t-1} \boldsymbol{\Delta}\mathbf{k}_t + \frac{2\gamma \sigma_{t-1}^2}{s_{t-1}} \mathbf{c}_{t-1}^\top \boldsymbol{\Delta}\mathbf{k}_t$

$\boldsymbol{\alpha}_t = \begin{pmatrix} \boldsymbol{\alpha}_{t-1} \\ 0 \end{pmatrix} + \frac{\mathbf{c}_t}{s_t} d_t$

$\mathbf{C}_t = \begin{bmatrix} \mathbf{C}_{t-1} & \mathbf{0} \\ \mathbf{0}^\top & 0 \end{bmatrix} + \frac{1}{s_t} \mathbf{c}_t \mathbf{c}_t^\top$

$\mathcal{D}_t = \mathcal{D}_{t-1} \cup \{(\mathbf{x}_t, \mathbf{u}_t)\}$

**end for**

**return** $\boldsymbol{\alpha}_t$, $\mathbf{C}_t$, $\mathcal{D}_t$

In Algorithm 5 the function **SemiGreedyAction** is used as an unspecified black box. In several common semi-greedy rules one is required to solve $\operatorname{argmax}_{\mathbf{u}} \hat{Q}(\mathbf{x}_t, \mathbf{u})$. It was mentioned above that, even when $\mathcal{U}$ is continuous, by appropriate choices of the way actions are represented and of the kernel function, it may be possible to solve this maximization problem in closed form. To demonstrate this, let us consider a common class of RL problems that involve a robot that is faced with a spatial navigation task. For simplicity let us assume that the state-space $\mathcal{X}$ is a bounded, connected subset of $n$-dimensional Euclidean space ($n$ is usually 2 or 3), and that the actions are steps of length 1 in any direction. The action space $\mathcal{U}$ is therefore either the $n$-dimensional unit sphere. The actual state transitions are noisy and possibly scaled versions of the actions, subject to constraints imposed by the environment, such as walls, obstacles, etc. We choose to represent an action by its

18. The deterministic-MDP algorithms presented earlier are not suitable to serve as substrates to GPSARSA algorithms, since semi-greedy action selection procedures typically include some stochasticity. Moreover, since the policy being followed is constantly changing, the transitions in the induced MRP $\mathcal{M}'$ are not deterministic, even without any explicit stochasticity in the action selection rule.

corresponding unit vector $\mathbf{u}$. Let us use the factored state-action kernel from Eq. 5.1. We leave the space kernel $k_x$ unspecified and focus on the action kernel. We define $k_u$ as follows,

$$k_u(\mathbf{u}, \mathbf{u}') = \frac{1-b}{2}\mathbf{u}^\top\mathbf{u}' + \frac{1+b}{2},$$

where $b$ is a constant in $[0, 1]$. Since $\mathbf{u}^\top\mathbf{u}'$ is the cosine of the angle between $\mathbf{u}$ and $\mathbf{u}'$, $k_u(\mathbf{u}, \mathbf{u}')$ attains its maximal value of 1 when the two actions are the same, and its minimal value of $b$ when the actions point in opposite directions. Setting $b$ to a positive value is reasonable, since even opposite actions from the same state are expected, a priori, to have positively correlated values. However, the most valuable feature of this kernel is its linearity, which makes it possible to maximize the value estimate over the actions analytically.

Assume that the agent is running a nonparametric sparse GPSARSA, so that it maintains a dictionary of state-action pairs $\mathcal{D}_t = \{(\tilde{\mathbf{x}}_i, \tilde{\mathbf{u}}_i)\}_{i=1}^m$. The agent's value estimate for its current state $\mathbf{x}$ and an arbitrary action $\mathbf{u}$ is

$$\begin{aligned}\hat{V}(\mathbf{x}, \mathbf{u}) &= \sum_{i=1}^m \tilde{\boldsymbol{\alpha}}_i k_x(\tilde{\mathbf{x}}_i, \mathbf{x}) k_u(\tilde{\mathbf{u}}_i, \mathbf{u}) \\ &= \sum_{i=1}^m \tilde{\boldsymbol{\alpha}}_i k_x(\tilde{\mathbf{x}}_i, \mathbf{x}) \left(\frac{1-b}{2}\tilde{\mathbf{u}}_i^\top\mathbf{u} + \frac{1+b}{2}\right).\end{aligned}$$

Maximizing this expression with respect to $\mathbf{u}$ amounts to maximizing $\sum_{i=1}^m \beta_i(\mathbf{x})\tilde{\mathbf{u}}_i^\top\mathbf{u}$ subject to the constraint $\|\mathbf{u}\| = 1$, where $\beta_i(\mathbf{x}) \stackrel{\text{def}}{=} \tilde{\boldsymbol{\alpha}}_i k_x(\tilde{\mathbf{x}}_i, \mathbf{x})$. Solving this problem using the suitable Lagrangian results in the greedy action $\mathbf{u}^* = \frac{1}{\lambda}\sum_{i=1}^m \beta_i(\mathbf{x})\tilde{\mathbf{u}}_i$ where $\lambda$ is a normalizing constant. It is also possible to maximize the variance estimate. This may be used to select non-greedy exploratory moves, by choosing the action whose value the agent is least certain about. Performing this maximization amounts to solving a $2 \times 2$ or $3 \times 3$ Eigenvalue problem.

## 6. Summary and Discussion

In this paper we presented a Bayesian formulation of the fundamental RL problem of policy evaluation. This was done by casting the probabilistic relation between the value function and the observed rewards as a linear statistical generative model over normally distributed random processes (Eq. 3.5). The Bayesian solution for the policy evaluation problem is embodied in the posterior value distribution conditioned on the observed state-reward or state-action-reward trajectory. This posterior is computed by employing Bayes' rule to "invert" the set of equations provided by the generative model. Apart from the value estimates given by the posterior mean, the Bayesian solution also provides the variance of values around this mean, supplying the practitioner with a measure of the reliability of value estimates.

In Rasmussen and Kuss (2004) an alternative approach to employing GPs in RL is proposed. In that paper a model based approach is proposed, in which one GP is used to learn the MDP's transition model, while another is used to estimate the value. This leads to an inherently off-line algorithm, which is not capable of interacting with the controlled system directly and updating its estimates as additional data arrive. Moreover, the state

dynamics is assumed to be factored, in the sense that each state coordinate is assumed to evolve in time independently of all others. Lastly, for analytical tractability, the selection of covariance kernels is restricted to either polynomial or Gaussian kernels (or a mixture of the two), due to the need to integrate over products of GPs.

Bayesian methods, such as ours, typically require the user to impart more domain specific knowledge to the learning system, than do classical methods. Such domain knowledge may be encoded in the prior and in the measurement model (e.g., in the measurement noise covariance). In many cases such domain knowledge is available, at least in a rough form. For instance, in the RL context, the user will usually know whether the MDP under investigation follows deterministic or stochastic dynamics. She will usually also have a good idea on the range of values of the rewards, and what makes states and actions similar or different from each other. All of this information may be incorporated into a GPTD generative model, and capitalized upon in the subsequent Bayesian posterior analysis. We made use of these inherent qualities of the Bayesian approach by defining different generative models, one which is suitable only for MDPs with deterministic transitions (in Engel et al. (2003)), and another suitable for general MDPs (in Engel et al. (2005) and the present paper). In the nonparametric case, the kernel function $k$ is used to encode notions of similarity and dissimilarity between states, or more correctly, how the values of different states are correlated[19]. Having said that, there are well known methods in the Bayesian and GP literature for learning the hyperparameters of a Bayesian model (a.k.a. model selection). These include maximizing the likelihood of the observations with respect to these hyperparameters, or alternatively, treating them as random variables (parameterized by hyper-hyperparameters) and performing Bayesian inference on them as well (this is known as a hierarchical Bayesian model). As these techniques are adequately covered elsewhere (e.g., Mackay, 1997; Gibbs and MacKay, 1997; Williams, 1999; Seeger, 2003) we chose not to elaborate on this subject here.

We showed in Sections 4.2 and 4.3 that the familiar family of LSTD($\lambda$) algorithms may in fact be viewed as classical parametric maximum-likelihood algorithms based on statistical generative models having the same structure as our GPTD model, and employing a specific form of $\lambda$-dependent noise covariance. As such, LSTD($\lambda$) solutions are limit points of GPTD solutions. In this sense, our GPTD framework subsumes all other TD methods employing linear function approximation architectures. This, we hope, should alleviate concerns regarding the applicability of the assumptions underlying the GPTD model, as the same assumptions are also implicitly made by LSTD($\lambda$) and TD($\lambda$).

By employing a nonparametric, kernel-based extension of the conventional parametric linear statistical model we were able to derive kernel-based variants of our algorithms. These kernel algorithms offer a degree of representational flexibility that is impossible to attain using standard linear FA architectures, in which a set of basis functions must be specified in advance. By using a kernel sparsification method we were able to obtain, for the first time to the best of our knowledge, *practical nonparametric online algorithms* for solving the policy evaluation problem. The constructive process through which these algorithms sparsify the full nonparametric solution may be thought of as a process of basis construction,

19. For instance, using the Gaussian kernel $c \exp\left(-\|\mathbf{x}' - \mathbf{x}\|^2/(2\sigma^2)\right)$, apart from encoding the belief that nearby states are more similar than states that are far apart, also states that, a priori, we believe that the variance of the values at any point in $\mathcal{X}$ equals $c$.

in which new basis functions are added if and when they are needed to maintain a sufficiently accurate approximation of the complete non-sparse solution (the basis functions are the kernel functions evaluated at the dictionary states, namely $\{k(\tilde{\mathbf{x}}_j, \cdot)\}_{j=1}^{|\mathcal{D}|}$). As such they enjoy both the representational flexibility of the complete nonparametric solution while also taking advantage of redundancies in the Hilbert space induced by the kernel. When the dictionary constructed by these algorithms ceases to grow (and we know that this will eventually happen, see Engel (2005)), then from that point on the algorithm may be regarded as a (linear) parametric algorithm.

Taking the frequentist perspective, the posterior mean provided by the Bayesian analysis may be viewed as the solution of a Tikhonov-regularized least-squares problem. Although in the frequentist view, such solutions are considered biased, they are known to be consistent (i.e., asymptotically unbiased), and usually enjoy lower variance than their frequentist counterparts. Specifically, this implies that in the parametric case, in the limit of an infinite trajectory, both GPTD($\lambda$) (including MC-GPTD = GPTD(1)) and LSTD($\lambda$) converge to the same solution (provided the same $\lambda$ is used). This should remove any concerns regarding the convergence of GPTD($\lambda$), as the asymptotic convergence properties of LSTD($\lambda$) are well understood (Tsitsiklis and Van Roy, 1996). When a sparse nonparametric representation is used, we already know that after a finite number of transitions are observed, our sparse algorithms are effectively computing parametric solutions, using a basis consisting of the kernel functions evaluated at the dictionary states. Therefore, the same conclusions regarding the convergence of our parametric methods also apply to our sparse nonparametric methods. More intricate convergence properties of GPs, such as generalization bounds and learning curves (but only in the context of supervised learning), are presently the subject of research in the GP community (e.g., Opper and Vivarelli, 1999; Malzahn and Opper, 2001; Seeger, 2003; Sollich and Williams, 2005). However, the GPTD model is generally more complex than the conventional GP models used in the supervised setting. Deriving results analogous to those mentioned above, in the RL setting, is left as an open direction for future research.

Another contribution is the extension of GPTD to the estimation of state-action values, or Q-values, leading to the policy-improving algorithms of Section 5. We show that learning Q-values makes the task of policy improvement in the absence of a transition model tenable, even when the action space is continuous and multi-dimensional. As mentioned above, we describe the experimental evaluation of these algorithms in a separate companion paper.

The availability of confidence intervals for Q-values significantly expands the repertoire of possible exploration strategies. In finite MDPs, strategies employing such confidence intervals have been experimentally shown to perform more efficiently than conventional $\varepsilon$-greedy or Boltzmann sampling strategies (e.g., Kaelbling, 1993; Dearden et al., 1998; Even-Dar et al., 2003; Strehl and Littman, 2004, 2005). Our approach allows such methods to be applied to infinite MDPs, and it remains to be seen whether significant improvements can be so achieved for realistic problems with continuous space and action spaces.

**Acknowledgments**

We would like to thank Nahum Shimkin, Ishai Menache, Dima Volkinshtein, Peter Szabo for fruitful discussions and insightful comments. Y. E. was supported by an Alberta Ingenuity

fellowship and the Alberta Ingenuity Center for Machine Learning. The work of R. M. was partially supported by EU Project PASCAL, by the Technion VPR fund for promotion of research and by the Ollendorff foundation.

## Appendix A. Monte-Carlo GPTD

### A.1 Parametric Monte-Carlo GPTD Updates

Recall the expressions for the parametric posterior moments:

$$\hat{\mathbf{w}}_t = \mathbf{\Delta\Phi}_t \mathbf{Q}_t \mathbf{r}_{t-1}, \quad \mathbf{P}_t = \mathbf{I} - \mathbf{\Delta\Phi}_t \mathbf{Q}_t \mathbf{\Delta\Phi}_t^\top, \quad \text{where } \mathbf{Q}_t = \left( \mathbf{\Delta\Phi}_t^\top \mathbf{\Delta\Phi}_t + \mathbf{\Sigma}_t \right)^{-1},$$

where we used the definition $\mathbf{\Delta\Phi}_t = \mathbf{\Phi}_t \mathbf{H}_t^\top$. In the sequel we will also use the following definitions:

$$\mathbf{h}_t = (0, \ldots, 1, -\gamma)^\top, \quad \text{and } \mathbf{\Delta}\boldsymbol{\phi}_t = \mathbf{\Phi}_t \mathbf{h}_t = \boldsymbol{\phi}(\mathbf{x}_{t-1}) - \gamma \boldsymbol{\phi}(\mathbf{x}_t).$$

The matrices $\mathbf{\Delta\Phi}_t$, $\mathbf{\Sigma}_t$ and $\mathbf{Q}_t^{-1}$ may be written recursively as follows:

$$\mathbf{\Delta\Phi}_t = \left[ \mathbf{\Delta\Phi}_{t-1}, \quad \mathbf{\Delta}\boldsymbol{\phi}_t \right], \tag{A.1}$$

$$\mathbf{\Sigma}_t = \begin{bmatrix} \mathbf{\Sigma}_{t-1} & , -\gamma \sigma_{t-1}^2 \mathbf{e} \\ -\gamma \sigma_{t-1}^2 \mathbf{e}^\top & , \sigma_{t-1}^2 + \gamma^2 \sigma_t^2 \end{bmatrix}, \quad \text{and}$$

$$\mathbf{Q}_t^{-1} = \begin{bmatrix} \mathbf{Q}_{t-1}^{-1} & , \mathbf{\Delta\Phi}_{t-1}^\top \mathbf{\Delta}\boldsymbol{\phi}_t - \gamma \sigma_{t-1}^2 \mathbf{e} \\ (\mathbf{\Delta\Phi}_{t-1}^\top \mathbf{\Delta}\boldsymbol{\phi}_t - \gamma \sigma_{t-1}^2 \mathbf{e})^\top & , \mathbf{\Delta}\boldsymbol{\phi}_t^\top \mathbf{\Delta}\boldsymbol{\phi}_t + \sigma_{t-1}^2 + \gamma^2 \sigma_t^2 \end{bmatrix}, \tag{A.2}$$

where $\mathbf{e} = (0, \ldots, 0, 1)^\top$. Using the partitioned matrix inversion formula (Scharf, 1991) we may invert $\mathbf{Q}_t^{-1}$ to obtain

$$\mathbf{Q}_t = \frac{1}{s_t} \begin{bmatrix} s_t \mathbf{Q}_{t-1} + \boldsymbol{g}_t \boldsymbol{g}_t^\top & -\boldsymbol{g}_t \\ -\boldsymbol{g}_t^\top & 1 \end{bmatrix},$$

where

$$\boldsymbol{g}_t = \mathbf{Q}_{t-1} \left( \mathbf{\Delta\Phi}_{t-1}^\top \mathbf{\Delta}\boldsymbol{\phi}_t - \gamma \sigma_{t-1}^2 \mathbf{e} \right)$$

$$s_t = \sigma_{t-1}^2 + \gamma^2 \sigma_t^2 + \mathbf{\Delta}\boldsymbol{\phi}_t^\top \mathbf{\Delta}\boldsymbol{\phi}_t - \left( \mathbf{\Delta\Phi}_{t-1}^\top \mathbf{\Delta}\boldsymbol{\phi}_t - \gamma \sigma_{t-1}^2 \mathbf{e} \right)^\top \boldsymbol{g}_t.$$

Let us write the expression for $\hat{\mathbf{w}}_t$:

$$\begin{aligned}
\hat{\mathbf{w}}_t &= \mathbf{\Delta\Phi}_t \mathbf{Q}_t \mathbf{r}_{t-1} \\
&= \frac{1}{s_t} \left[ \mathbf{\Delta\Phi}_{t-1}, \mathbf{\Delta}\boldsymbol{\phi}_t \right] \begin{bmatrix} s_t \mathbf{Q}_{t-1} + \boldsymbol{g}_t \boldsymbol{g}_t^\top & -\boldsymbol{g}_t \\ -\boldsymbol{g}_t^\top & 1 \end{bmatrix} \begin{pmatrix} \mathbf{r}_{t-2} \\ r_{t-1} \end{pmatrix} \\
&= \mathbf{\Delta\Phi}_{t-1} \mathbf{Q}_{t-1} \mathbf{r}_{t-2} + \frac{1}{s_t} \left[ \mathbf{\Delta\Phi}_{t-1}, \quad \mathbf{\Delta}\boldsymbol{\phi}_t \right] \begin{pmatrix} \boldsymbol{g}_t \\ -1 \end{pmatrix} \left( \boldsymbol{g}_t^\top, \quad -1 \right) \begin{pmatrix} \mathbf{r}_{t-2} \\ r_{t-1} \end{pmatrix} \\
&= \hat{\mathbf{w}}_{t-1} + \frac{1}{s_t} \left( \mathbf{\Delta\Phi}_{t-1} \boldsymbol{g}_t - \mathbf{\Delta}\boldsymbol{\phi}_t \right) \left( \boldsymbol{g}_t^\top \mathbf{r}_{t-2} - r_{t-1} \right) \\
&= \hat{\mathbf{w}}_{t-1} + \frac{\mathbf{p}_t}{s_t} d_t,
\end{aligned}$$

where we have defined $d_t = r_{t-1} - \boldsymbol{g}_t^\top \mathbf{r}_{t-2}$ and $\mathbf{p}_t = \boldsymbol{\Delta\phi}_t - \boldsymbol{\Delta\Phi}_{t-1}\boldsymbol{g}_t$.

We treat $\mathbf{P}_t$ similarly:

$$
\begin{aligned}
\mathbf{P}_t &= \mathbf{I} - \boldsymbol{\Delta\Phi}_t \mathbf{Q}_t \boldsymbol{\Delta\Phi}_t^\top \\
&= \mathbf{I} - \frac{1}{s_t}\left[\boldsymbol{\Delta\Phi}_{t-1}, \boldsymbol{\Delta\phi}_t\right] \begin{bmatrix} s_t\mathbf{Q}_{t-1} + \boldsymbol{g}_t\boldsymbol{g}_t^\top & -\boldsymbol{g}_t \\ -\boldsymbol{g}_t^\top & 1 \end{bmatrix} \begin{bmatrix} \boldsymbol{\Delta\Phi}_{t-1}^\top \\ \boldsymbol{\Delta\phi}_t^\top \end{bmatrix} \\
&= \mathbf{I} - \boldsymbol{\Delta\Phi}_{t-1}\mathbf{Q}_{t-1}\boldsymbol{\Delta\Phi}_{t-1}^\top - \frac{1}{s_t}\left[\boldsymbol{\Delta\Phi}_{t-1}, \quad \boldsymbol{\Delta\phi}_t\right] \begin{pmatrix} \boldsymbol{g}_t \\ -1 \end{pmatrix} \left(\boldsymbol{g}_t^\top, \quad -1\right) \begin{bmatrix} \boldsymbol{\Delta\Phi}_{t-1}^\top \\ \boldsymbol{\Delta\phi}_t^\top \end{bmatrix} \\
&= \mathbf{P}_{t-1} - \frac{1}{s_t}\left(\boldsymbol{\Delta\Phi}_{t-1}\boldsymbol{g}_t - \boldsymbol{\Delta\phi}_t\right)\left(\boldsymbol{\Delta\Phi}_{t-1}\boldsymbol{g}_t - \boldsymbol{\Delta\phi}_t\right)^\top \\
&= \mathbf{P}_{t-1} - \frac{1}{s_t}\mathbf{p}_t\mathbf{p}_t^\top.
\end{aligned}
$$

We are not done quite yet, since $d_t$, $\mathbf{p}_t$ and $s_t$ are expressed in terms of several $t$ (or $t-1$) dimensional vectors and matrices, making their computation inefficient. Next, we derive efficient recursive update rules for $d_t$, $\mathbf{p}_t$ and $s_t$, that involve, at most, $n$-dimensional entities. Let us begin with $d_t$:

$$
\begin{aligned}
d_t &= r_{t-1} - \boldsymbol{g}_t^\top \mathbf{r}_{t-2} \\
&= r_{t-1} - \left(\boldsymbol{\Delta\Phi}_{t-1}^\top \boldsymbol{\Delta\phi}_t - \gamma\sigma_{t-1}^2\mathbf{e}\right)^\top \mathbf{Q}_{t-1}\mathbf{r}_{t-2} \\
&= r_{t-1} - \boldsymbol{\Delta\phi}_t^\top \hat{\mathbf{w}}_{t-1} + \frac{\gamma\sigma_{t-1}^2}{s_{t-1}}\left[-\boldsymbol{g}_{t-1}^\top, \quad 1\right] \begin{pmatrix} \mathbf{r}_{t-3} \\ r_{t-2} \end{pmatrix} \\
&= \frac{\gamma\sigma_{t-1}^2}{s_{t-1}} d_{t-1} + r_{t-1} - \left(\boldsymbol{\phi}(\mathbf{x}_{t-1}) - \gamma\boldsymbol{\phi}(\mathbf{x}_t)\right)^\top \hat{\mathbf{w}}_{t-1}.
\end{aligned}
$$

Next, we treat $\mathbf{p}_t$:

$$
\begin{aligned}
\mathbf{p}_t &= \boldsymbol{\Delta\phi}_t - \boldsymbol{\Delta\Phi}_{t-1}\boldsymbol{g}_t \\
&= \boldsymbol{\Delta\phi}_t - \boldsymbol{\Delta\Phi}_{t-1}\mathbf{Q}_{t-1}\left(\boldsymbol{\Delta\Phi}_{t-1}^\top\boldsymbol{\Delta\phi}_t - \gamma\sigma_{t-1}^2\mathbf{e}\right) \\
&= \mathbf{P}_{t-1}\boldsymbol{\Delta\phi}_t + \frac{\gamma\sigma_{t-1}^2}{s_{t-1}}\left[\boldsymbol{\Delta\Phi}_{t-2}, \quad \boldsymbol{\Delta\phi}_{t-1}\right] \begin{bmatrix} -\boldsymbol{g}_{t-1} \\ 1 \end{bmatrix} \\
&= \frac{\gamma\sigma_{t-1}^2}{s_{t-1}}\mathbf{p}_{t-1} + \mathbf{P}_{t-1}\left(\boldsymbol{\phi}(\mathbf{x}_{t-1}) - \gamma\boldsymbol{\phi}(\mathbf{x}_t)\right).
\end{aligned}
$$

Finally, $s_t$:

$$
s_t = \sigma_{t-1}^2 + \gamma^2\sigma_t^2 + \boldsymbol{\Delta\phi}_t^\top\boldsymbol{\Delta\phi}_t - \left(\boldsymbol{\Delta\Phi}_{t-1}^\top\boldsymbol{\Delta\phi}_t - \gamma\sigma_{t-1}^2\mathbf{e}\right)^\top \boldsymbol{g}_t.
$$

The last term on the r.h.s. is

$$\begin{aligned}
&\left(\boldsymbol{\Delta\Phi}_{t-1}^\top \boldsymbol{\Delta\phi}_t - \gamma\sigma_{t-1}^2 \mathbf{e}\right)^\top \boldsymbol{g}_t = \\
&\left(\boldsymbol{\Delta\Phi}_{t-1}^\top \boldsymbol{\Delta\phi}_t - \gamma\sigma_{t-1}^2 \mathbf{e}\right)^\top \mathbf{Q}_{t-1} \left(\boldsymbol{\Delta\Phi}_{t-1}^\top \boldsymbol{\Delta\phi}_t - \gamma\sigma_{t-1}^2 \mathbf{e}\right) = \\
&\boldsymbol{\Delta\phi}_t^\top \left(\mathbf{I} - \mathbf{P}_{t-1}\right) \boldsymbol{\Delta\phi}_t - 2\gamma\sigma_{t-1}^2 \mathbf{e}^\top \mathbf{Q}_{t-1} \boldsymbol{\Delta\Phi}_{t-1}^\top \boldsymbol{\Delta\phi}_t + \gamma^2 \sigma_{t-1}^4 \mathbf{e}^\top \mathbf{Q}_{t-1} \mathbf{e} = \\
&\boldsymbol{\Delta\phi}_t^\top \left(\mathbf{I} - \mathbf{P}_{t-1}\right) \boldsymbol{\Delta\phi}_t - \frac{2\gamma\sigma_{t-1}^2}{s_{t-1}} \left[-\boldsymbol{g}_{t-1}^\top, \quad 1\right] \boldsymbol{\Delta\Phi}_{t-1}^\top \boldsymbol{\Delta\phi}_t + \gamma^2 \sigma_{t-1}^4 \mathbf{e}^\top \mathbf{Q}_{t-1} \mathbf{e} = \\
&\boldsymbol{\Delta\phi}_t^\top \left(\mathbf{I} - \mathbf{P}_{t-1}\right) \boldsymbol{\Delta\phi}_t - \frac{2\gamma\sigma_{t-1}^2}{s_{t-1}} \left(\boldsymbol{\Delta\phi}_{t-1} - \boldsymbol{\Delta\Phi}_{t-2} \boldsymbol{g}_{t-1}\right)^\top \boldsymbol{\Delta\phi}_t + \gamma^2 \sigma_{t-1}^4 \mathbf{e}^\top \mathbf{Q}_{t-1} \mathbf{e} = \\
&\boldsymbol{\Delta\phi}_t^\top \left(\mathbf{I} - \mathbf{P}_{t-1}\right) \boldsymbol{\Delta\phi}_t - \frac{2\gamma\sigma_{t-1}^2}{s_{t-1}} \mathbf{p}_{t-1}^\top \boldsymbol{\Delta\phi}_t + \frac{\gamma^2 \sigma_{t-1}^4}{s_{t-1}}
\end{aligned}$$

Assuming that we already computed $\mathbf{p}_t$, we may substitute

$$\mathbf{P}_{t-1} \boldsymbol{\Delta\phi}_t = \mathbf{p}_t - \frac{\gamma\sigma_{t-1}^2}{s_{t-1}} \mathbf{p}_{t-1}^\top,$$

resulting in our final expression for $s_t$:

$$\begin{aligned}
s_t &= \sigma_{t-1}^2 + \gamma^2 \sigma_t^2 + \boldsymbol{\Delta\phi}_t^\top \mathbf{P}_{t-1} \boldsymbol{\Delta\phi}_t + \frac{2\gamma\sigma_{t-1}^2}{s_{t-1}} \mathbf{p}_{t-1}^\top \boldsymbol{\Delta\phi}_t - \frac{\gamma^2 \sigma_{t-1}^4}{s_{t-1}} \\
&= \sigma_{t-1}^2 + \gamma^2 \sigma_t^2 + \left(\mathbf{p}_t + \frac{\gamma\sigma_{t-1}^2}{s_{t-1}} \mathbf{p}_{t-1}\right)^\top \left(\boldsymbol{\phi}(\mathbf{x}_{t-1}) - \gamma \boldsymbol{\phi}(\mathbf{x}_t)\right) - \frac{\gamma^2 \sigma_{t-1}^4}{s_{t-1}}.
\end{aligned}$$

Throughout the derivations above we repeatedly made use of $\mathbf{I} - \mathbf{P}_{t-1} = \boldsymbol{\Phi}_{t-1} \mathbf{H}_{t-1}^\top \mathbf{Q}_{t-1} \mathbf{H}_{t-1} \boldsymbol{\Phi}_{t-1}^\top$, $\mathbf{Q}_{t-1} \mathbf{e} = [-\boldsymbol{g}_{t-1}^\top, 1]^\top$, and the recursive expressions for $\mathbf{H}_{t-1}$, $\mathbf{Q}_{t-1}$, $\boldsymbol{\Phi}_{t-1}$ and $\mathbf{r}_{t-1}$.

### A.2 Nonparametric Monte-Carlo GPTD

Recall the expressions for the nonparametric posterior mean and variance (Eq. 3.10):

$$\hat{V}_t(\mathbf{x}) = \mathbf{k}_t(\mathbf{x})^\top \boldsymbol{\alpha}_t, \quad \text{and } P_t(\mathbf{x}, \mathbf{x}') = k(\mathbf{x}, \mathbf{x}') - \mathbf{k}_t(\mathbf{x})^\top \mathbf{C}_t \mathbf{k}_t(\mathbf{x}'),$$

respectively, with $\boldsymbol{\alpha}_t = \mathbf{H}_t^\top \mathbf{Q}_t \mathbf{r}_{t-1}$, $\mathbf{C}_t = \mathbf{H}_t^\top \mathbf{Q}_t \mathbf{H}_t$ and $\mathbf{Q}_t = \left(\mathbf{H}_t \mathbf{K}_t \mathbf{H}_t^\top + \boldsymbol{\Sigma}_t\right)^{-1}$. Let us write recursive expressions for $\mathbf{H}_t$, $\boldsymbol{\Sigma}_t$ and $\mathbf{Q}_t^{-1}$. Define $\mathbf{e} = (0, \ldots, 0, 1)^\top$ and $\mathbf{h}_t = (\mathbf{e}^\top, -\gamma)^\top = (0, \ldots, 1, -\gamma)^\top$, then,

$$\begin{aligned}
\mathbf{H}_t &= \begin{bmatrix} \mathbf{H}_{t-1}, \ \mathbf{0} \\ \hline \mathbf{h}_t^\top \end{bmatrix}, \quad \mathbf{K}_t = \begin{bmatrix} \mathbf{K}_{t-1} & \mathbf{k}_{t-1}(\mathbf{x}_t) \\ \mathbf{k}_{t-1}(\mathbf{x}_t)^\top & k(\mathbf{x}_t, \mathbf{x}_t) \end{bmatrix}, \\
\boldsymbol{\Sigma}_t &= \begin{bmatrix} \boldsymbol{\Sigma}_{t-1} & -\gamma\sigma_{t-1}^2 \mathbf{e} \\ -\gamma\sigma_{t-1}^2 \mathbf{e}^\top & \sigma_{t-1}^2 + \gamma^2 \sigma_t^2 \end{bmatrix}.
\end{aligned}$$

Substituting these into the expression for $\mathbf{Q}_t^{-1}$ we get,

$$\mathbf{Q}_t^{-1} = \mathbf{H}_t \mathbf{K}_t \mathbf{H}_t^\top + \boldsymbol{\Sigma}_t = \begin{bmatrix} \mathbf{Q}_{t-1}^{-1} & \mathbf{H}_{t-1} \boldsymbol{\Delta}\mathbf{k}_t - \gamma\sigma_{t-1}^2 \mathbf{e} \\ \left(\mathbf{H}_{t-1} \boldsymbol{\Delta}\mathbf{k}_t - \gamma\sigma_{t-1}^2 \mathbf{e}\right)^\top & \Delta k_{tt} + \sigma_{t-1}^2 + \gamma^2 \sigma_t^2 \end{bmatrix},$$

where we made use of the definitions

$$\begin{aligned}\boldsymbol{\Delta}\mathbf{k}_t &\stackrel{\text{def}}{=} \mathbf{k}_{t-1}(\mathbf{x}_{t-1}) - \gamma\mathbf{k}_{t-1}(\mathbf{x}_t),\\ \Delta k_{tt} &\stackrel{\text{def}}{=} k(\mathbf{x}_{t-1},\mathbf{x}_{t-1}) - 2\gamma k(\mathbf{x}_{t-1},\mathbf{x}_t) + \gamma^2 k(\mathbf{x}_t,\mathbf{x}_t).\end{aligned}$$

$\mathbf{Q}_t$ may now be obtained using the partitioned matrix inversion formula (Scharf, 1991):

$$\mathbf{Q}_t = \frac{1}{s_t}\begin{bmatrix} s_t\mathbf{Q}_{t-1} + \boldsymbol{g}_t\boldsymbol{g}_t^\top & -\boldsymbol{g}_t \\ -\boldsymbol{g}_t^\top & 1\end{bmatrix},$$

where $\boldsymbol{g}_t = \mathbf{Q}_{t-1}\left(\mathbf{H}_{t-1}\boldsymbol{\Delta}\mathbf{k}_t - \gamma\sigma_{t-1}^2\mathbf{e}\right)$, and $s_t = \sigma_{t-1}^2 + \gamma^2\sigma_t^2 + \Delta k_{tt} - \boldsymbol{g}_t^\top\left(\mathbf{H}_{t-1}\boldsymbol{\Delta}\mathbf{k}_t - \gamma\sigma_{t-1}^2\mathbf{e}\right)$. Let us define

$$\mathbf{c}_t = \mathbf{h}_t - \begin{pmatrix}\mathbf{H}_{t-1}^\top\boldsymbol{g}_t \\ 0\end{pmatrix}, \quad \text{and } d_t = r_{t-1} - \boldsymbol{g}_t^\top\mathbf{r}_{t-2}.$$

We are now ready to derive the recursions for $\mathbf{C}_t$ and $\boldsymbol{\alpha}_t$:

$$\begin{aligned}\mathbf{C}_t &= \mathbf{H}_t^\top\mathbf{Q}_t\mathbf{H}_t\\ &= \frac{1}{s_t}\left[\begin{array}{cc|c}\mathbf{H}_{t-1}^\top & & \mathbf{h}_t\\ \mathbf{0}^\top & & \end{array}\right]\left[\begin{array}{cc}s_t\mathbf{Q}_{t-1} + \boldsymbol{g}_t\boldsymbol{g}_t^\top & -\boldsymbol{g}_t\\ -\boldsymbol{g}_t^\top & 1\end{array}\right]\left[\begin{array}{cc}\mathbf{H}_{t-1} & \mathbf{0}\\ \hline \mathbf{h}_t^\top & \end{array}\right]\\ &= \begin{bmatrix}\mathbf{H}_{t-1}\mathbf{Q}_{t-1}\mathbf{H}_{t-1}^\top & \mathbf{0}\\ \mathbf{0}^\top & 0\end{bmatrix} + \frac{1}{s_t}\left(\begin{pmatrix}\mathbf{H}_{t-1}^\top\boldsymbol{g}_t\\ 0\end{pmatrix} - \mathbf{h}_t\right)\left(\begin{pmatrix}\mathbf{H}_{t-1}^\top\boldsymbol{g}_t\\ 0\end{pmatrix} - \mathbf{h}_t\right)^\top\\ &= \begin{bmatrix}\mathbf{C}_{t-1} & \mathbf{0}\\ \mathbf{0}^\top & 0\end{bmatrix} + \frac{1}{s_t}\mathbf{c}_t\mathbf{c}_t^\top.\end{aligned}$$

$$\begin{aligned}\boldsymbol{\alpha}_t &= \mathbf{H}_t^\top\mathbf{Q}_t\mathbf{r}_{t-1}\\ &= \frac{1}{s_t}\left[\begin{array}{cc|c}\mathbf{H}_{t-1}^\top & & \mathbf{h}_t\\ \mathbf{0}^\top & & \end{array}\right]\left[\begin{array}{cc}s_t\mathbf{Q}_{t-1} + \boldsymbol{g}_t\boldsymbol{g}_t^\top & -\boldsymbol{g}_t\\ -\boldsymbol{g}_t^\top & 1\end{array}\right]\begin{bmatrix}\mathbf{r}_{t-2}\\ r_{t-1}\end{bmatrix}\\ &= \begin{pmatrix}\boldsymbol{\alpha}_{t-1}\\ 0\end{pmatrix} + \frac{1}{s_t}\left(\begin{pmatrix}\mathbf{H}_{t-1}^\top\boldsymbol{g}_t\\ 0\end{pmatrix} - \mathbf{h}_t\right)\left(\boldsymbol{g}_t^\top\mathbf{r}_{t-2} - r_{t-1}\right)\\ &= \begin{pmatrix}\boldsymbol{\alpha}_{t-1}\\ 0\end{pmatrix} + \frac{\mathbf{c}_t}{s_t}d_t.\end{aligned}$$

We are not done quite yet, since $d_t$, $\mathbf{c}_t$ and $s_t$ are not explicitly given in terms of current-time-step quantities, as in the case of deterministic transitions. We show next that, similarly to the parametric updates derived above, $d_t$, $\mathbf{c}_t$ and $s_t$ may be recursively updated without requiring any additional bookkeeping, apart of maintaining the sequence of states seen so far. Let us begin with $d_t$:

$$\begin{aligned}d_t &= r_{t-1} - \boldsymbol{g}_t^\top\mathbf{r}_{t-2}\\ &= r_{t-1} - \left(\mathbf{H}_{t-1}\boldsymbol{\Delta}\mathbf{k}_t - \gamma\sigma_{t-1}^2\mathbf{e}\right)^\top\mathbf{Q}_{t-1}\mathbf{r}_{t-2}\\ &= r_{t-1} - \boldsymbol{\Delta}\mathbf{k}_t^\top\boldsymbol{\alpha}_{t-1} - \frac{\gamma\sigma_{t-1}^2}{s_{t-1}}\left(\boldsymbol{g}_{t-1}^\top\mathbf{r}_{t-3} - r_{t-2}\right)\\ &= \frac{\gamma\sigma_{t-1}^2}{s_{t-1}}d_{t-1} + r_{t-1} - \boldsymbol{\Delta}\mathbf{k}_t^\top\boldsymbol{\alpha}_{t-1}.\end{aligned}$$

Turning to $\mathbf{c}_t$, first notice that

$$\begin{aligned}
\mathbf{H}_t^\top \mathbf{Q}_t \mathbf{e} &= \frac{1}{s_t} \left[\begin{array}{c|c} \mathbf{H}_{t-1}^\top & \mathbf{h}_t \\ \mathbf{0}^\top & \end{array}\right] \begin{bmatrix} s_t \mathbf{Q}_{t-1} + \boldsymbol{g}_t \boldsymbol{g}_t^\top & -\boldsymbol{g}_t \\ -\boldsymbol{g}_t^\top & 1 \end{bmatrix} \mathbf{e} \\
&= \frac{1}{s_t} \left[\begin{array}{c|c} \mathbf{H}_{t-1}^\top & \mathbf{h}_t \\ \mathbf{0}^\top & \end{array}\right] \begin{pmatrix} -\boldsymbol{g}_t \\ 1 \end{pmatrix} \\
&= \frac{1}{s_t} \left( \mathbf{h}_t - \begin{pmatrix} \mathbf{H}_{t-1}^\top \boldsymbol{g}_t \\ 0 \end{pmatrix} \right) \\
&= \frac{\mathbf{c}_t}{s_t}.
\end{aligned}$$

Therefore,

$$\begin{aligned}
\mathbf{c}_t &= \mathbf{h}_t - \begin{pmatrix} \mathbf{H}_{t-1}^\top \boldsymbol{g}_t \\ 0 \end{pmatrix} \\
&= \mathbf{h}_t - \begin{pmatrix} \mathbf{H}_{t-1}^\top \mathbf{Q}_{t-1} \left( \mathbf{H}_{t-1} \boldsymbol{\Delta}\mathbf{k}_t - \gamma \sigma_{t-1}^2 \mathbf{e} \right) \\ 0 \end{pmatrix} \\
&= \mathbf{h}_t - \begin{pmatrix} \mathbf{C}_{t-1} \boldsymbol{\Delta}\mathbf{k}_t - \gamma \sigma_{t-1}^2 \mathbf{H}_{t-1}^\top \mathbf{Q}_{t-1} \mathbf{e} \\ 0 \end{pmatrix} \\
&= \mathbf{h}_t - \begin{pmatrix} \mathbf{C}_{t-1} \boldsymbol{\Delta}\mathbf{k}_t - \frac{\gamma \sigma_{t-1}^2}{s_{t-1}} \mathbf{c}_{t-1} \\ 0 \end{pmatrix} \\
&= \frac{\gamma \sigma_{t-1}^2}{s_{t-1}} \begin{pmatrix} \mathbf{c}_{t-1} \\ 0 \end{pmatrix} + \mathbf{h}_t - \begin{pmatrix} \mathbf{C}_{t-1} \boldsymbol{\Delta}\mathbf{k}_t \\ 0 \end{pmatrix}.
\end{aligned}$$

Finally, $s_t$:

$$\begin{aligned}
s_t &= \sigma_{t-1}^2 + \gamma^2 \sigma_t^2 + \Delta k_{tt} - \boldsymbol{g}_t^\top \left( \mathbf{H}_{t-1} \boldsymbol{\Delta}\mathbf{k}_t - \gamma \sigma_{t-1}^2 \mathbf{e} \right) \\
&= \sigma_{t-1}^2 + \gamma^2 \sigma_t^2 + \Delta k_{tt} - \left( \mathbf{H}_{t-1} \boldsymbol{\Delta}\mathbf{k}_t - \gamma \sigma_{t-1}^2 \mathbf{e} \right)^\top \mathbf{Q}_{t-1} \left( \mathbf{H}_{t-1} \boldsymbol{\Delta}\mathbf{k}_t - \gamma \sigma_{t-1}^2 \mathbf{e} \right) \\
&= \sigma_{t-1}^2 + \gamma^2 \sigma_t^2 + \Delta k_{tt} - \boldsymbol{\Delta}\mathbf{k}_t^\top \mathbf{C}_{t-1} \boldsymbol{\Delta}\mathbf{k}_t + 2\gamma \sigma_{t-1}^2 \mathbf{e}^\top \mathbf{Q}_{t-1} \mathbf{H}_{t-1} \boldsymbol{\Delta}\mathbf{k}_t - \frac{\gamma^2 \sigma_{t-1}^4}{s_{t-1}} \\
&= \sigma_{t-1}^2 + \gamma^2 \sigma_t^2 + \Delta k_{tt} - \boldsymbol{\Delta}\mathbf{k}_t^\top \mathbf{C}_{t-1} \boldsymbol{\Delta}\mathbf{k}_t + \frac{2\gamma \sigma_{t-1}^2}{s_{t-1}} \mathbf{c}_{t-1}^\top \boldsymbol{\Delta}\mathbf{k}_t - \frac{\gamma^2 \sigma_{t-1}^4}{s_{t-1}}.
\end{aligned}$$

### A.3 Sparse Nonparametric Monte-Carlo GPTD

At each time step the current sampled state $\mathbf{x}_t$ may either be left out of the dictionary, in which case $\mathcal{D}_t = \mathcal{D}_{t-1}$, or added to it, in which case $\mathcal{D}_t = \mathcal{D}_{t-1} \cup \{\mathbf{x}_t\}$, as determined by the sparsification criterion. Let us begin with the updates for the first case.

CASE 1: $\mathcal{D}_t = \mathcal{D}_{t-1}$

Since the dictionary remains unchanged, $\tilde{\mathbf{K}}_t = \tilde{\mathbf{K}}_{t-1}$. Defining $\tilde{\mathbf{h}}_t = \boldsymbol{a}_{t-1} - \gamma \boldsymbol{a}_t$ and $\mathbf{e} = [0, \ldots, 0, 1]^\top$, we have

$$\mathbf{A}_t = \begin{bmatrix} \mathbf{A}_{t-1} \\ \boldsymbol{a}_t^\top \end{bmatrix}, \; \tilde{\mathbf{H}}_t = \begin{bmatrix} \tilde{\mathbf{H}}_{t-1} \\ \tilde{\mathbf{h}}_t^\top \end{bmatrix}, \; \boldsymbol{\Sigma}_t = \begin{bmatrix} \boldsymbol{\Sigma}_{t-1} & -\gamma \sigma_{t-1}^2 \mathbf{e} \\ -\gamma \sigma_{t-1}^2 \mathbf{e}^\top & \sigma_{t-1}^2 + \gamma^2 \sigma_t^2 \end{bmatrix}.$$

Consequently, we may obtain a recursive formula for $\tilde{\mathbf{Q}}_t^{-1}$ (see Eq. 3.28):

$$\begin{aligned}\tilde{\mathbf{Q}}_t^{-1} &= \begin{bmatrix}\tilde{\mathbf{H}}_{t-1}\\ \tilde{\mathbf{h}}_t^\top\end{bmatrix}\tilde{\mathbf{K}}_{t-1}\left[\tilde{\mathbf{H}}_{t-1}, \quad \tilde{\mathbf{h}}_t\right] + \begin{bmatrix}\boldsymbol{\Sigma}_{t-1} & -\gamma\sigma_{t-1}^2\mathbf{e}\\ -\gamma\sigma_{t-1}^2\mathbf{e}^\top & \sigma_{t-1}^2+\gamma^2\sigma_t^2\end{bmatrix}\\ &= \begin{bmatrix}\tilde{\mathbf{Q}}_{t-1}^{-1} & \tilde{\mathbf{H}}_{t-1}\tilde{\boldsymbol{\Delta}\mathbf{k}}_t - \gamma\sigma_{t-1}^2\mathbf{e}\\ (\tilde{\mathbf{H}}_{t-1}\tilde{\boldsymbol{\Delta}\mathbf{k}}_t - \gamma\sigma_{t-1}^2\mathbf{e})^\top & , \Delta k_{tt}+\sigma_{t-1}^2+\gamma^2\sigma_t^2\end{bmatrix},\end{aligned}$$

where in the last equality we used the definition $\tilde{\boldsymbol{\Delta}\mathbf{k}}_t \stackrel{\text{def}}{=} \tilde{\mathbf{K}}_{t-1}\tilde{\mathbf{h}}_t = \tilde{\mathbf{k}}_{t-1}(\mathbf{x}_{t-1}) - \gamma\tilde{\mathbf{k}}_{t-1}(\mathbf{x}_t)$, and $\Delta k_{tt} = \tilde{\mathbf{h}}_t^\top\tilde{\boldsymbol{\Delta}\mathbf{k}}_t$. We may now invert $\tilde{\mathbf{Q}}_t^{-1}$ using the partitioned matrix inversion formula:

$$\tilde{\mathbf{Q}}_t = \frac{1}{s_t}\begin{bmatrix}s_t\tilde{\mathbf{Q}}_{t-1}+\tilde{\boldsymbol{g}}_t\tilde{\boldsymbol{g}}_t^\top & -\tilde{\boldsymbol{g}}_t\\ -\tilde{\boldsymbol{g}}_t^\top & 1\end{bmatrix},$$

where $\tilde{\boldsymbol{g}}_t = \tilde{\mathbf{Q}}_{t-1}\left(\tilde{\mathbf{H}}_{t-1}\tilde{\boldsymbol{\Delta}\mathbf{k}}_t - \gamma\sigma^2\mathbf{e}\right)$, and $s_t = (1+\gamma^2)\sigma^2 + \Delta k_{tt} - \tilde{\boldsymbol{g}}_t^\top\left(\tilde{\mathbf{H}}_{t-1}\tilde{\boldsymbol{\Delta}\mathbf{k}}_t - \gamma\sigma^2\mathbf{e}\right)$. Let us define $\tilde{\mathbf{c}}_t = \left(\tilde{\mathbf{h}}_t - \tilde{\mathbf{H}}_{t-1}\tilde{\boldsymbol{g}}_t\right)$. The update of the covariance parameter matrix $\tilde{\mathbf{C}}_t$ is:

$$\begin{aligned}\tilde{\mathbf{C}}_t &= \tilde{\mathbf{H}}_t^\top\tilde{\mathbf{Q}}\tilde{\mathbf{H}}_t\\ &= \frac{1}{s_t}\left[\tilde{\mathbf{H}}_{t-1}^\top, \quad \tilde{\mathbf{h}}_t\right]\begin{bmatrix}s_t\tilde{\mathbf{Q}}_{t-1}+\tilde{\boldsymbol{g}}_t\tilde{\boldsymbol{g}}_t^\top & -\tilde{\boldsymbol{g}}_t\\ -\tilde{\boldsymbol{g}}_t^\top & 1\end{bmatrix}\begin{bmatrix}\tilde{\mathbf{H}}_{t-1}\\ \tilde{\mathbf{h}}_t^\top\end{bmatrix}\\ &= \tilde{\mathbf{H}}_{t-1}^\top\tilde{\mathbf{Q}}_{t-1}\tilde{\mathbf{H}}_{t-1} + \frac{1}{s_t}\left[\tilde{\mathbf{H}}_{t-1}^\top \quad \tilde{\mathbf{h}}_t\right]\begin{pmatrix}\tilde{\boldsymbol{g}}_t\\ -1\end{pmatrix}\begin{pmatrix}\tilde{\boldsymbol{g}}_t^\top & -1\end{pmatrix}\begin{bmatrix}\tilde{\mathbf{H}}_{t-1}\\ \tilde{\mathbf{h}}_t^\top\end{bmatrix}\\ &= \tilde{\mathbf{C}}_{t-1} + \frac{1}{s_t}\left(\tilde{\mathbf{H}}_{t-1}\tilde{\boldsymbol{g}}_t - \tilde{\mathbf{h}}_t\right)\left(\tilde{\mathbf{H}}_{t-1}\tilde{\boldsymbol{g}}_t - \tilde{\mathbf{h}}_t\right)^\top\\ &= \tilde{\mathbf{C}}_{t-1} + \frac{1}{s_t}\tilde{\mathbf{c}}_t\tilde{\mathbf{c}}_t^\top.\end{aligned}$$

Next, we compute $\tilde{\boldsymbol{\alpha}}_t$:

$$\begin{aligned}\tilde{\boldsymbol{\alpha}}_t &= \tilde{\mathbf{H}}_t^\top\tilde{\mathbf{Q}}_t\mathbf{r}_{t-1}\\ &= \frac{1}{s_t}\left[\tilde{\mathbf{H}}_{t-1}^\top, \tilde{\mathbf{h}}_t\right]\begin{bmatrix}s_t\tilde{\mathbf{Q}}_{t-1}+\tilde{\boldsymbol{g}}_t\tilde{\boldsymbol{g}}_t^\top & , -\tilde{\boldsymbol{g}}_t\\ -\tilde{\boldsymbol{g}}_t^\top & , 1\end{bmatrix}\begin{pmatrix}\mathbf{r}_{t-2}\\ r_{t-1}\end{pmatrix}\\ &= \tilde{\boldsymbol{\alpha}}_{t-1} + \frac{1}{s_t}(\tilde{\mathbf{H}}_{t-1}^\top\tilde{\boldsymbol{g}}_t - \tilde{\mathbf{h}}_t)(\tilde{\boldsymbol{g}}_t^\top\mathbf{r}_{t-2} - r_{t-1})\\ &= \tilde{\boldsymbol{\alpha}}_{t-1} + \frac{\tilde{\mathbf{c}}_t}{s_t}d_t,\end{aligned}$$

where we have defined

$$d_t = r_{t-1} - \tilde{\boldsymbol{g}}_t^\top\mathbf{r}_{t-2}.$$

Note that $\tilde{\boldsymbol{g}}_t$ is a $(t-1)\times 1$ vector; we would therefore like to eliminate it from our updates, since we aim at keeping the memory and time requirements of each update independent of $t$. Due to the special form of the matrix $\boldsymbol{\Sigma}_t$ we can derive simple recursive formulas for $\tilde{\mathbf{c}}_t$,

$d_t$ and $s_t$, thus eliminating $\tilde{\boldsymbol{g}}_t$ from the updates. Let us begin with $\tilde{\mathbf{c}}_t$:

$$\begin{aligned}
\tilde{\mathbf{c}}_t &= \tilde{\mathbf{h}}_t - \tilde{\mathbf{H}}_{t-1}^\top \tilde{\boldsymbol{g}}_t \\
&= \tilde{\mathbf{h}}_t - \tilde{\mathbf{H}}_{t-1}^\top \tilde{\mathbf{Q}}_{t-1} \left( \tilde{\mathbf{H}}_{t-1} \tilde{\boldsymbol{\Delta}\mathbf{k}}_t - \gamma \sigma_{t-1}^2 \mathbf{e} \right) \\
&= \tilde{\mathbf{h}}_t - \tilde{\mathbf{C}}_{t-1} \tilde{\boldsymbol{\Delta}\mathbf{k}}_t + \gamma \sigma_{t-1}^2 \tilde{\mathbf{H}}_{t-1} \tilde{\mathbf{Q}}_{t-1} \mathbf{e} \\
&= \tilde{\mathbf{h}}_t - \tilde{\mathbf{C}}_{t-1} \tilde{\boldsymbol{\Delta}\mathbf{k}}_t + \frac{\gamma \sigma_{t-1}^2}{s_{t-1}} \left( \tilde{\mathbf{h}}_{t-1} - \tilde{\mathbf{H}}_{t-2} \tilde{\boldsymbol{g}}_{t-1} \right) \\
&= \frac{\gamma \sigma_{t-1}^2}{s_{t-1}} \tilde{\mathbf{c}}_{t-1} + \tilde{\mathbf{h}}_t - \tilde{\mathbf{C}}_{t-1} \tilde{\boldsymbol{\Delta}\mathbf{k}}_t .
\end{aligned}$$

Turning to $d_t$:

$$\begin{aligned}
d_t &= r_{t-1} - \tilde{\boldsymbol{g}}_t^\top \mathbf{r}_{t-2} \\
&= r_{t-1} - \left( \tilde{\mathbf{H}}_{t-1} \tilde{\boldsymbol{\Delta}\mathbf{k}}_t - \gamma \sigma_{t-1}^2 \mathbf{e} \right)^\top \tilde{\mathbf{Q}}_{t-1} \mathbf{r}_{t-2} \\
&= r_{t-1} - \tilde{\boldsymbol{\Delta}\mathbf{k}}_t^\top \tilde{\boldsymbol{\alpha}}_{t-1} - \frac{\gamma \sigma_{t-1}^2}{s_{t-1}} \left( \tilde{\boldsymbol{g}}_{t-1}^\top \mathbf{r}_{t-3} - r_{t-2} \right) \\
&= \frac{\gamma \sigma_{t-1}^2}{s_{t-1}} d_{t-1} + r_{t-1} - \tilde{\boldsymbol{\Delta}\mathbf{k}}_t^\top \tilde{\boldsymbol{\alpha}}_{t-1}
\end{aligned}$$

Finally, $s_t$:

$$\begin{aligned}
s_t &= \sigma_{t-1}^2 + \gamma^2 \sigma_t^2 + \Delta k_{tt} - \tilde{\boldsymbol{g}}_t^\top \left( \tilde{\mathbf{H}}_{t-1} \tilde{\boldsymbol{\Delta}\mathbf{k}}_t - \gamma \sigma_{t-1}^2 \mathbf{e} \right) \\
&= \sigma_{t-1}^2 + \gamma^2 \sigma_t^2 + \Delta k_{tt} - \left( \tilde{\mathbf{H}}_{t-1} \tilde{\boldsymbol{\Delta}\mathbf{k}}_t - \gamma \sigma_{t-1}^2 \mathbf{e} \right)^\top \tilde{\mathbf{Q}}_{t-1} \left( \tilde{\mathbf{H}}_{t-1} \tilde{\boldsymbol{\Delta}\mathbf{k}}_t - \gamma \sigma_{t-1}^2 \mathbf{e} \right) \\
&= \sigma_{t-1}^2 + \gamma^2 \sigma_t^2 + \Delta k_{tt} - \tilde{\boldsymbol{\Delta}\mathbf{k}}_t^\top \tilde{\mathbf{C}}_{t-1} \tilde{\boldsymbol{\Delta}\mathbf{k}}_t + 2\gamma \sigma_{t-1}^2 \mathbf{e}^\top \tilde{\mathbf{Q}}_{t-1} \tilde{\mathbf{H}}_{t-1} \tilde{\boldsymbol{\Delta}\mathbf{k}}_t - \frac{\gamma^2 \sigma_{t-1}^4}{s_{t-1}} \\
&= \sigma_{t-1}^2 + \gamma^2 \sigma_t^2 + \tilde{\mathbf{h}}_t^\top \tilde{\boldsymbol{\Delta}\mathbf{k}}_t - \tilde{\boldsymbol{\Delta}\mathbf{k}}_t^\top \tilde{\mathbf{C}}_{t-1} \tilde{\boldsymbol{\Delta}\mathbf{k}}_t + \frac{2\gamma \sigma_{t-1}^2}{s_{t-1}} \tilde{\mathbf{c}}_{t-1}^\top \tilde{\boldsymbol{\Delta}\mathbf{k}}_t - \frac{\gamma^2 \sigma_{t-1}^4}{s_{t-1}}
\end{aligned}$$

Recall that $\tilde{\mathbf{C}}_{t-1} \tilde{\boldsymbol{\Delta}\mathbf{k}}_t$ was computed for the $\tilde{\mathbf{c}}_t$. Given $\tilde{\mathbf{c}}_t$ we may substitute $\tilde{\mathbf{C}}_{t-1} \tilde{\boldsymbol{\Delta}\mathbf{k}}_t = \frac{\gamma \sigma^2}{s_{t-1}} \tilde{\mathbf{c}}_{t-1} - \tilde{\mathbf{c}}_t + \tilde{\mathbf{h}}_t$. This removes the only matrix-vector product from the update, resulting in

$$\begin{aligned}
s_t &= \sigma_{t-1}^2 + \gamma^2 \sigma_t^2 + \tilde{\mathbf{h}}_t^\top \tilde{\boldsymbol{\Delta}\mathbf{k}}_t \\
&\quad - \tilde{\boldsymbol{\Delta}\mathbf{k}}_t^\top \left( \frac{\gamma \sigma^2}{s_{t-1}} \tilde{\mathbf{c}}_{t-1} - \tilde{\mathbf{c}}_t + \tilde{\mathbf{h}}_t \right) + \frac{2\gamma \sigma^2}{s_{t-1}} \tilde{\mathbf{c}}_{t-1}^\top \tilde{\boldsymbol{\Delta}\mathbf{k}}_t - \frac{\gamma^2 \sigma_{t-1}^4}{s_{t-1}} \\
&= \sigma_{t-1}^2 + \gamma^2 \sigma_t^2 + \tilde{\boldsymbol{\Delta}\mathbf{k}}_t^\top \left( \tilde{\mathbf{c}}_t + \frac{\gamma \sigma^2}{s_{t-1}} \tilde{\mathbf{c}}_{t-1} \right) - \frac{\gamma^2 \sigma_{t-1}^4}{s_{t-1}} .
\end{aligned}$$

Case 2: $\mathcal{D}_t = \mathcal{D}_{t-1} \cup \{\mathbf{x}_t\}$

Here, $\tilde{\mathbf{K}}_t$ is given by Eq. 3.23, which we repeat here for clarity:

$$\tilde{\mathbf{K}}_t = \begin{bmatrix} \tilde{\mathbf{K}}_{t-1} & \tilde{\mathbf{k}}_{t-1}(\mathbf{x}_t) \\ \tilde{\mathbf{k}}_{t-1}(\mathbf{x}_t)^\top & k(\mathbf{x}_t, \mathbf{x}_t) \end{bmatrix}$$

Furthermore, $\boldsymbol{a}_t = (0, \ldots, 1)^\top$ since the last member added to the dictionary is $\mathbf{x}_t$ itself (and $\boldsymbol{\phi}(\mathbf{x}_t)$ is exactly representable by itself). Therefore

$$\mathbf{A}_t = \begin{bmatrix} \mathbf{A}_{t-1} & \mathbf{0} \\ \mathbf{0}^\top & 1 \end{bmatrix}, \quad \tilde{\mathbf{H}}_t = \begin{bmatrix} \tilde{\mathbf{H}}_{t-1} & \mathbf{0} \\ \boldsymbol{a}_{t-1}^\top & -\gamma \end{bmatrix} = \left[\begin{array}{cc} \tilde{\mathbf{H}}_{t-1} & \mathbf{0} \\ \hline \tilde{\mathbf{h}}_t^\top & \end{array}\right],$$

where we defined

$$\tilde{\mathbf{h}}_t = \begin{pmatrix} \boldsymbol{a}_{t-1} \\ 0 \end{pmatrix} - \gamma \boldsymbol{a}_t = \begin{pmatrix} \boldsymbol{a}_{t-1} \\ -\gamma \end{pmatrix}.$$

The recursion for $\tilde{\mathbf{Q}}_t^{-1}$ is given by:

$$\begin{aligned} \tilde{\mathbf{Q}}_t^{-1} &= \left[\begin{array}{cc} \tilde{\mathbf{H}}_{t-1} & \mathbf{0} \\ \hline \tilde{\mathbf{h}}_t^\top & \end{array}\right] \begin{bmatrix} \tilde{\mathbf{K}}_{t-1} & \tilde{\mathbf{k}}_{t-1}(\mathbf{x}_t) \\ \tilde{\mathbf{k}}_{t-1}(\mathbf{x}_t)^\top & k(\mathbf{x}_t, \mathbf{x}_t) \end{bmatrix} \left[\begin{array}{cc|c} \tilde{\mathbf{H}}_{t-1}^\top & \mathbf{0}^\top & \tilde{\mathbf{h}}_t \\ & & \end{array}\right] + \begin{bmatrix} \boldsymbol{\Sigma}_{t-1} & -\gamma\sigma_{t-1}^2 \mathbf{e} \\ -\gamma\sigma_{t-1}^2 \mathbf{e}^\top & \sigma_{t-1}^2 + \gamma^2 \sigma_t^2 \end{bmatrix} \\ &= \begin{bmatrix} \tilde{\mathbf{Q}}_{t-1}^{-1} & \tilde{\mathbf{H}}_{t-1} \boldsymbol{\Delta}\tilde{\mathbf{k}}_t \\ (\tilde{\mathbf{H}}_{t-1} \boldsymbol{\Delta}\tilde{\mathbf{k}}_t)^\top & \Delta k_{tt} + \sigma_{t-1}^2 + \gamma^2 \sigma_t^2 \end{bmatrix}, \end{aligned}$$

where we defined

$$\begin{aligned} \boldsymbol{\Delta}\tilde{\mathbf{k}}_t &\stackrel{\text{def}}{=} \tilde{\mathbf{K}}_{t-1} \boldsymbol{a}_{t-1} - \gamma \tilde{\mathbf{k}}_{t-1}(\mathbf{x}_t) = \tilde{\mathbf{k}}_{t-1}(\mathbf{x}_{t-1}) - \gamma \tilde{\mathbf{k}}_{t-1}(\mathbf{x}_t), \\ \Delta k_{tt} &\stackrel{\text{def}}{=} \boldsymbol{a}_{t-1}^\top \tilde{\mathbf{K}}_{t-1} \boldsymbol{a}_{t-1} - 2\gamma \boldsymbol{a}_{t-1}^\top \tilde{\mathbf{k}}_{t-1}(\mathbf{x}_t) + \gamma k(\mathbf{x}_t, \mathbf{x}_t) \\ &= \boldsymbol{a}_{t-1}^\top \left( \tilde{\mathbf{k}}_{t-1}(\mathbf{x}_{t-1}) - 2\gamma \tilde{\mathbf{k}}_{t-1}(\mathbf{x}_t) \right) + \gamma^2 k(\mathbf{x}_t, \mathbf{x}_t). \end{aligned}$$

Defining, as before, $\boldsymbol{g}_t = \tilde{\mathbf{Q}}_{t-1} \left( \tilde{\mathbf{H}}_{t-1} \boldsymbol{\Delta}\tilde{\mathbf{k}}_t - \gamma\sigma^2 \mathbf{e} \right)$ and $s_t = \sigma_{t-1}^2 + \gamma^2 \sigma_t^2 + \Delta k_{tt} - \boldsymbol{g}_t^\top \left( \tilde{\mathbf{H}}_{t-1} \boldsymbol{\Delta}\tilde{\mathbf{k}}_t - \gamma\sigma^2 \mathbf{e} \right)$, and using again the partitioned matrix inversion formula we get

$$\tilde{\mathbf{Q}}_t = \frac{1}{s_t} \left[\begin{array}{cc} s_t \tilde{\mathbf{Q}}_{t-1} + \tilde{\boldsymbol{g}}_t \tilde{\boldsymbol{g}}_t^\top & -\tilde{\boldsymbol{g}}_t \\ -\tilde{\boldsymbol{g}}_t^\top & 1 \end{array}\right].$$

Let us define

$$\tilde{\mathbf{c}}_t = \tilde{\mathbf{h}}_t - \begin{pmatrix} \tilde{\mathbf{H}}_{t-1}^\top \tilde{\boldsymbol{g}}_t \\ 0 \end{pmatrix}, \quad \text{and } d_t = r_{t-1} - \boldsymbol{g}_t^\top \mathbf{r}_{t-2}.$$

We are now ready to compute $\tilde{\mathbf{C}}_t$ and $\tilde{\boldsymbol{\alpha}}_t$:

$$\begin{aligned} \tilde{\mathbf{C}}_t &= \tilde{\mathbf{H}}_t^\top \tilde{\mathbf{Q}}_t \tilde{\mathbf{H}}_t \\ &= \frac{1}{s_t} \left[\begin{array}{cc|c} \tilde{\mathbf{H}}_{t-1}^\top & & \tilde{\mathbf{h}}_t \\ \mathbf{0}^\top & & \end{array}\right] \left[\begin{array}{cc} s_t \tilde{\mathbf{Q}}_{t-1} + \tilde{\boldsymbol{g}}_t \tilde{\boldsymbol{g}}_t^\top & -\tilde{\boldsymbol{g}}_t \\ -\tilde{\boldsymbol{g}}_t^\top & 1 \end{array}\right] \left[\begin{array}{cc} \tilde{\mathbf{H}}_{t-1} & \mathbf{0} \\ \hline \tilde{\mathbf{h}}_t^\top & \end{array}\right] \\ &= \begin{bmatrix} \tilde{\mathbf{H}}_{t-1} \tilde{\mathbf{Q}}_{t-1} \tilde{\mathbf{H}}_{t-1}^\top & \mathbf{0} \\ \mathbf{0}^\top & 0 \end{bmatrix} + \frac{1}{s_t} \left( \begin{pmatrix} \tilde{\mathbf{H}}_{t-1}^\top \tilde{\boldsymbol{g}}_t \\ 0 \end{pmatrix} - \tilde{\mathbf{h}}_t \right) \left( \begin{pmatrix} \tilde{\mathbf{H}}_{t-1}^\top \tilde{\boldsymbol{g}}_t \\ 0 \end{pmatrix} - \tilde{\mathbf{h}}_t \right)^\top \\ &= \begin{bmatrix} \tilde{\mathbf{C}}_{t-1} & \mathbf{0} \\ \mathbf{0}^\top & 0 \end{bmatrix} + \frac{1}{s_t} \tilde{\mathbf{c}}_t \tilde{\mathbf{c}}_t^\top. \end{aligned}$$

$$
\begin{aligned}
\tilde{\boldsymbol{\alpha}}_t &= \tilde{\mathbf{H}}_t^\top \tilde{\mathbf{Q}}_t \mathbf{r}_{t-1} \\
&= \frac{1}{s_t} \left[\begin{array}{c|c} \tilde{\mathbf{H}}_{t-1}^\top & \tilde{\mathbf{h}}_t \\ \mathbf{0}^\top & \end{array}\right] \begin{bmatrix} s_t \tilde{\mathbf{Q}}_{t-1} + \tilde{\boldsymbol{g}}_t \tilde{\boldsymbol{g}}_t^\top & -\tilde{\boldsymbol{g}}_t \\ -\tilde{\boldsymbol{g}}_t^\top & 1 \end{bmatrix} \begin{bmatrix} \mathbf{r}_{t-2} \\ r_{t-1} \end{bmatrix} \\
&= \begin{pmatrix} \tilde{\boldsymbol{\alpha}}_{t-1} \\ 0 \end{pmatrix} + \frac{1}{s_t} \left( \begin{pmatrix} \tilde{\mathbf{H}}_{t-1}^\top \tilde{\boldsymbol{g}}_t \\ 0 \end{pmatrix} - \tilde{\mathbf{h}}_t \right) \left( \boldsymbol{g}_t^\top \mathbf{r}_{t-2} - r_{t-1} \right) \\
&= \begin{pmatrix} \tilde{\boldsymbol{\alpha}}_{t-1} \\ 0 \end{pmatrix} + \frac{\tilde{\mathbf{c}}_t}{s_t} d_t.
\end{aligned}
$$

As above, we still need to derive recursions for $d_t$, $\tilde{\mathbf{c}}_t$ and $s_t$. Let us begin with $d_t$:

$$
\begin{aligned}
d_t &= r_{t-1} - \boldsymbol{g}_t^\top \mathbf{r}_{t-2} \\
&= r_{t-1} - \left( \tilde{\mathbf{H}}_{t-1} \tilde{\boldsymbol{\Delta}\mathbf{k}}_t - \gamma \sigma_{t-1}^2 \mathbf{e} \right)^\top \mathbf{Q}_{t-1} \mathbf{r}_{t-2} \\
&= r_{t-1} - \tilde{\boldsymbol{\Delta}\mathbf{k}}_t^\top \tilde{\boldsymbol{\alpha}}_{t-1} - \frac{\gamma \sigma_{t-1}^2}{s_{t-1}} \left( \boldsymbol{g}_{t-1}^\top \mathbf{r}_{t-3} - r_{t-2} \right) \\
&= \frac{\gamma \sigma_{t-1}^2}{s_{t-1}} d_{t-1} + r_{t-1} - \tilde{\boldsymbol{\Delta}\mathbf{k}}_t^\top \tilde{\boldsymbol{\alpha}}_{t-1}.
\end{aligned}
$$

Turning to $\tilde{\mathbf{c}}_t$, first notice that

$$
\begin{aligned}
\tilde{\mathbf{H}}_t^\top \mathbf{Q}_t \mathbf{e} &= \frac{1}{s_t} \left[\begin{array}{c|c} \tilde{\mathbf{H}}_{t-1}^\top & \tilde{\mathbf{h}}_t \\ \mathbf{0}^\top & \end{array}\right] \begin{bmatrix} s_t \mathbf{Q}_{t-1} + \boldsymbol{g}_t \boldsymbol{g}_t^\top & -\boldsymbol{g}_t \\ -\boldsymbol{g}_t^\top & 1 \end{bmatrix} \mathbf{e} \\
&= \frac{1}{s_t} \left[\begin{array}{c|c} \tilde{\mathbf{H}}_{t-1}^\top & \tilde{\mathbf{h}}_t \\ \mathbf{0}^\top & \end{array}\right] \begin{pmatrix} -\boldsymbol{g}_t \\ 1 \end{pmatrix} \\
&= \frac{1}{s_t} \left( \tilde{\mathbf{h}}_t - \begin{pmatrix} \tilde{\mathbf{H}}_{t-1}^\top \boldsymbol{g}_t \\ 0 \end{pmatrix} \right) \\
&= \frac{\tilde{\mathbf{c}}_t}{s_t}.
\end{aligned}
$$

Therefore,

$$
\begin{aligned}
\tilde{\mathbf{c}}_t &= \tilde{\mathbf{h}}_t - \begin{pmatrix} \tilde{\mathbf{H}}_{t-1}^\top \boldsymbol{g}_t \\ 0 \end{pmatrix} \\
&= \tilde{\mathbf{h}}_t - \begin{pmatrix} \tilde{\mathbf{H}}_{t-1}^\top \mathbf{Q}_{t-1} \left( \tilde{\mathbf{H}}_{t-1} \tilde{\boldsymbol{\Delta}\mathbf{k}}_t - \gamma \sigma_{t-1}^2 \mathbf{e} \right) \\ 0 \end{pmatrix} \\
&= \tilde{\mathbf{h}}_t - \begin{pmatrix} \tilde{\mathbf{C}}_{t-1} \tilde{\boldsymbol{\Delta}\mathbf{k}}_t - \gamma \sigma_{t-1}^2 \tilde{\mathbf{H}}_{t-1}^\top \mathbf{Q}_{t-1} \mathbf{e} \\ 0 \end{pmatrix} \\
&= \tilde{\mathbf{h}}_t - \begin{pmatrix} \tilde{\mathbf{C}}_{t-1} \tilde{\boldsymbol{\Delta}\mathbf{k}}_t - \frac{\gamma \sigma_{t-1}^2}{s_{t-1}} \tilde{\mathbf{c}}_{t-1} \\ 0 \end{pmatrix} \\
&= \frac{\gamma \sigma_{t-1}^2}{s_{t-1}} \begin{pmatrix} \tilde{\mathbf{c}}_{t-1} \\ 0 \end{pmatrix} + \tilde{\mathbf{h}}_t - \begin{pmatrix} \tilde{\mathbf{C}}_{t-1} \tilde{\boldsymbol{\Delta}\mathbf{k}}_t \\ 0 \end{pmatrix}.
\end{aligned}
$$

Finally, $s_t$:

$$\begin{aligned}
s_t &= \sigma_{t-1}^2 + \gamma^2\sigma_t^2 + \Delta k_{tt} - \boldsymbol{g}_t^\top \left(\tilde{\mathbf{H}}_{t-1}\boldsymbol{\Delta}\tilde{\mathbf{k}}_t - \gamma\sigma_{t-1}^2\mathbf{e}\right) \\
&= \sigma_{t-1}^2 + \gamma^2\sigma_t^2 + \Delta k_{tt} - \left(\tilde{\mathbf{H}}_{t-1}\boldsymbol{\Delta}\tilde{\mathbf{k}}_t - \gamma\sigma_{t-1}^2\mathbf{e}\right)^\top \mathbf{Q}_{t-1}\left(\tilde{\mathbf{H}}_{t-1}\boldsymbol{\Delta}\tilde{\mathbf{k}}_t - \gamma\sigma_{t-1}^2\mathbf{e}\right) \\
&= \sigma_{t-1}^2 + \gamma^2\sigma_t^2 + \Delta k_{tt} - \boldsymbol{\Delta}\tilde{\mathbf{k}}_t^\top\tilde{\mathbf{C}}_{t-1}\boldsymbol{\Delta}\tilde{\mathbf{k}}_t + 2\gamma\sigma_{t-1}^2\mathbf{e}^\top\mathbf{Q}_{t-1}\tilde{\mathbf{H}}_{t-1}\boldsymbol{\Delta}\tilde{\mathbf{k}}_t - \frac{\gamma^2\sigma_{t-1}^4}{s_{t-1}} \\
&= \sigma_{t-1}^2 + \gamma^2\sigma_t^2 + \Delta k_{tt} - \boldsymbol{\Delta}\tilde{\mathbf{k}}_t^\top\tilde{\mathbf{C}}_{t-1}\boldsymbol{\Delta}\tilde{\mathbf{k}}_t + \frac{2\gamma\sigma_{t-1}^2}{s_{t-1}}\tilde{\mathbf{c}}_{t-1}^\top\boldsymbol{\Delta}\tilde{\mathbf{k}}_t - \frac{\gamma^2\sigma_{t-1}^4}{s_{t-1}}.
\end{aligned}$$

## Appendix B. Mathematical Formulae

The formulae noted in this appendix may be found in Scharf (1991), as well as most introductory textbooks on probability and statistics. The only exceptions are the conditional covariance formula, and what we call the "Transparency Lemma". The first is well known and may be easily derived from the definition of the covariance of a random vector, using the conditional expectation formula. A proof of the latter we were not able to locate elsewhere, and we therefore state and prove it in Section B.5. In the sequel read $\mathbb{P}$ as the probability density function of its argument, unless that argument is countable, in which case $\mathbb{P}$ denotes probability and $\int$ denotes a sum.

### B.1 Bayes' Rule

Let $X$ and $Y$ be random variables or vectors, then

$$\mathbb{P}(X|Y) = \frac{\mathbb{P}(Y|X)\mathbb{P}(X)}{\mathbb{P}(Y)} = \frac{\mathbb{P}(Y|X)\mathbb{P}(X)}{\int dX'\mathbb{P}(Y|X')\mathbb{P}(X')}.$$

### B.2 The Multivariate Normal Distribution

Let $X$ be an $n$ dimensional random vector. It is said that $X$ is normally distributed (or alternatively, that its components are jointly normally distributed) as $\mathcal{N}\{\mathbf{m}, \boldsymbol{\Sigma}\}$ if its probability density function satisfies

$$\mathbb{P}(X) = (2\pi)^{-n/2}|\boldsymbol{\Sigma}|^{-1/2}\exp\left(-\frac{1}{2}(X-\mathbf{m})^\top\boldsymbol{\Sigma}^{-1}(X-\mathbf{m})\right).$$

Conditioning (Gauss-Markov Theorem)

Let $X$ and $Y$ be random vectors that are distributed according to the multivariate normal distribution

$$\begin{pmatrix} X \\ Y \end{pmatrix} \sim \mathcal{N}\left\{\begin{pmatrix} \mathbf{m}_x \\ \mathbf{m}_y \end{pmatrix}, \begin{bmatrix} \mathbf{K}_{xx} & \mathbf{K}_{xy} \\ \mathbf{K}_{yx} & \mathbf{K}_{yy} \end{bmatrix}\right\}$$

Then $X|Y \sim \mathcal{N}\left\{\hat{X}, \mathbf{P}\right\}$, where

$$\begin{aligned}
\hat{X} &= \mathbf{m}_x + \mathbf{K}_{xy}\mathbf{K}_{yy}^{-1}(Y - \mathbf{m}_y) \\
\mathbf{P} &= \mathbf{K}_{xx} - \mathbf{K}_{xy}\mathbf{K}_{yy}^{-1}\mathbf{K}_{yx}.
\end{aligned}$$

$\mathbf{P}$ is also known as the *Schur complement* of $\mathbf{K}_{xx}$ in the partitioned matrix $\begin{bmatrix}\mathbf{K}_{xx} & \mathbf{K}_{xy}\\ \mathbf{K}_{yx} & \mathbf{K}_{yy}\end{bmatrix}$.

### B.3 Conditional Expectation and Covariance Formulae

Let $X$, $Y$, $Z$ be random variables or vectors, then

$$\begin{aligned}\mathbf{E}[X] &= \mathbf{E}_Y\left[\mathbf{E}[X|Y]\right],\\ \mathbf{Cov}[X,Z] &= \mathbf{E}_Y\left[\mathbf{Cov}[X,Z|Y]\right] + \mathbf{Cov}_Y\left[\mathbf{E}[X|Y],\mathbf{E}[Z|Y]\right].\end{aligned}$$

Specializing the last equation to X=Z scalars, we get the conditional variance formula:

$$\mathbf{Var}[X] = \mathbf{E}_Y\left[\mathbf{Var}[X|Y]\right] + \mathbf{Var}_Y\left[\mathbf{E}[X|Y]\right].$$

### B.4 Matrix Inversion Formulae

Matrix Inversion Lemma (Sherman-Morrison-Woodbury)

Let $\mathbf{U}$ be a square invertible matrix, then

$$(\mathbf{U}+\beta\mathbf{X}\mathbf{Y})^{-1} = \mathbf{U}^{-1} - \mathbf{U}^{-1}\mathbf{X}\left(\beta\mathbf{I}+\mathbf{Y}\mathbf{U}^{-1}\mathbf{X}\right)^{-1}\mathbf{Y}\mathbf{U}^{-1}.$$

Partitioned Covariance Matrix Inverse Formula

Let $\mathbf{K}_t$ be a $t\times t$ (symmetric and positive-definite) covariance matrix, partitioned as

$$\mathbf{K}_t = \begin{bmatrix}\mathbf{K}_{t-1} & \mathbf{k}_t\\ \mathbf{k}_t^\top & k_{tt}\end{bmatrix}.$$

Then

$$\mathbf{K}_t^{-1} = \begin{bmatrix}\mathbf{K}_{t-1}^{-1} & \mathbf{0}\\ \mathbf{0}^\top & 0\end{bmatrix} + \frac{1}{s_t}\begin{bmatrix}\mathbf{K}_{t-1}^{-1}\mathbf{k}_t\\ -1\end{bmatrix}\begin{bmatrix}\mathbf{k}_t^\top\mathbf{K}_{t-1}^{-1} & -1\end{bmatrix},$$

where $s_t = k_{tt} - \mathbf{k}_t^\top\mathbf{K}_{t-1}^{-1}\mathbf{k}_t$. $s_t$ is called the *Schur complement* of $k_{tt}$ in $\mathbf{K}_t$. If $X = (X_1,\ldots,X_t)^\top$ is a jointly Gaussian random vector, the self-covariance of which satisfies $\mathbf{Cov}[X] = \mathbf{K}_t$, then by the Gauss-Markov theorem (see Sec. B.2), $s_t$ satisfies

$$s_t = \mathbf{Var}[X_t|X_1,\ldots,X_{t-1}].$$

### B.5 Transparency Lemma

Let $\mathbf{A}$ and $\mathbf{B}$ be matrices of dimensions $n\times m$ and $m\times n$, respectively, and let $\mathbf{B}\mathbf{A}+\mathbf{I}$ be nonsingular, then $\mathbf{A}\mathbf{B}+\mathbf{I}$ is also nonsingular, and

$$\mathbf{A}\left(\mathbf{B}\mathbf{A}+\mathbf{I}\right)^{-1} = \left(\mathbf{A}\mathbf{B}+\mathbf{I}\right)^{-1}\mathbf{A}.$$

Note that $\mathbf{I}$ on the l.h.s. is the $m\times m$ identity matrix, while on the r.h.s. it is the $n\times n$ identity matrix[20].

20. The name "Transparency" notes the fact that this formula behaves as if the inverse sign were absent.

**Proof** Assume to the contrary that $\mathbf{AB} + \mathbf{I}$ is singular, then there exists a vector $\mathbf{y} \neq \mathbf{0}$ such that

$$(\mathbf{AB} + \mathbf{I})\mathbf{y} = \mathbf{0}. \tag{B.1}$$

Premultiply this by $\mathbf{B}$ to obtain

$$\mathbf{B}(\mathbf{AB} + \mathbf{I})\mathbf{y} = (\mathbf{BA} + \mathbf{I})\mathbf{By} = \mathbf{0}.$$

Now, $\mathbf{By} \neq \mathbf{0}$, for otherwise we would have from Eq. B.1 that $\mathbf{y} = \mathbf{0}$. Therefore, for $\mathbf{x} \stackrel{\text{def}}{=} \mathbf{By} \neq \mathbf{0}$ we have

$$(\mathbf{BA} + \mathbf{I})\mathbf{x} = \mathbf{0},$$

which means that $(\mathbf{BA} + \mathbf{I})$ is singular, resulting in a contradiction, thus proving the first part of the Lemma.

The second part is proved as follows:

$$\begin{aligned}
\mathbf{A}\left(\mathbf{BA} + \mathbf{I}\right)^{-1} &= (\mathbf{AB} + \mathbf{I})^{-1}(\mathbf{AB} + \mathbf{I})\mathbf{A}\left(\mathbf{BA} + \mathbf{I}\right)^{-1} \\
&= (\mathbf{AB} + \mathbf{I})^{-1}\mathbf{A}(\mathbf{BA} + \mathbf{I})(\mathbf{BA} + \mathbf{I})^{-1} \\
&= (\mathbf{AB} + \mathbf{I})^{-1}\mathbf{A}.
\end{aligned}$$

## References


R.E. Bellman. A problem in the sequential design of experiments. *Sankhya*, 16, 1956.

R.E. Bellman. *Dynamic Programming*. Princeton University Press, 1957.

D.P. Bertsekas. *Dynamic Programming and Optimal Control*. Athena Scientific, 1995.

D.P. Bertsekas and J.N. Tsitsiklis. *Neuro-Dynamic Programming*. Athena Scientific, 1996.

J.A. Boyan. Least-squares temporal difference learning. In *Proc. 16th International Conference on Machine Learning*, pages 49–56. Morgan Kaufmann, San Francisco, CA, 1999.

J.A. Boyan and A.W. Moore. Generalization in reinforcement learning: Safely approximating the value function. In G. Tesauro, D. S. Touretzky, and T. K. Leen, editors, *Advances in Neural Information Processing Systems 7*, pages 369–376, Cambridge, MA, 1995. MIT Press.

S.J. Bradtke and A.G. Barto. Linear least-squares algorithms for temporal difference learning. *Machine Learning*, 22:33–57, 1996.

C.J.C. Burges. Simplified support vector decision rules. In *International Conference on Machine Learning*, pages 71–77, 1996.

R.H. Crites and A.G. Barto. Improving elevator performance using reinforcement learning. In David S. Touretzky, Michael C. Mozer, and Michael E. Hasselmo, editors, *Advances in Neural Information Processing Systems*, volume 8, pages 1017–1023. MIT Press, 1996.

L. Csató and M. Opper. Sparse on-line Gaussian processes. *Neural Computation*, 14:641–668, 2002.

R. Dearden, N. Friedman, and D. Andre. Model based Bayesian exploration. In *Proc. of the Fifteenth Conf. on Uncertainty in Artificial Intelligence*, pages 150–159, 1999.

R. Dearden, N. Friedman, and S. Russell. Bayesian Q-learning. In *Proc. of the Fifteenth National Conference on Artificial Intelligence*, 1998.

M. Duff. *Optimal Learning: Computational procedures for Bayes-adaptive Markov decision processes.* PhD thesis, University of Massassachusetts Amherst, 2002.

Y. Engel. *Algorithms and Representations for Reinforcement Learning.* PhD thesis, The Hebrew University of Jerusalem, 2005. URL `www.cs.ualberta.ca/∼yaki/papers/thesis.ps`.

Y. Engel, S. Mannor, and R. Meir. Sparse online greedy support vector regression. In *13th European Conference on Machine Learning*, 2002.

Y. Engel, S. Mannor, and R. Meir. Bayes meets Bellman: The Gaussian process approach to temporal difference learning. In *Proc. of the 20th International Conference on Machine Learning*, 2003.

Y. Engel, S. Mannor, and R. Meir. Reinforcement learning with Gaussian processes. In *Proc. of the 22nd International Conference on Machine Learning*, 2005.

E. Even-Dar, S. Mannor, and Y. Mansour. Action elimination and stopping conditions for reinforcement learning. In *Proc. of the 20th International Conference on Machine Learning*, 2003.

M. Gibbs and D. MacKay. Efficient implementation of Gaussian processes. Technical report, Department of Physics, Cavendish Laboratory, Cambridge University, 1997.

G.J. Gordon. Stable fitted reinforcement learning. In David S. Touretzky, Michael C. Mozer, and Michael E. Hasselmo, editors, *Advances in Neural Information Processing Systems*, volume 8, pages 1052–1058. MIT Press, 1996.

R. Howard. *Dynamic Programming and Markov Processes.* MIT Press, Cambridge, MA, 1960.

Leslie Pack Kaelbling. *Learning in embedded systems.* MIT Press, 1993. ISBN 0-262-11174-8.

L.P. Kaelbling, M.L. Littman, , and A.R. Cassandra. Planning and acting in partially observable stochastic domains. *Artificial Intelligence*, 101:99–134, 1998.

V. Konda and J. Tsitsiklis. Actor-critic algorithms. In *Advances in Neural Information Processing Systems 12*, pages 1008–1014, 2000.

M.G. Lagoudakis, R. Parr, and M.L. Littman. Least-squares methods in reinforcement learning for control. In *SETN*, pages 249–260, 2002.

Michael L. Littman. Markov games as a framework for multi-agent reinforcement learning. In *Proceedings of the 11th International Conference on Machine Learning (ICML-94)*, pages 157–163, New Brunswick, NJ, 1994. Morgan Kaufmann. URL `citeseer.ist.psu.edu/littman94markov.html`.

D.J.C. Mackay. Gaussian processes: A replacement for supervised neural networks? Technical report, Cavendish Laboratory, Cambridge University, 1997. URL `http://citeseer.ist.psu.edu/mackay97gaussian.html`.

Dörthe Malzahn and Manfred Opper. Learning curves for gaussian processes regression: A framework for good approximations. In Todd K. Leen, Thomas G. Dietterich, and Volker Tresp, editors, *Advances in Neural Information Processing Systems 13*, pages 273–279. MIT Press, 2001.

S. Mannor, D. Simester, P. Sun, and J.N. Tsitsiklis. Bias and variance in value function estimation. In *Proc. of the 21st International Conference on Machine Learning*, 2004.

R. Munos. A study of reinforcement learning in the continuous case by the means of viscosity solutions. *Machine Learning*, 40(3):265–299, 2000.

Rémi Munos. Error bounds for approximate policy iteration. In *Machine Learning, Proceedings of the Twentieth International Conference (ICML 2003), August 21-24, 2003, Washington, DC, USA*, pages 560–567. AAAI Press, 2003.

A. Nedic and D. P. Bertsekas. Least squares policy evaluation algorithms with linear function approximation. *J. of Discrete Event Systems*, 13:79–110, 2003. URL `citeseer.nj.nec.com/nedic02least.html`.

A. O'Hagan. Curve fitting and optimal design for regression. *Journal of the Royal Statistical Society B*, 40:1–42, 1978.

Manfred Opper and Francesco Vivarelli. General bounds on bayes errors for regression with gaussian processes. In *Advances in Neural Information Processing Systems 11*, pages 302–308, 1999.

P. Poupart, N.A. Vlassis, J. Hoey, and K. Regan. An analytic solution to discrete Bayesian reinforcement learning. In *Proceedings of the Twenty-Third International Conference on Machine Learning,*, pages 697–704, 2006.

M.L. Puterman. *Markov Decision Processes: Discrete Stochastic Dynamic Programming*. John Wiley & Sons, 1994.

C.E. Rasmussen and M. Kuss. Gaussian processes in reinforcement learning. In Sebastian Thrun, Lawrence Saul, and Bernhard Schölkopf, editors, *Advances in Neural Information Processing Systems 16*, Cambridge, MA, 2004. MIT Press.

C.E. Rasmussen and C.K.I. Williams. *Gaussian Processes for Machine Learning*. MIT Press, Cambridge, Massachusetts, 2006.

G. Rummery and M. Niranjan. On-line Q-learning using connectionist systems. Technical Report CUED/F-INFENG/TR 166, Cambridge University Engineering Department, 1994. URL `overcite.lcs.mit.edu/rummery94line.html`.

L.L. Scharf. *Statistical Signal Processing*. Addison-Wesley, 1991.

B. Schölkopf and A. Smola. *Learning with Kernels*. MIT Press, Cambridge, MA, 2002.

N.N. Schraudolph, P. Dayan, and T.J. Sejnowski. Temporal difference learning of position evaluation in the game of Go. In Jack D. Cowan, Gerald Tesauro, and Joshua Alspector, editors, *Advances in Neural Information Processing Systems*, volume 6, pages 817–824. Morgan Kaufmann Publishers, Inc., 1994.

M. Seeger. *Bayesian Gaussian Process Models: PAC-Bayesian Generalisation Error Bounds and Sparse Approximations*. PhD thesis, University of Edinburgh, 2003.

M. Seeger. Gaussian processes for machine learning. *International Journal of Neural Systems*, 14(2): 69–106, 2004. URL `citeseer.ist.psu.edu/seeger04gaussian.html`.

J. Shawe-Taylor and N. Cristianini. *Kernel Methods for Pattern Analysis*. Cambridge University Press, Cambridge, England, 2004.

S.P. Singh, T. Jaakkola, and M.I. Jordan. Reinforcement learning with soft state aggregation. In G. Tesauro, D. Touretzky, and T. Leen, editors, *Advances in Neural Information Processing Systems 7*, volume 7, pages 361–368. MIT Press, 1995.

A.J. Smola and P.L. Bartlett. Sparse greedy Gaussian process regression. In *Advances in Neural Information Processing Systems 13*, pages 619–625. MIT Press, 2001.

M.J. Sobel. The variance of discounted Markov decision processes. *Journal of Applied Probability*, 19:794–802, 1982.

Peter Sollich and Christopher K. I. Williams. Using the equivalent kernel to understand gaussian process regression. In Lawrence K. Saul, Yair Weiss, and Léon Bottou, editors, *Advances in Neural Information Processing Systems 17*, pages 1313–1320, Cambridge, MA, 2005. MIT Press.

A.L. Strehl and M.L. Littman. An empirical evaluation of interval estimation for markov decision processes. In *Proceedings of the 16th IEEE International on Tools with Artificial Intelligence Conference*, pages 128–135, 2004.

A.L. Strehl and M.L. Littman. A theoretical analysis of model-based interval estimation. In *Proceedings of the 22nd International Conference on Machine Learning*, pages 857–864, 2005.

M. Strens. A Bayesian framework for reinforcement learning. In *Proc. of the 17th International Conf. on Machine Learning*, pages 943–950. Morgan Kaufmann, San Francisco, CA, 2000. URL `citeseer.ist.psu.edu/strens00bayesian.html`.

R.S. Sutton. Learning to predict by the method of temporal differences. *Machine Learning*, 3:9–44, 1988.

R.S. Sutton and Andrew G. Barto. *Reinforcement Learning: An Introduction*. MIT Press, 1998.

G. Tesauro. Temporal difference learning and TD-Gammon. *Comm. ACM*, 38:58–68, 1995.

J.N. Tsitsiklis and B. Van Roy. An analysis of temporal-difference learning with function approximation. Technical Report LIDS-P-2322, MIT, 1996.

T. Wang, D. Lizotte, M. Bowling, and D. Schuurmans. Bayesian sparse sampling for on-line reward optimization. In *Proc. of the 22nd international conference on Machine learning*, pages 956–963, 2005.

C. Williams. Prediction with Gaussian processes: From linear regression to linear prediction and beyond. In M. I. Jordan, editor, *Learning and Inference in Graphical Models*. Kluwer, 1999.

C. Williams and M. Seeger. Using the Nyström method to speed up kernel machines. In *Advances in Neural Information Processing Systems 13*, pages 682–688, 2001.