Appendix for the Paper:

Prediction under Uncertainty in Sparse Spectrum Gaussian Processes with Applications to Filtering and Control

Yunpeng Pan[1, 2], Xinyan Yan[1, 3], Evangelos A. Theodorou[1, 2], and Byron Boots[1, 3]

[1]Georgia Institute of Technology, Atlanta, Georgia, USA
[2]School of Aerospace Engineering
[3]School of Interactive Computing

1 Introduction

In this supplementary material we provide details for the proposed methods presented in the main paper, and additional experimental results.

2 Sparse Spectral Reprsentation of Gaussian processes

To assist later deriviations and avoid reiterating the material in the main paper, we briefly present important equations for Sparse Spectrum GPs (SSGPs). Based on Bochner's theorem, continuous shift-invariant kernels can be unbiasedly approximated by an explicit finite-dimensional feature map. Leveraging this approximation, we consider SSGPs which is a class of Gaussian processes with kernel in the form:

$$\begin{aligned} k(x, x') &= \phi(x)^T \phi(x') + \sigma_n^2 \delta(x - x'),\ \phi(x) = \begin{bmatrix} \phi^c(x) \\ \phi^s(x) \end{bmatrix}, \\ \phi_i^c(x) &= \sigma_k \cos(\omega_i^T x),\ \phi_i^s(x) = \sigma_k \sin(\omega_i^T x),\ \omega_i \sim p(\omega), \end{aligned} \tag{1}$$

where function $\phi : \mathbf{R}^d \to \mathbf{R}^{2m}$ is the explicit finite-dimensional feature map, scalar σ_k is a scaling coefficient, function δ is the Kronecker delta function, and vectors ω_i are sampled according to the spectral density of the kernel to approximate. Consider the task of learning the function $f : \mathbf{R}^d \to \mathbf{R}$, given IID data $\mathcal{D} = \{x_i, y_i\}_{i=1}^n$, with each pair related by

$$y = f(x) + \epsilon, \quad \epsilon \sim \mathcal{N}(0, \sigma_n^2), \tag{2}$$

where ϵ is IID additive Gaussian noise, using Gaussian process regression (GPR). Because of the explicit finite-dimensional feature map ϕ, each SSGP is equivalent to a Gaussian

distribution over the weights of the features $w \in \mathbf{R}^{2m}$. Assuming that prior distribution of weights w is $\mathcal{N}(0, I)$, and the feature map is fixed, after conditioning on data $\mathcal{D} = \{x_i, y_i\}_{i=1}^n$ the posterior distribution of w is

$$w \sim \mathcal{N}(\alpha,\ \sigma_n^2 A^{-1}), \tag{3}$$

$$\alpha = A^{-1}\Phi Y, \quad A = \Phi\Phi^T + \sigma_n^2 I, \tag{4}$$

which can be derived through Bayesian linear regression. In (4), the column vector Y and the matrix Φ are specified by the data $\mathcal{D}$: $Y = \begin{bmatrix} y_1 & \dots & y_n \end{bmatrix}^T$, $\Phi = \begin{bmatrix} \phi(x_1) & \dots & \phi(x_n) \end{bmatrix}$. Consequently, the posterior distribution over the output y in (2) at a test point x is *exactly Gaussian*, in which the posterior variance explicitly captures the model uncertainty in prediction with input x:

$$p(y|x) = \mathcal{N}(\alpha^T\phi(x),\ \sigma_n^2 + \sigma_n^2\|\phi(x)\|_{A^{-1}}^2). \tag{5}$$

We consider multivariate outputs by utilizing conditionally independent scalar models for each output dimension, *i.e.*, assuming for outputs in different dimension y_a and y_b, $p(y_a, y_b|x) = p(y_a|x)p(y_b|x)$.

3 Prediction with uncertain input

In this section, we detail the two proposed methods of propogating uncertainty of x through the probabilitic model $p(y|x)$, with the assumption that 1) the input x is Gaussian distributed: $x \sim \mathcal{N}(\mu, \Sigma)$, and 2) the probabilistic model is represented as a SSGP. Through marginalization, the probability density function of the output is:

$$p(y) = \int p(y|x)p(x)\,\mathrm{d}x.$$

Computing this double integral exactly is intractable. Hence we apply analytically computed Gaussian approximation of the output distribution through: 1) exact moment matching, and 2) linearization.

3.1 Preliminaries

We first provide some identities to facilitate later exposition of the closed-form expressions of exact moment matching (§3.4) and linerization (§3.5). We start with the derivatives for the feature map ϕ (1) using shorthand notation:

$$D\cos(\omega^T x) = -\sin(\omega^T x)\omega^T = c_{d\omega}, \quad D\sin(\omega^T x) = \cos(\omega^T x)\omega^T = s_{d\omega},$$

$$Dc_{d\omega} = -s_{d\omega}^T\omega^T, \quad Ds_{d\omega} = c_{d\omega}^T\omega^T,$$

$$D\phi(x) = M_x = \begin{bmatrix} D\phi^c(x) \\ D\phi^s(x) \end{bmatrix}, \quad D\phi_i^c(x) = \sigma_k c_{d\omega_i}, \quad D\phi_i^s(x) = \sigma_k s_{d\omega_i}.$$

Then Proposition 1 in the main paper for the expectation of sinusoids over multivariate Gaussian distributions can be expressed as

$$\mathbf{E}\cos(\omega^T x) = \exp(-\frac{1}{2}\|\omega\|_\Sigma^2)\cos(\omega^T\mu) = c_\omega, \quad \mathbf{E}\sin(\omega^T x) = \exp(-\frac{1}{2}\|\omega\|_\Sigma^2)\sin(\omega^T\mu) = s_\omega.$$

Proposition 2 for the expectation of the multiplication of sinusoids and linear functions over multivariate Gaussian distributions can be presented similarly:

$$\mathbf{E}\left(\cos(\omega^T x)x\right) = c_\omega\mu - s_\omega\Sigma\omega = c_{x\omega}, \quad \mathbf{E}\left(\sin(\omega^T x)x\right) = s_\omega\mu + c_\omega\Sigma\omega = s_{x\omega}.$$

We can further derive the expectations related to the feature map ϕ defined in (1):

$$\mathbf{E}\,\phi(x) = \psi, \quad \psi = \begin{bmatrix}\psi^c \\ \psi^s\end{bmatrix}, \quad \psi_i^c = \mathbf{E}\,\phi_i^c(x) = \sigma_k c_{\omega_i},$$

$$\mathbf{E}\left(\phi(x)\phi(x)^T\right) = \Psi, \quad \Psi = \begin{bmatrix}\Psi^{cc} & \Psi^{cs} \\ \Psi^{sc} & \Psi^{ss}\end{bmatrix}, \quad \Psi_{ij}^{cc} = \frac{\sigma_k^2}{2}(c_{\omega_i+\omega_j} + c_{\omega_i-\omega_j}),$$

$$\Psi_{ij}^{cs} = \frac{\sigma_k^2}{2}(s_{\omega_i+\omega_j} - s_{\omega_i-\omega_j}), \quad \Psi_{ij}^{ss} = \frac{\sigma_k^2}{2}(-c_{\omega_i+\omega_j} + c_{\omega_i-\omega_j}),$$

$$\mathbf{E}\left(((\alpha^T\phi(x))x)\right) = \Upsilon\alpha, \quad \Upsilon = \begin{bmatrix}\Upsilon_1^c & \dots & \Upsilon_m^c & \Upsilon_1^s & \dots & \Upsilon_m^s\end{bmatrix}, \quad \Upsilon_i^c = \sigma_k c_{x\omega_i}.$$

The derivatives of the preceding expectations to input statistics can be computed, with the notation $D_\mu f$ denoting the derivative of function f with respect to μ:

$$D_\mu c_\omega = -s_w\omega^T, \quad D_\mu s_\omega = c_w\omega^T,$$

$$D_\Sigma c_\omega = -\frac{1}{2}c_\omega\omega\omega^T, \quad D_\Sigma s_\omega = -\frac{1}{2}s_\omega\omega\omega^T,$$

$$D_\mu c_{x\omega} = \mu D_\mu c_\omega + c_\omega I - \Sigma\omega D_\mu s_\omega,$$

$$D_\mu s_{x\omega} = \mu D_\mu s_\omega + s_\omega I + \Sigma\omega D_\mu c_\omega,$$

$$D_\Sigma c_{x\omega} = \mu \otimes D_\Sigma c_\omega - \Sigma\omega \otimes D_\Sigma s_\omega - s_\omega(D_\Sigma\Sigma)\omega, \quad D_\Sigma\Sigma_{i,j} = J_{ji},$$

$$D_\Sigma s_{x\omega} = \mu \otimes D_\Sigma s_\omega + \Sigma\omega \otimes D_\Sigma c_\omega + c_\omega(D_\Sigma\Sigma)\omega,$$

where J_{ij} is the matrix with all zeros except ijth entry being 1, $\otimes$ is tesnor product, and assuming the operators' precedence: $D >$ matrix multiplication $> \otimes$.

3.2 Proof for Proposition 1

The three useful identities involving quadratic exponentials to prove Proposition 1 and 2 are:

$$\int \exp(-x^T Ax + v^T x)\,\mathrm{d}x = \pi^{\frac{d}{2}}\det(A)^{-\frac{1}{2}}\exp(\frac{1}{4}v^T A^{-1}v) = \eta, \tag{6}$$

$$\int (a^T x)\exp(-x^T Ax + v^T x)\,\mathrm{d}x = a^T(\frac{1}{2}A^{-1}v)\eta, \tag{7}$$

$$p(x) = (2\pi)^{-\frac{d}{2}}\det(\Sigma)^{-\frac{1}{2}}\exp(-\frac{1}{2}\|x-\mu\|_{\Sigma^{-1}}^2) \quad x \sim \mathcal{N}(\mu,\Sigma). \tag{8}$$

Proof for $\mathbf{E}\cos(\omega^T x) = \exp(-\frac{1}{2}\|\omega\|_\Sigma^2)\cos(\omega^T\mu)$:

$$
\begin{aligned}
&\int \cos(\omega^T x)\ p(x)\ \mathrm{d}x, \quad x \sim \mathcal{N}(\mu, \Sigma) \\
=&\mathbf{Re}\left(\int \left(\cos(\omega^T x) + i\sin(\omega^T x)\right)\ p(x)\ \mathrm{d}x\right) \\
=&\mathbf{Re}\left(\int \exp(i\omega^T x)(2\pi)^{-\frac{d}{2}}\det(\Sigma)^{-\frac{1}{2}}\exp(-\frac{1}{2}(x-\mu)^T\Sigma^{-1}(x-\mu))\mathrm{d}x\right) \qquad \text{(Use (8))} \\
=&\mathbf{Re}\left(\int \exp(i\omega^T\mu)\exp(i\omega^T(x-\mu))(2\pi)^{-\frac{d}{2}}\det(\Sigma)^{-\frac{1}{2}}\exp(-(x-\mu)^T(2\Sigma)^{-1}(x-\mu))\mathrm{d}x\right) \\
=&(2\pi)^{-\frac{d}{2}}\det(\Sigma)^{-\frac{1}{2}}\,\mathbf{Re}\left(\exp(i\omega^T\mu)\int \exp(-(x-\mu)^T(2\Sigma)^{-1}(x-\mu) + (i\omega)^T(x-\mu))\mathrm{d}x\right) \\
=&(2\pi)^{-\frac{d}{2}}\det(\Sigma)^{-\frac{1}{2}}\,\mathbf{Re}\left(\exp(i\omega^T\mu)\pi^{\frac{d}{2}}\det(\frac{1}{2}\Sigma^{-1})^{-\frac{1}{2}}\exp(\frac{1}{4}(i\omega)^T 2\Sigma(i\omega))\right) \qquad \text{(Use (6))} \\
=&\exp(-\frac{1}{2}\|\omega\|_\Sigma^2)\cos(\omega^T\mu).
\end{aligned}
$$

The other part of Proposition 1: $\mathbf{E}\sin(\omega^T x) = \exp(-\frac{1}{2}\|\omega\|_\Sigma^2)\sin(\omega^T\mu)$ can be shown in a similar fashion, except that the imaginary operator **Im** will be used, instead of **Re**.

3.3 Proof for Proposition 2

To prove Proposition 2, we first prove that:

$$
\begin{aligned}
&\int a^T x\ \cos(\omega^T x)\ p(x)\ \mathrm{d}x, \quad x \sim \mathcal{N}(\mu, \Sigma) \\
=&\exp\left(-\frac{\omega^T\Sigma\omega}{2}\right)\cos(\omega^T\mu)a^T\mu - \exp\left(-\frac{\omega^T\Sigma\omega}{2}\right)\sin(\omega^T\mu)a^T\Sigma\omega.
\end{aligned} \tag{9}
$$

$$
\begin{aligned}
&\int a^T x\ \cos(\omega^T x)\ p(x)\ \mathrm{d}x, \quad x \sim \mathcal{N}(\mu, \Sigma) \\
=&\mathbf{Re}\left(\int a^T x\exp(i\omega^T x)(2\pi)^{-\frac{n}{2}}\det(\Sigma)^{-\frac{1}{2}}\exp(-\frac{1}{2}\|x-\mu\|_{\Sigma^{-1}}^2)\mathrm{d}x\right) \qquad \text{(Use (8))} \\
=&(2\pi)^{-\frac{n}{2}}\det(\Sigma)^{-\frac{1}{2}}\,\mathbf{Re}\left(\int \left(a^T\mu + a^T(x-\mu)\right)\exp(i\omega^T\mu)\exp(i\omega^T(x-\mu))\exp(-\|x-\mu\|_{(2\Sigma)^{-1}}^2)\mathrm{d}x\right) \\
=&a^T\mu\exp(-\frac{1}{2}\|\omega\|_\Sigma^2)\cos(\omega^T\mu) + (2\pi)^{-\frac{n}{2}}\det(\Sigma)^{-\frac{1}{2}}\,\mathbf{Re}\left(\left(\exp(i\omega^T\mu)\int a^T x\exp(-\frac{1}{2}\|x\|_{\Sigma^{-1}} + (i\omega)^T x)\mathrm{d}x\right)\right. \\
=&a^T\mu\exp(-\frac{1}{2}\|\omega\|_\Sigma^2)\cos(\omega^T\mu) + (2\pi)^{-\frac{n}{2}}\det(\Sigma)^{-\frac{1}{2}}\,\mathbf{Re}\left(\exp(i\omega^T\mu)\ ia^T\Sigma\omega\ \pi^{\frac{n}{2}}\det(\frac{1}{2}\Sigma^{-1})^{-\frac{1}{2}}\exp(\frac{1}{4}\|i\omega\|_{2\Sigma}^2)\right) \\
=&a^T\mu\exp(-\frac{1}{2}\|\omega\|_\Sigma^2)\cos(\omega^T\mu) - a^T\Sigma\omega\exp(-\frac{1}{2}\omega^T\Sigma\omega)\sin(\omega^T\mu),
\end{aligned} \tag{10}
$$

where the fourth equality uses (7). Then we can select a as indicator vectors e_i which contains zeros, except with the ith element being 1. After stacking these elements together, we recover the identity of cosine part in Proposition 2. For the sine part, the same techniques apply.

3.4 Exact moment matching

In exact moment matching, we match the mean and variance of the approximated Gaussian distribution with the ones of the true output distribution exactly. For simplicity of notations, henceforth, we suppress the dependency of $\phi(x)$ on x, and keep using y for scalar function values. The predictive mean, variance, covariance between outputs, and cross-covariance between inputs and outputs are

$$\begin{aligned}
\mathbf{E}\, y &= \mathbf{E}\, \mathbf{E}(y|x) = \mathbf{E}\left(\alpha^T \phi\right) = \alpha^T \psi, \\
\mathbf{Var}\, y &= \mathbf{E}\, \mathbf{Var}(y|x) + \mathbf{Var}\, \mathbf{E}(y|x) = \sigma_n^2 + \sigma_n^2\, \mathbf{E}\, \|\phi\|_{A^{-1}}^2 + \mathbf{E}(\alpha^T \phi)^2 - (\mathbf{E}\, y)^2 \\
&= \sigma_n^2 + \mathbf{Tr}\left(\underbrace{(\sigma_n^2 A^{-1} + \alpha\alpha^T)}_{P} \Psi \right) - (\mathbf{E}\, y)^2, \\
\mathbf{Cov}(y_a, y_b) &= \mathbf{Cov}\left(\mathbf{E}(y_a|x), \mathbf{E}(y_b|x)\right) = \alpha_a^T \Psi_{ab} \alpha_b - (\mathbf{E}\, y_a)(\mathbf{E}\, y_b), \\
\mathbf{Cov}(x, y) &= \mathbf{Cov}(x, \mathbf{E}(y|x)) = \mathbf{E}\left(x\, \mathbf{E}(y|x)\right) - (\mathbf{E}\, x)(\mathbf{E}\, y) = \Upsilon\alpha - (\mathbf{E}\, y)\mu,
\end{aligned}$$

where Ψ_{ab} denotes that ω_i and ω_j in the definition of Ψ come from the models for y_a and y_b respectively, since different output dimenstions can have different feature maps, and the constant $\frac{\sigma_k^2}{2}$ is changed to $\frac{\sigma_{k,a}\sigma_{k,b}}{2}$ accordingly. The corresponding derivatives are derived using the chain rule:

$$\begin{aligned}
D_\mu(\mathbf{E}\, y) &= \alpha^T D_\mu \psi, \\
D_\mu(\mathbf{Var}\, y) &= \mathbf{Tr}\left(P D_\mu \Psi\right) - 2(\mathbf{E}\, y)(D_\mu\, \mathbf{E}\, y), \\
D_\mu\, \mathbf{Cov}(y_a, y_b) &= \alpha_a^T (D_\mu \Psi_{ab}) \alpha_b - (\mathbf{E}\, y_a)(D_\mu\, \mathbf{E}\, y_b) - (\mathbf{E}\, y_b)(D_\mu\, \mathbf{E}\, y_a), \\
D_\mu\, \mathbf{Cov}(x, y) = (D_\mu \Upsilon)\alpha - (\mathbf{E}\, y) I - \mu D_\mu(\mathbf{E}\, y), &\quad D_\Sigma\, \mathbf{Cov}(x, y) = (D_\Sigma \Upsilon)\alpha - \mu \otimes D_\Sigma(\mathbf{E}\, y),
\end{aligned}$$

where I is an identity matrix with proper size. Substituting D_μ with D_Σ yields the derivatives to the input covariance matrix Σ if the expressions for D_Σ are not explicitly provided above.

3.5 Linearization

An alternative approach to Gaussian approximations of the predictive distribution is based on the linearization of the posterior mean function in (5) at the input mean μ:

$$m(x) \approx m(\mu) + Dm(\mu)(x - \mu), \quad Dm(\mu) = \alpha^T M_\mu.$$

Then the mean, (co)variance, and cross-covariance for the approximated Gaussian distribution can be computed as

$$\begin{gathered}
\mathbf{E}\, y = \mathbf{E}\, \mathbf{E}[y|x] \approx \mathbf{E}\left(m(\mu) + \alpha^T M(x-\mu)\right) = m(\mu), \\
\mathbf{Var}\, y = \mathbf{E}\, \mathbf{Var}(y|x) + \mathbf{Var}\, \mathbf{E}(y|x) \approx \sigma_n^2 + \sigma_n^2 \|\phi(\mu)\|^2_{A^{-1}} + \alpha^T M \Sigma M^T \alpha, \\
\mathbf{Cov}(y_a, y_b) = \mathbf{Cov}\left(\mathbf{E}(y_a|x), \mathbf{E}(y_b|x)\right) = \mathbf{E}\left(\alpha_a^T M_a (x-\mu)(x-\mu)^T M_b^T \alpha_b\right) = \alpha_a^T M_a \Sigma M_b^T \alpha_b, \\
\mathbf{Cov}(x, y) = \mathbf{Cov}(x, \mathbf{E}(y|x)) = \mathbf{E}\left((x-\mu)(\alpha^T M(x-\mu))\right) = \alpha^T M \Sigma,
\end{gathered}$$

where we omit the subscript μ for M_μ, and M_a, M_b stand for M_μ for model y_a, y_b respectively. Their derivatives to the input statistics are

$$\begin{gathered}
D_\mu(\mathbf{E}\, y) = \alpha^T M, \quad D_\Sigma(\mathbf{E}\, y) = 0, \\
D_\mu(\mathbf{Var}\, y) = 2\sigma_n^T \phi(\mu)^T A^{-1} M + 2\alpha^T (D_\mu M) \Sigma M^T \alpha, \quad D_\Sigma(\mathbf{Var}\, y) = \alpha^T M (D_\Sigma \Sigma) M^T \alpha, \\
D_\mu\, \mathbf{Cov}(y_a, y_b) = \alpha_a^T \left((D_\mu M_a)\Sigma M_b + M_a \Sigma (D_\mu M_b)\right) \alpha_b, \quad D_\Sigma\, \mathbf{Cov}(y_a, y_b) = \alpha_a^T M_a (D_\Sigma \Sigma) M_b^T \alpha_b, \\
D_\mu\, \mathbf{Cov}(x, y) = \alpha^T (D_\mu M) \Sigma, \quad D_\Sigma\, \mathbf{Cov}(x, y) = \alpha^T M (D_\Sigma \Sigma),
\end{gathered}$$

where for simplicity in notation, we omit the fact that approximation has been made in the equations for the derivatives.

4 Trajectory Optimization

4.1 Belief space representation of dynamics

Given samples from the state space dynamics,

$$x_{k+1} = x_k + f(x_k, u_k) + w_k, \quad w_k \sim \mathcal{N}(0, \Sigma_w), \tag{11}$$

We create a SSGP model over f. Given a distribution $p(x_k) = \mathcal{N}(\mu_k, \Sigma_k)$, we can compute the predictive distribution $p(x_{k+1}) \approx \mathcal{N}(\mu_{k+1}, \Sigma_{k+1})$ as follows

$$\begin{aligned}
\mu_{k+1} &= \mu_k + \mathbf{E}\, f_k \\
\Sigma_{k+1} &= \Sigma_k + \mathbf{Cov}\, f_k + \mathbf{Cov}(x_k, f_k) + \mathbf{Cov}(f_k, x_k).
\end{aligned} \tag{12}$$

Note that we use subscript k to denote time step. We define the belief as the predictive distribution $b_k = [\mu_k \ \mathrm{vec}(\Sigma_k)]^{\mathrm{T}}$ over state x_k, where $\mathrm{vec}(\Sigma_k)$ is the vectorization of Σ_k. Therefore eq (12) can be written in a compact form

$$b_{k+1} = \mathcal{F}(b_k, u_k), \tag{13}$$

where $\mathcal{F}$ is defined by (12). The above equation corresponds to the belief space representation of the unknown dynamics (12) in discrete-time.

4.2 Trajectory optimization in belief space

In order to incorporate dynamics model uncertainty explicitly, we perform trajectory optimization in the *Gaussain belief space*. Our proposed framework is based on Differential Dynamic Programming (DDP) [JM70], at each iteration we create a local model along a nominal trajectory through the belief space $(\bar{b}_k, \bar{u}_k)$ including: 1) a linear approximation of the belief dynamics model; 2) a second-order local approximation of the value function. We denote the belief and control nominal trajectory as $(\bar{b}_{1:N}, \bar{u}_{1:N})$ and deviations from this trajectory as $\delta b_k = b_k - \bar{b}_k$, $\delta u_k = u_k - \bar{u}_k$. The linear approximation of the belief dynamics along the nominal trajectory is

$$\delta b_{k+1} \approx \begin{bmatrix} \frac{\partial \mu_{k+1}}{\partial \mu_k} & \frac{\partial \mu_{k+1}}{\partial \Sigma_k} \\ \frac{\partial \Sigma_{k+1}}{\partial \mu_k} & \frac{\Sigma_{k+1}}{\partial \Sigma_k} \end{bmatrix} \delta b_k + \begin{bmatrix} \frac{\partial \mu_{k+1}}{\partial u_k} \\ \frac{\partial \Sigma_{k+1}}{\partial u_k} \end{bmatrix} \delta u_k = \mathcal{F}_k^b \delta b_k + \mathcal{F}_k^u \delta u_k. \tag{14}$$

For a general non-quadratic cost function we approximate it as a quadratic function along the nominal belief and control trajectory (b, u), i.e.,

$$\mathcal{L}(b_k, u_k) \approx \mathcal{L}_k^0 + (\mathcal{L}_k^b)^\mathsf{T} \delta b_k + (\mathcal{L}_k^u)^\mathsf{T} \delta u_k + \frac{1}{2} \begin{bmatrix} \delta b_k \\ \delta u_k \end{bmatrix}^\mathrm{T} \begin{bmatrix} \mathcal{L}_k^{bb} & \mathcal{L}_k^{bu} \\ \mathcal{L}_k^{ub} & \mathcal{L}_k^{uu} \end{bmatrix} \begin{bmatrix} \delta b_k \\ \delta u_k \end{bmatrix}, \tag{15}$$

where superscripts denote partial derivatives, e.g., $\mathcal{L}_k^b = \nabla_b \mathcal{L}_k(b_k, u_k)$ and $\mathcal{L}_k^0 = \mathcal{L}(b_k, u_k)$. We will use this superscript rule for all cost-related terms. All partial derivatives are computed analytically. Based on the dynamic programming principle, the value function is the solution to the Bellman equation

$$V(b_k, k) = \min_{u_k} \big(\underbrace{\mathcal{L}(b_k, u_k) + V\big(\mathcal{F}(b_k, u_k), k+1\big)}_{Q(b_k, u_k)} \big). \tag{16}$$

where V is the value function for the belief b at time step k. At the terminal time step $V(b_N, N) = \mathbf{E}\, h(x(N))$ where $h(x(N))$ is the final cost. $\mathcal{L}(b_k, u_k) = \mathbf{E}\, l(x_k, u_k)$ with l the running cost function. Given the state dynamics in (11) and the cost in (15), our goal is to obtain a quadratic approximation of the value function along the nominal trajectory $\bar{b}_{1:N}$. We wil write this second order approximation as

$$V(b_k, k) \approx V_k^0 + (V_k^b)^\mathsf{T} \delta b_k + \frac{1}{2} \delta b_k^\mathsf{T} V_k^{bb} \delta b_k. \tag{17}$$

where again superscripts denote partial derivatives, e.g., $V_k^b = \nabla_b V_k(b_k, u_k)$ and $V_k^0 = V(b_k, u_k)$. We can do so by expanding the Q-function defined in (16) along $(b_{1:N}, u_{1:N})$

$$Q_k(b_k + \delta b_k, u_k + \delta u_k) \approx Q_k^0 + Q_k^b \delta b_k + Q_k^u \delta u_k + \frac{1}{2} \begin{bmatrix} \delta b_k \\ \delta u_k \end{bmatrix}^\mathrm{T} \begin{bmatrix} Q_k^{bb} & Q_k^{bu} \\ Q_k^{ub} & Q_k^{uu} \end{bmatrix} \begin{bmatrix} \delta b_k \\ \delta u_k \end{bmatrix}, \tag{18}$$

where

$$
\begin{aligned}
Q_k^b &= \mathcal{L}_k^b + V_k^b \mathcal{F}_k^b, \quad Q_k^u = \mathcal{L}_k^u + V_k^b \mathcal{F}_k^u, \\
Q_k^{bb} &= \mathcal{L}_k^{bb} + (\mathcal{F}_k^b)^{\mathrm{T}} V_k^{bb} \mathcal{F}_k^b, \quad Q_k^{ub} = \mathcal{L}_k^{ub} + (\mathcal{F}_k^u)^{\mathrm{T}} V_k^{bb} \mathcal{F}_k^b, \\
Q_k^{uu} &= \mathcal{L}_k^{uu} + (\mathcal{F}_k^u)^{\mathrm{T}} V_k^{bb} \mathcal{F}_k^u,
\end{aligned} \tag{19}
$$

The local optimal control law is computed by minimizing the approximated Q function

$$
\begin{aligned}
\delta \hat{u}_k &= \arg\min_{\delta u_k} \left[Q_k(b_k + \delta b_k, u_k + \delta u_k) \right] \\
&= -(Q_k^{uu})^{-1} Q_k^u - (Q_k^{uu})^{-1} Q_k^{ub} \delta b_k,
\end{aligned} \tag{20}
$$

where the superscripts of Q indicate partial derivatives. The new optimal control is obtained as $\hat{u}_k = \bar{u}_k + \delta \hat{u}_k$. Plugging the optimal control of (20) into the approximated Q-function given by (18) results in the following backward propagation of the value function

$$
V_{k-1} = V_k - Q_k^u (Q_k^{uu})^{-1} Q_k^u, \quad V_{k-1}^b = Q_k^b - Q_k^u (Q_k^{uu})^{-1} Q_k^{ub}, \quad V_{k-1}^{bb} = Q_k^{bb} - Q_k^{vu} (Q_k^{uu})^{-1} Q_k^{ub}.
$$

The optimized control policy $\hat{u}_{1:N}$ is applied to the belief dynamics to generate a new nominal trajectory in a forward pass. We keep optimizing the control policy using this backward-forward scheme iteratively until convergence.

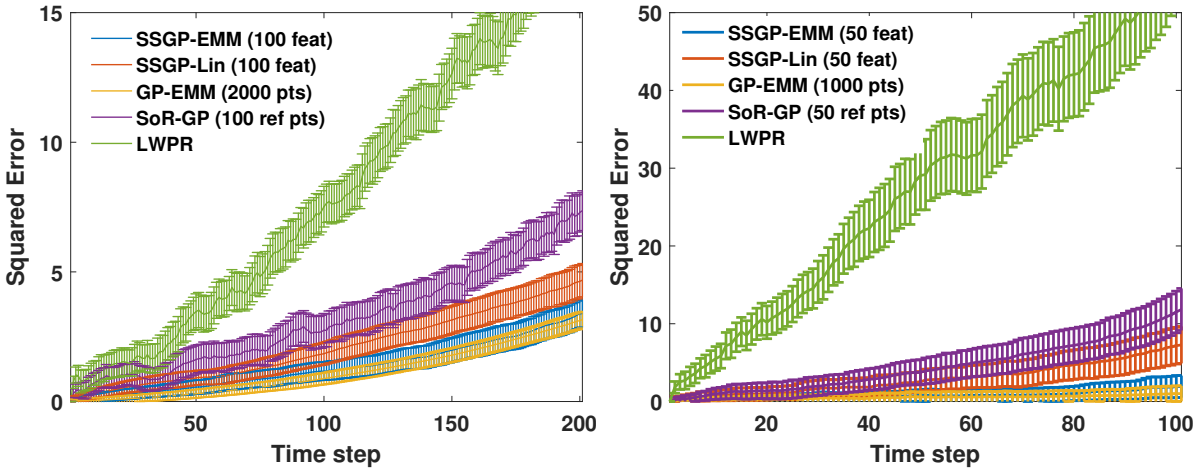
5 Experiments

5.1 Additional experiments on approximate inference

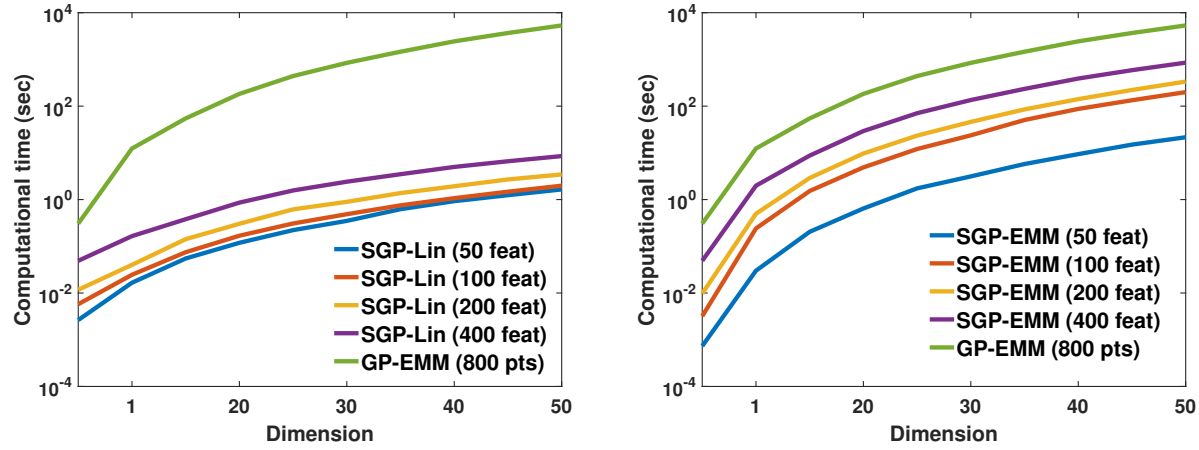
We compare the proposed approximate inference methods with three existing approaches: the full GP exact moment matching (GP-EMM) approach [CGLR03, GRQCMS03, Kus06], Subset of Regressors GP (SoR-GP) [WR06] used in AGP-iLQR [BSWR14], and LWPR [VDS05] used in iLQG-LD [MKV10]. Note that SoR-GP and LWPR do not take into account input uncertainty when performing regressions. We consider two multi-step prediction tasks using the dynamics models of a quadrotor (16 state dimensions, 4 control dimensions) and a Puma-560 manipulator (12 state dimensions, 6 control dimensions).

5.1.1 Accuracy of multi-step prediction

In the following, we evaluate the performance in terms of prediction accuracy. We collected training sets of 1000 and 2000 data points for the quadrotor and puma task, respectively. We used 100 and 50 random features for our methods. We used 100 and 50 reference points for SoR-GP. Based on the learned models, we used a set of 10 initial states and control sequences to perform rollouts (200 steps for quadrotor and 100 steps for Puma) and compute the cost expectations at each step. Fig.1(a)(b) shows the cost prediction errors, i.e.$(\mathcal{L}(x_k) - \mathbf{E}\,\mathcal{L}(x_k))^2$. It can be seen that SSGP-EMM is very close to GP-EMM and SSGP-EMM performs slightly better than SSGP-Lin in all cases. Since SoR-GP and LWPR do not take into account input uncertainty when performing regression, our methods outperform them consistently.



(a) Quadrotor **(b)** Puma 560



(c) Computation time **(d)** Computation time

Figure 1: (a)-(b): Approximate inference accuracy test. The vertical axis is the squared error of cost predictions for (a) quadrotor system and (b) Puma 560 system. Error bars represent standard deviations over 10 independent rollouts. (c)-(d): Comparison of computation time on a log scale between (c) SSGP-Lin and GP-EMM; (d) SSGP-EMM and GP-EMM. The horizontal axis is the input and output dimension (equal in this case). Vertical axis is the CPU time in seconds.

5.1.2 Computational efficiency

In terms of the computational demand, we tested the CPU time for one-step prediction using SSGP-EMM and SSGP-Lin and full GP-EMM. We used sets of 800 random data points of 1,10,20,30,40,50,60,70,80,90 and 100 dimensions to learn SSGP and GP models. The results are shown in fig.1c,1d. Both SSGP-EMM and SSGP-Lin show significantly less computational demand than GP-EMM with similar prediction performance (fig.1c,1d). Our methods are more scalable than GP-EMM, which is the major computational bottleneck for probabilistic model-based RL approaches [DFR15, PT14].

5.2 Model Predictive Control Task descriptions

In the following we describe the model predictive control tasks considered in the main paper. The Bayesian filtering tasks are self-contained therefore we omit further discussions here. In our experiments, we use the popular squared exponential (SE) kernel for SSGPs in order to compare with related methods that are based on the SE kernel. However, as we mentioned in the paper, our proposed methods can be generalized to any continuous, shift-invariant and positive definite kernels. In contrast, analytic moment-based methods for GPs [CGLR03, GRQCMS03, Kus06, DFR15] are restricted to the SE kernel.

5.2.1 PUMA-560 task: moving target and model parameter changes

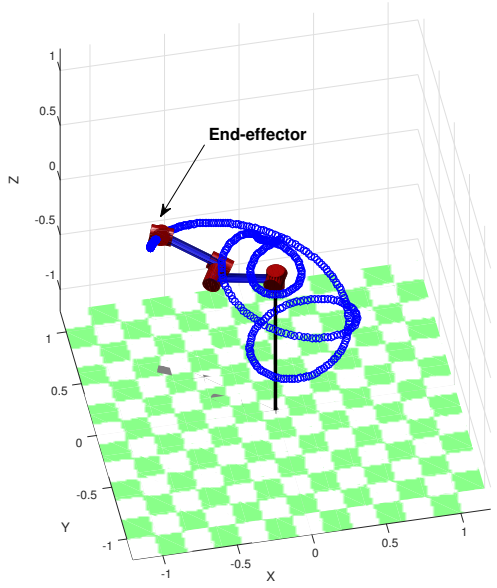
The task is to steer the end-effector to the desired position and orientation. The desired state is time-varying over 800 time steps as shown in fig.2a. We collected 1000 data points offline and sampled 50 random features for both of our methods. Similarly for AGP-iLQR we used 50 reference points. In order to show the effect of online adaptation, we increased the mass of the end-effector by 500% at the beginning of online learning (it is fixed during learning). In the main paper we show the cost reduction results averaged over 3 independent trials. Our method based on SSGP-EMM slightly outperforms SSGP-lin based method.

5.2.2 Quadrotor task: time-varying tasks and dynamics

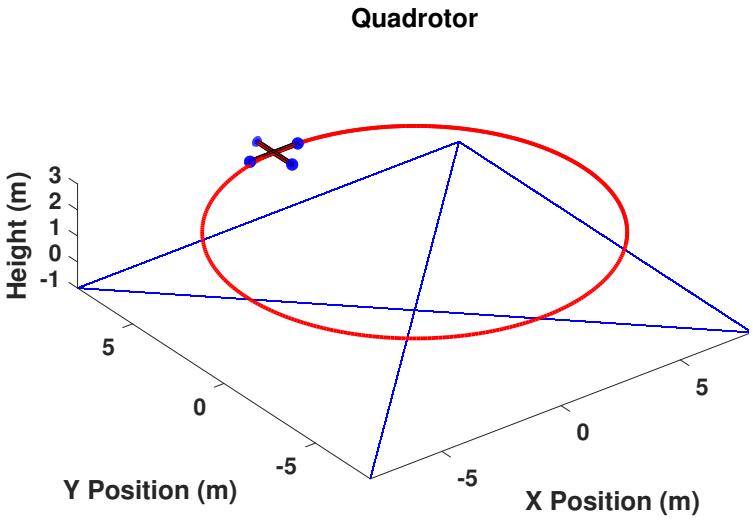
The objective is to start at (-1, 1, 0.5) and track a moving target as shown in fig.**??** for 400 steps. The mass of the quadrotor is decreasing at a rate of 0.02 kg/step. The controls are thrust forces of the 4 rotors and we consider the control constraint $u_{\min} = 0.5, u_{\max} = 3$. We collected 3000 data points offline, and sampled 100 and 400 features for online learning. The forgetting factor for online learning $\lambda = 0.992$. SSGP-Lin was used for approximate inference. The receding-horizon DDP (RH-DDP) [TES07] with full knowledge of the dynamics model was used as a baseline.

5.2.3 Autonomous drifting: steady-state stabilization

In this example, we study the control of a wheeled vehicle during extreme operation conditions (powerslide). The task is to stabilize the vehicle to a specified steady-state using purely longitudinal control. The desired steady-state consists of velocity V , side slip angle β, and yaw rate $\frac{V}{R}$ where R is the path radius. This problem has been studied in [VFT10] where the authors developed a LQR control scheme based on analytic linearization of the dynamics model. However, this method is restrictive due to the assumption of full knowledge of the complex dynamics model. We applied our method to this task under unknown dynamics with 2500 offline data points, which were sampled from the empirical vehicle model in [VFT10]. We used 50, 150, and 400 random features and SSGP-Lin for approximate inference in our experiments. Results and comparisons with the solution in [VFT10] are shown in the main paper.



(a) Puma 560 task



(b) Quadrotor task

Figure 2: PUMA-560 and quadrotor tasks

5.3 Comparison with other methods

Our proposed algorithm is related to several algorithms and differs (listed in Table 1). First, SSGPs are more robust to modeling error than Locally Weighted Projection Regression (LWPR) used in iLQG-LD [MKV10]. See a numerical comparison in [GM13]. Receptive

Field Weighted Regression (RFWR) used by Minimax DDP [MA02] is very similar to LWPR so the conclusion is analogous. Second, we efficiently propagate uncertainty in multi-step prediction which is crucial in MPC. In contrast, AGP-iLQR [BSWR14] drops the input uncertainty and uses subset of regressors (SoR-GP) which lacks a principled way to select reference points. PDDP [PT14] uses GPs which are computationally expensive for online optimization. Two deep neural networks are used for modeling in [YA16], which make it difficult to perform online incremental learning, as we do here.

	Our method	iLQG-LD	PDDP	AGP-iLQR	Minimax DDP	SDDP with NN
Uncertainty propagation	Yes	No	Yes	No	No	Yes
Dynamics model	SSGP	LWPR	GP	SoR-GP	RFWR (partial)	Neural Network
Online model/policy update	Yes/Yes	Yes/No	No/No	Yes/Yes	No/No	No/No

Table 1: Comparison of trajectory optimization-related methods with learned dynamics models. They include: our proposed algorithm, iLQG-LD [MKV10], PDDP [PT14], AGP-iLQR [BSWR14], Minimax DDP [MA02] and SDDP with NN [YA16].

6 Discussion

6.1 Conditional independence between outputs

To deal with multivariate outputs, the assumption of conditional independence between any two output dimensions is imposed, which implies that 1) the noise for different outputs are independent, *e.g.*, Gaussian noise with diagonal covariance matrix, and 2) there's no cross-dependence between channels in the prior, *e.g.*, a vector-valued Gaussian process (GP) prior with a matrix-valued kernel function that only has nonzero entries on the diagonal. These two conditions may be violated in practice. On one hand, the noise may not be independent in general, *e.g.*, wind blowing in some direction causes coupled noise on acceleration for aircraft. On the other hand, one may wish to exploit useful structure between different channels by incorporating them in the prior, *e.g.*, dependence of velocity and acceleration in learning dynamics, and the relation between inverse dynamics models for different loads [WKVC09]. But, nevertheless, conditional independence is assumed in most of the related work [DFR15, KF09]. And we made this assumption in our experiments as well.

To accommodate conditional dependence in a principled way, vector-valued Gaussian processes can be used [BF05, ARL+12]. This generally results in a more complicated model with additional computational cost. Incorporating vector-valued GPs would be an interesting extension of this work.

6.2 SSGP-EKF vs. SSGP-ADF

Given the results shown in fig. 1 in the main paper, SSGP-EKF seems to underestimate the variance of the filtered distribution when the input is around zero compared with GP-EKF. In this experiment, only 10 random Fourier features were used to approximate an SE kernel. As the number of features increases, the filtered distributions using GP-EKF and SSGP-EKF look more similar.

In general, we hypothesize that SSGP-Lin is more sensitive to a small number of features than SSGP-EMM. In SSGP-Lin we use a locally linear approximation of a nonlinear function, and map a Gaussian distribution through this linear function. Intuitively, this locally linear approximation may vary significantly with the number of features, especially when the number of features is small. In contrast, in SSGP-EMM we compute the moments of the predictive distribution without this locally linear approximation. In order to validate this hypothesis, we performed additional experiments on a one-step approximate inference task, shown in fig. 3. This exact phenomenon was observed in these experimental results. More precisely, when a small number of features are used (less than 20), the difference between SSGP-Lin and GP-Lin is greater than the difference between SSGP-EMM and GP-EMM, where the difference is measured by the KL divergence between the predictive distributions.

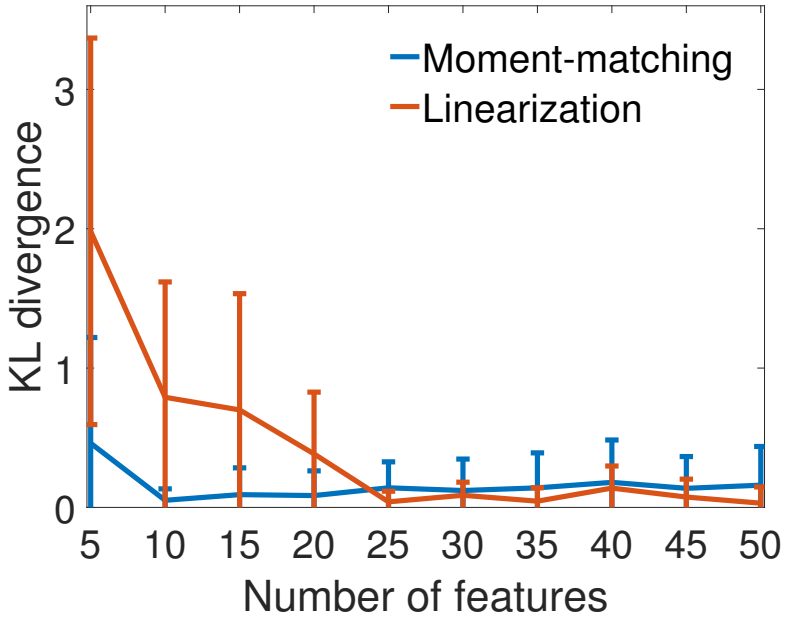


Figure 3: KL divergences between SSGP-EMM and GP-EMM, SSGP-Lin and GP-Lin

References

[ARL+12] Mauricio A Alvarez, Lorenzo Rosasco, Neil D Lawrence, et al. Kernels for vector-valued functions: A review. *Foundations and Trends® in Machine*

Learning, 4(3):195–266, 2012.

[BF05] Phillip Boyle and Marcus Frean. Dependent gaussian processes. In *Advances in neural information processing systems*, pages 217–224, 2005.

[BSWR14] J. Boedecker, JT. Springenberg, J. Wulfing, and M. Riedmiller. Approximate real-time optimal control based on sparse Gaussian process models. In *ADPRL 2014*, pages 1–8. IEEE, 2014.

[CGLR03] J. Quinonero Candela, A. Girard, J. Larsen, and C. E. Rasmussen. Propagation of uncertainty in Bayesian kernel models-application to multiple-step ahead forecasting. In *IEEE International Conference on Acoustics, Speech, and Signal Processing*. IEEE, 2003.

[DFR15] M. Deisenroth, D. Fox, and C. Rasmussen. Gaussian processes for data-efficient learning in robotics and control. *IEEE Transsactions on Pattern Analysis and Machine Intelligence*, 27:75–90, 2015.

[GM13] A. Gijsberts and G. Metta. Real-time model learning using incremental sparse spectrum Gaussian process regression. *Neural Networks*, 41:59–69, 2013.

[GRQCMS03] A. Girard, C.E. Rasmussen, J. Quinonero-Candela, and R. Murray-Smith. Gaussian process priors with uncertain inputs application to multiple-step ahead time series forecasting. In *NIPS*, 2003.

[JM70] D. Jacobson and D. Mayne. Differential dynamic programming. 1970.

[KF09] J. Ko and D. Fox. Gp-bayesfilters: Bayesian filtering using gaussian process prediction and observation models. *Autonomous Robots*, 27(1):75–90, 2009.

[Kus06] Malte Kuss. *Gaussian process models for robust regression, classification, and reinforcement learning*. PhD thesis, Technische Universität, 2006.

[MA02] J. Morimoto and CG Atkeson. Minimax differential dynamic programming: An application to robust biped walking. In *NIPS*, pages 1539–1546, 2002.

[MKV10] D. Mitrovic, S. Klanke, and S. Vijayakumar. Adaptive optimal feedback control with learned internal dynamics models. In *From Motor Learning to Interaction Learning in Robots*, pages 65–84. Springer, 2010.

[PT14] Y. Pan and E. Theodorou. Probabilistic differential dynamic programming. In *Advances in Neural Information Processing Systems (NIPS)*, pages 1907–1915, 2014.

[TES07] Y. Tassa, T. Erez, and W. D. Smart. Receding horizon differential dynamic programming. In *NIPS*, 2007.

[VDS05] Sethu Vijayakumar, Aaron D'souza, and Stefan Schaal. Incremental online learning in high dimensions. *Neural computation*, 17(12):2602–2634, 2005.

[VFT10] E. Velenis, E. Frazzoli, and P. Tsiotras. Steady-state cornering equilibria and stabilisation for a vehicle during extreme operating conditions. *International Journal of Vehicle Autonomous Systems*, 8(2-4):217–241, 2010.

[WKVC09] Christopher Williams, Stefan Klanke, Sethu Vijayakumar, and Kian M Chai. Multi-task gaussian process learning of robot inverse dynamics. In *Advances in Neural Information Processing Systems*, pages 265–272, 2009.

[WR06] C.K.I Williams and C.E. Rasmussen. *Gaussian processes for machine learning*. MIT Press, 2006.

[YA16] Akihiko Yamaguchi and Christopher G Atkeson. Neural networks and differential dynamic programming for reinforcement learning problems. In *2016 IEEE International Conference on Robotics and Automation (ICRA)*, pages 5434–5441. IEEE, 2016.