Online Learning of Inverse Dynamics via Gaussian Process Regression*

Joseph Sun de la Cruz[1], William Owen[2] and Dana Kulić[2]

Abstract— **Model-based control strategies for robot manipulators can present numerous performance advantages when an accurate model of the system dynamics is available. In practice, obtaining such a model is a challenging task which involves modeling such physical processes as friction, which may not be well understood and difficult to model. This paper proposes an approach for online learning of the inverse dynamics model using Gaussian Process Regression. The Sparse Online Gaussian Process (SOGP) algorithm is modified to allow for incremental updates of the model and hyperparameters. The influence of initialization on the performance of the learning algorithms, based on any a-priori knowledge available, is also investigated. The proposed approach is compared to existing learning and fixed control algorithms and shown to be capable of fast initialization and learning rate.**

I. INTRODUCTION

The use of robotics worldwide is most prevalent in the manufacturing industry where the environment is highly controlled and relatively constant. Simple decentralized control strategies such as independent joint PD control [1] are commonly used for basic manipulation tasks such as pick-and-place motions. Unlike decentralized controllers, control strategies that are based on the dynamic model of the manipulator, known as model-based controllers, present numerous advantages such as increased performance during high-speed movements, reduced energy consumption, improved tracking accuracy and the possibility of compliance [2]. However, model based controllers require an accurate model of the system dynamics to achieve good performance. Deriving an accurate model analytically is a difficult task, especially due to physical phenomena which are not well understood or difficult to model, such as friction. Even with the use of dynamic parameter estimation [3], unmodeled dynamics can still reduce the performance of model-based control systems. While adaptive controllers [4], [5], [1] are able to deal with gradual changes in parameter values that may occur from daily wear and tear, they are still unable to account for modeling errors or changes in the model structure. As an alternative to modeling the complex behaviour of these systems, machine learning algorithms for supervised learning can be applied to learn the dynamics. Recent developments in this area of learning control have already yielded promising results that enable robots to learn their own inverse dynamics function with no a-priori knowledge of the system [6],[7].

Locally Weighted Projection Regression (LWPR) is frequently applied [6],[7],[8] to learn the inverse dynamics of a manipulator, due to its use of simple local, linear models which allow online and incremental learning. However, due to its highly localized learning, the system must be first be trained in the expected regions of operation, or appropriately initialized through, for example, motor babbling [8], [6], [9]. Other regression techniques have also been investigated for learning control. Gaussian Process Regression (GPR) has been applied [10] to learn the inverse dynamics function of a manipulator, but due to its heavy computational requirements, GPR has been limited mainly to offline learning. More recent efforts [11],[12] in making GPR computationally tractable for real-time control have resulted in several approximations which operate on a select subset, or sparse representation of the entire training data set.

A comparison between learning approaches such as LWPR and classical control techniques [13] shows that both the adaptive controller and LWPR controller have comparable performance in the presence of parametric uncertainty. However, the learning controller is unable to generalize well outside of the regions in which it has been trained. Hence, achieving good performance requires significant amounts of training in the anticipated region of operation. On the other hand, the adaptive controller shows poor performance when the structure of the dynamics is not fully known, and when the trajectories executed are not persistently exciting.

This paper develops an online, incremental learning algorithm based on Gaussian Process learning, which does not require any knowledge of the structure of the dynamics. A modified version of the the Sparse Online Gaussian Processes (SOGP) algorithm is proposed to allow for online, incremental updates of both the training set and the hyperparameters. We demonstrate the advantage of incorporating any prior knowledge into the learning model by updating the means of the GP model. It is shown that the proposed approach allows the system to operate well even without any initial training data, and further performance improvement can be achieved with additional online training. Furthermore, even partial knowledge of the system dynamics, for example, only the gravity loading vector, can be used effectively to initialize the learning.

II. INVERSE DYNAMICS LEARNING FOR ROBOT MANIPULATORS

The dynamics of a manipulator characterizes the relationship between its motion (position, velocity and acceleration) and the joint torques that cause these motions. The closed-form solution for this relationship is

$$\mathbf{M}(\mathbf{q})\ddot{\mathbf{q}} + \mathbf{C}(\mathbf{q}, \dot{\mathbf{q}}) + \mathbf{G}(\mathbf{q}) = \boldsymbol{\tau} + \mathbf{V_f} \qquad (1)$$

*This work was supported in part by the Canada Natural Sciences and Engineering Research Council.

[1]J. Sun de la Cruz is with National Instruments, Austin, Texas, USA.

[2] W. Owen and D. Kulić are with the University of Waterloo, Canada.

where q is the $n \times 1$ joint position vector, $\mathbf{M}(\mathbf{q})$ is the $n \times n$ inertia matrix, $\mathbf{C}(\mathbf{q}, \dot{\mathbf{q}})$ is the $n \times 1$ centripetal and Coriolis torque vector, $G(q)$ is the gravity loading vector, $\boldsymbol{\tau}$ is the torque vector, and $\mathbf{V_f}$ describes additional torque components due to forces not included in the rigid body model, such as friction and contact forces.

Model-based controllers are a broad class of controllers which apply the joint space dynamic equation (1) to cancel the nonlinear and coupling effects of the manipulator. One example is the inverse dynamics approach [4],[10], also known as computed torque control. In this approach, equation (1) is applied to compensate for nonlinear and coupling effects.The control signal, $\mathbf{u}$ is computed as:

$$\mathbf{u} = \hat{\mathbf{M}}(\mathbf{q})\ddot{\mathbf{q}}_\mathbf{r} + \hat{\mathbf{C}}(\mathbf{q}, \dot{\mathbf{q}}) + \hat{\mathbf{G}}(\mathbf{q}) \tag{2}$$

While model-based approaches can provide superior performance to independent joint control [2], this is contingent upon the assumption that the dynamic model (1) closely matches the actual system, both in the values of the parameters and the structure of the dynamics. In practice, obtaining such a model is a challenging task which involves modeling physical processes that are not well understood or difficult to model, such as friction [14]. Changes to operating conditions can also cause the structure of the system model to change, causing imperfect cancelation of the nonlinearities and coupling in (1).

One solution is the use of dynamic parameter identification [15], [3]. Because the identification procedure is carried out offline, the trajectories can be optimized specifically to excite the dynamics of the system in order to yield more accurate results. However, due to its offline nature, this procedure is not well-suited to deal with systems in which the parameters may vary over time. Adaptive control [4], [16] deals with model uncertainty by attempting to identify a more accurate model of the system through online parameter update laws. While adaptive control can provide an online estimate of the dynamic parameters, is still reliant upon adequate knowledge of the structure of the dynamic model and is thus particularly susceptible to the effects of unmodeled dynamics [13].

Newer approaches to manipulator control involve data-driven learning of the inverse dynamics relationship of a manipulator, thus eliminating the need for a-priori knowledge of the system model. By analyzing the input torques and resulting motion of the manipulator, an approximation of the dynamic equation of the manipulator can be obtained. Unlike the adaptive control strategies, many of these approaches do not assume an underlying structure but rather attempt to infer a model which describes the observed data as closely as possible. Thus, it is possible to encode nonlinearities whose structure may not be well-known. They are particularly useful for modeling processes which are not easily modeled by Newtonian physics, such as nonlinear friction. Related work in supervised learning can be broadly categorized into two types [8] - global methods such Gaussian Process Regression (GPR) [17] and Support Vector Regression (SVR) [2], and local methods such as Locally Weighted Projection Regression (LWPR) [7]. Local learning approaches fit nonlinear functions with spatially localized models, usually in the original input space of the training data [8] [7]. Typically, simple models (linear or low-order polynomial) are used for the local models, and the learning algorithm automatically adjusts the complexity (i.e., number of local models and their locality) to accurately account for the nonlinearities and distributions of the target function.

Global learning methods fit nonlinear functions globally, typically by transforming the input space with predefined or parameterized basis functions and subsequent linear combinations of the transformed inputs. Gaussian Process Regression (GPR) [17] is a global supervised learning technique which employs Gaussian process (GP) models to formulate a Bayesian framework for regression. A GP is characterized by its mean and covariance functions which provide the prior for Bayesian inference. After updating the GP prior with training data to yield the posterior GP, the parameters of the mean and covariance function, known as the hyperparameters, are updated according to the training data. This is typically done by selecting hyperparameters that maximize the probability of observing the training data. The GPR framework has been applied in several previous works [18],[9],[10],[17] to learn the inverse dynamics function of a robot manipulator. However, due to the computational complexity of GPR which scales cubically with the number of training points ($\mathcal{O}(N^3)$), the learning is performed entirely offline. Several approaches to alleviate this problem have been proposed [19] [11],[17],[12]. In [19], online learning is achieved by combining GPR with the approach of local learning, such that each local GP sees only a fraction of the total training data and hence remains computationally tractable. Other researchers rely on sparse representations of the full GP which typically involves representing the full training data set of size N with a subset of data consisting of M_s elements, where $M_s << N$ [11],[17],[12]. A common method of choosing this subset of the data is based on an information gain criterion [12].

A recent example of an approximation of the full GPR procedure is the Sparse Pseudo-input Gaussian Processes (SPGP) [12]. SPGP introduces a method of simultaneously finding an active set of point locations for 'pseudo-inputs', denoted by $\bar{\mathbf{X}}$ while learning the hyperparameters of the Gaussian process in a smooth joint optimization scheme. These pseudo-inputs can be viewed as a parametrization of an approximation of the GP covariance function [12]. Unlike other sparse approximations, the pseudo-inputs are not a fixed subset of the training data but are treated as parameters to be optimized to yield a better fit to the data.

Similar to SPGP, Sparse Online Gaussian Processes (SOGP) [11],[20] is another sparse approximation of the full GPR framework which reduces the computational load of GPR by keeping track of a sparse set of inputs named Basis Vectors (BV). SOGP also incorporates a method of incrementally processing data by treating the posterior mean $\bar{\mathbf{f}}_N$ and covariance functions $K(\mathbf{X}, \mathbf{X})$ as linear combinations of the prior covariance functions $K_0(\mathbf{x}, \mathbf{x}')$.

The initial performance of on-line learning algorithms can be significantly improved by appropriate initialization. Recent work in learning control aims to incorporate the RBD model (1), to provide a global characterization of the dynamics and improve performance in terms of generalization and prediction accuracy [21].

III. THE SPARSE ONLINE GP ALGORITHM

The goal of GPR is to find the function f which maps the inputs $\mathbf{X}$ to their target output values $\mathbf{y}$. A single observed output can be described by

$$y \sim \mathcal{N}(0, K(\mathbf{X}, \mathbf{X}) + \sigma_n^2 \mathbf{I}_n) \tag{3}$$

where σ_n^2 is the noise variance, $\mathbf{I}_n$ is the $n \times n$ identity matrix, and $K(\mathbf{X}, \mathbf{X})$ is the covariance matrix which is composed of the covariances, $k(\mathbf{x}, \mathbf{x}')$, evaluated at all pairs of the training points. When applying the GPR process to learning the inverse dynamics of a robot manipulator, $\mathbf{X}$ are set to the joint angles, velocities and accelerations $\mathbf{q}, \dot{\mathbf{q}}, \ddot{\mathbf{q}}$, and $\mathbf{y}$ is the joint torques $\boldsymbol{\tau}$. A widely used covariance function is given by the squared exponential (SE) form as

$$k(\mathbf{x}, \mathbf{x}') = \theta_1 \exp(-\theta_2 \|\mathbf{x} - \mathbf{x}'\|^2) \tag{4}$$

where θ_1 and θ_2 are the two hyperparameters of the SE covariance term which control the amplitude and characteristic length-scale respectively. The hyperparameters of the Gaussian process can be learned for a particular data set by maximizing the log marginal likelihood using optimization procedures such as gradient-based methods.

To make a prediction $\bar{f}_*(\mathbf{x}^*)$ given a new input vector $\mathbf{x}^*$, the joint distribution of the output values and the predicted function is given by

$$\begin{bmatrix} y \\ \bar{f}_*(\mathbf{x}^*) \end{bmatrix} \sim \mathcal{N}(0, \begin{pmatrix} \begin{bmatrix} K(\mathbf{X}, \mathbf{X}) + \sigma_n^2 \mathbf{I}_n & k(\mathbf{X}, \mathbf{x}^*) \\ k(\mathbf{x}^*, \mathbf{X}) & k(\mathbf{x}^*, \mathbf{x}^*) \end{bmatrix} \end{pmatrix} \tag{5}$$

A training data set of size N requires the inversion of the $N \times N$ matrix, resulting in a computational complexity of $\mathcal{O}(N^3)$ for training with the standard GPR. Once this inversion is done, prediction of the mean and variance require $\mathcal{O}(N)$ and $\mathcal{O}(N^2)$ respectively.

Sparse Online Gaussian Processes (SOGP) [11],[20] is a sparse approximation of the full GPR framework which reduces the computational load of GPR by keeping track of a sparse set of inputs named Basis Vectors (BV). SOGP also incorporates a method of incrementally processing data by treating the posterior mean $\bar{\mathbf{f}}_N$ and covariance functions $K(\mathbf{X}, \mathbf{X})$ as linear combinations of the prior covariance functions $K_0(\mathbf{x}, \mathbf{x}')$:

$$\bar{\mathbf{f}}_N = \bar{\mathbf{f}}_0 + \sum_{i=1}^{N} K_0(\mathbf{x}, \mathbf{x}_i)\alpha_N(i) = \bar{\mathbf{f}}_0 + \boldsymbol{\alpha}_N^T \mathbf{k}_x \tag{6}$$

$$K(\mathbf{X}, \mathbf{X}) = K_0(\mathbf{x}, \mathbf{x}') + \mathbf{k}_x^T \mathbf{C}_N^{GP} \mathbf{k}_{x'} \tag{7}$$

where $\boldsymbol{\alpha}_N = [\alpha_N(1), ..., \alpha_N(t)]^T$, $\mathbf{C}_N^{GP} = \{C_N(ij)\}_{i,j=1}$ are the mean and covariance parameters, and $\mathbf{k}_x = [K_0(\mathbf{x}_1, \mathbf{x}), .., K_0(\mathbf{x}_N, \mathbf{x})]^T$ is the vector of kernel functions centered on each combination of training points. The parameters $\boldsymbol{\alpha}$ and $\mathbf{C}^{GP}$ can be iteratively updated [11] through the following:

$$\begin{aligned} \boldsymbol{\alpha}_{N+1} &= T_{N+1}(\boldsymbol{\alpha}_N) + q_{N+1}(\mathbf{s}_{N+1}) \\ \mathbf{C}_{N+1}^{GP} &= U_{N+1}(\mathbf{C}_N^{GP}) + r_{N+1}(\mathbf{s}_{N+1}\mathbf{s}_{N+1}^T) \\ \mathbf{s}_{N+1} &= T_{N+1}(\mathbf{C}_N^{GP}\mathbf{k}_{N+1}) + \mathbf{e}_{N+1} \end{aligned} \tag{8}$$

where $\mathbf{e}_{N+1} = [0, .., 1]^T$ is a unit vector of length $N+1$, and the vector $\mathbf{s}_{N+1}$ is introduced for clarity of the equations. Operators T_{N+1} and U_{N+1} extend an N-dimensional vector and $N \times N$ dimensional matrix to an $N+1$ vector and $(N+1) \times (N+1)$ matrix respectively by appending a zero to the end of the vector, and a zero row and column to the matrix. The scalars q_{N+1} and r_{N+1} are computed as follows:

$$\begin{aligned} q_{N+1} &= \frac{\delta}{\delta \bar{f}_N(\mathbf{x}_{N+1})} ln\langle P(y_{N+1}|f_N(\mathbf{x}_{N+1}))\rangle_N \\ r_{N+1} &= \frac{\delta^2}{\delta \bar{f}_N^2(\mathbf{x}_{N+1})} ln\langle P(y_{N+1}|f_N(\mathbf{x}_{N+1}))\rangle_N \end{aligned} \tag{9}$$

where $\langle\rangle_N$ denotes the average with respect to the GP at the N^{th} iteration. As seen in Equation 8, the dimensions of $\boldsymbol{\alpha}$ and $\mathbf{C}^{GP}$ increase with each data point added, which is problematic for computational tractability when dealing with large data sets. However, if the new input $\mathbf{x}_{N+1}$ can be represented by a linear combination of covariance functions constructed about the existing N data points, the dimensions of $\boldsymbol{\alpha}$ and $\mathbf{C}^{GP}$ do not need to be increased. This is expressed in the following:

$$K(\mathbf{x}, \mathbf{x}_{N+1}) = \sum_{i=1}^{N} \hat{\mathbf{e}}_{N+1}(i) K_0(\mathbf{x}, \mathbf{x}_i) \tag{10}$$

If the vector $\hat{\mathbf{e}}_{N+1}$ can be found, then the updated GP could be represented using only the first N inputs. However, for most covariance functions and inputs $\mathbf{x}_{N+1}$, Equation 10 cannot be satisfied for all x [11]. Instead, an approximation of the solution can be achieved by minimizing the error measure:

$$\left\| K_0(\mathbf{x}, \mathbf{x}_{N+1}) - \sum_{i=1}^{N} \hat{\mathbf{e}}_{N+1}(i) K_0(\mathbf{x}, \mathbf{x}_i) \right\|^2 \tag{11}$$

where $\|\cdot\|^2$ is a suitably defined norm, which is selected in [11] as a kernel Hilbert space norm. The solution of the minimization problem in Equation 11 is given by:

$$\hat{\mathbf{e}}_{N+1} = \mathbf{K}_N^{-1} \mathbf{k}_{N+1} \tag{12}$$

where $\mathbf{K}_N = \{K_0(\mathbf{x}_i, \mathbf{x}_j)\}_{i,j=1}$. Applying this solution to Equation 10 results in:

$$\hat{K}_0(\mathbf{x}, \mathbf{x}_{N+1}) = \sum_{i=1}^{N} \hat{\mathbf{e}}_{N+1}(i) K_0(\mathbf{x}, \mathbf{x}_i) \tag{13}$$

where $\hat{K}_0(\mathbf{x}, \mathbf{x}_{N+1})$ can be seen as the orthogonal projection of $K_0(\mathbf{x}, \mathbf{x}_{N+1})$ on the linear span of the functions $K_0(\mathbf{x}, \mathbf{x}_i)$.

To determine whether or not the current input $\mathbf{x}_N$ will be

Algorithm 1 Sparse Online Gaussian Process Regression

For a new data point, $(y_{N+1}, \mathbf{x}_{N+1})$
Compute q^{N+1}, r^{N+1} (9), $\hat{\mathbf{e}}_{N+1}$ (12), and γ_{N+1} (14)
if $\gamma_{N+1} < \epsilon_{tol}$ **then**
    Compute $\boldsymbol{\alpha}$ and $\mathbf{C}^{GP}$ without extending their size:
    $\boldsymbol{\alpha}_{N+1} = (\boldsymbol{\alpha}_N) + q_{N+1}(\mathbf{s}_{N+1})$
    $\mathbf{C}^{GP}_{N+1} = \mathbf{C}^{GP}_N + r_{N+1}\mathbf{s}_{N+1}(\mathbf{s}^T_{N+1})$
    $\mathbf{s}_{N+1} = \mathbf{C}^{GP}_N \mathbf{k}_{N+1} + \hat{\mathbf{e}}_{N+1}$ where $\hat{\mathbf{e}}_{N+1}$ is computed according to (12)
else
    Compute $\boldsymbol{\alpha}$ and $\mathbf{C}^{GP}$ according to (8)
    Add current input $\mathbf{x}_{B+1}$ to the BV set
end if
if Size of BV set $> M_{BV}$ **then**
    Compute the novelty, γ, of each BV and remove the lowest scoring one
end if

included in the BV set, the residual error vector from the projection in (13) is calculated:

$$\gamma_{N+1} = K_0(\mathbf{x}_{N+1}, \mathbf{x}_{N+1}) - \mathbf{k}^T_{N+1}\mathbf{K}^{-1}_N \mathbf{k}_{N+1} \tag{14}$$

where γ_{N+1} can be treated as a measure of the 'novelty' of the current input. An input resulting in a high value of γ indicates that the new data cannot be described well through linear combinations of the existing data points, and hence should be added to the BV set. The overall SOGP algorithm is described in Algorithm 1.

IV. ADAPTATION FOR INCREMENTAL LEARNING AND INITIALIZATION

The RBD equation (1) provides a globally valid model representing the dynamics of a robot manipulator. Hence, providing the RBD equation as prior knowledge to the learning controllers would greatly improve their generalization performance. However, due to the complexity of robotic systems, obtaining a complete and accurate RBD model can be difficult and tedious. Therefore, even if partial knowledge of the RBD model is available, this information should be incorporated into the learning algorithms to potentially boost performance. The simplest term of the RBD model in (1) is the gravity loading vector, $G(q)$, as it depends upon the least number of inertial parameters of the system compared to the inertia and centripetal/Coriolis terms of the model [1].

With Gaussian process regression techniques, including the SPGP and SOGP algorithms described above, typically a zero mean function for the Gaussian process is assumed, and hence predictions in areas of the workspace where few training points have been observed are inaccurate. A simple method of incorporating a-priori knowledge is to set the mean function in Equation (3) equal to the available model of the system, thus biasing the system towards the specified a-priori knowledge [21].

While the mean function is biased towards the prior knowledge, the covariance function k (3) still requires adaptation to the data that is observed. The SOGP algorithm proposed by Csato and Opper [11] does not provide a mechanism for incrementally updating the hyperparameters, which can lead to poor performance compared to other GP methods [2]. The hyperparameters of the covariance function should be optimized to the incoming training data. Nonlinear conjugate gradient descent (NCG) [22] is used to minimize the negative log marginal likelihood with respect to the hyperparameters, $\boldsymbol{\Theta}$. As an iterative gradient-based optimization technique, NCG requires the specification of an initial guess of $\boldsymbol{\Theta}$, which is determined by performing the optimization on a batch of training data collected from the system during motor babbling [13].

With standard gradient descent optimization, the location, $\boldsymbol{\chi}$, of the extremum of a function, f, is found by iteratively taking steps in the direction of the gradient of the function at the current point. That is:

$$\boldsymbol{\chi}_{i+1} = \boldsymbol{\chi}_i + a_i \mathbf{d}_i \tag{15}$$

where $\boldsymbol{\chi}_i$ is the current solution (i^{th} iteration) to the optimization problem, a_i represents the length of the step to be taken, and $\mathbf{d}_i$ is the step direction. This iterative procedure often results in steps being taken in the same direction as in earlier steps resulting in a slower convergence. By employing NCG, this repetition is avoided by ensuring that all search directions are conjugate, or A-orthogonal, and that the necessary step lengths are taken such that only one step must be taken in each search direction to arrive at the extremum for that direction. Thus, the convergence of NCG is better than or at least the same as standard gradient descent.

In order to find the set of conjugate search directions, $\mathbf{d}$, the Polak-Ribiére formula is used to generate coefficients $\boldsymbol{\xi}$:

$$\boldsymbol{\xi}_{i+1} = \frac{\mathbf{r}^T_{i+1}(\mathbf{r}_{i+1} - \mathbf{r}_i)}{\mathbf{r}^T_i \mathbf{r}_i} \tag{16}$$

where i represents the current iteration, and the residual $\mathbf{r}$ is set to the negation of the gradient at the current location $\boldsymbol{\chi}_i$, i.e. $\mathbf{r} = -f'(\boldsymbol{\chi}_i)$. Thus, the conjugate search directions, $\mathbf{d}$, can be determined:

$$\mathbf{d}_{i+1} = \mathbf{r}_{i+1} + \boldsymbol{\xi}_{i+1}\mathbf{d}_i \tag{17}$$

With the selection of the step direction, $\mathbf{d}_i$, the problem is reduced to a one-dimensional optimization case which involves finding the best step length, a, to take in direction $\mathbf{d}_i$. This is done by a line search procedure which finds the a_i that minimizes $f(\boldsymbol{\chi}_i + a_i\mathbf{d}_i)$. The interpolation method of line searching achieves this by firstly approximating the function $f(\boldsymbol{\chi} + a\mathbf{d})$ as a polynomial through a Taylor series expansion. Here we drop the iteration index for simplicity:

$$\begin{aligned} f(\boldsymbol{\chi} + a\mathbf{d}) \approx & f(\boldsymbol{\chi}) + a[\frac{d}{da} f(\boldsymbol{\chi} + a\mathbf{d})]_{a=0} \\ & + \frac{a^2}{2}[\frac{d^2}{da^2} f(\boldsymbol{\chi} + a\mathbf{d})]_{a=0} \\ = & f(\boldsymbol{\chi}) + a[f'(\boldsymbol{\chi})]^T\mathbf{d} + \frac{a^2}{2}\mathbf{d}^T f''(\boldsymbol{\chi})\mathbf{d} \end{aligned} \tag{18}$$

Algorithm 2 Nonlinear Conjugate Gradient Optimization

given: initial guess $\boldsymbol{\chi}$, maximum allowable CG and line search iterations, $\epsilon_{CG}, \epsilon_{LS}$
initialize: $\mathbf{r} \leftarrow -f'(\boldsymbol{\chi})$, $d \leftarrow \mathbf{r}$, $i \leftarrow 0$
while $i < i_{max}$ and $\|\mathbf{r}_i\| \leq \epsilon_{CG}\|\mathbf{r}_0\|$ **do**
  initialize: $j \leftarrow 0$
  $\boldsymbol{\zeta}_{prev} \leftarrow [f'(\boldsymbol{\chi} + \sigma\mathbf{d})]^T\mathbf{d}$
  do
    $\boldsymbol{\zeta} \leftarrow [f'(\boldsymbol{\chi})]^T\mathbf{d}$
    $a \leftarrow -\sigma\frac{\boldsymbol{\zeta}}{\boldsymbol{\zeta}_{prev} - \boldsymbol{\zeta}}$
    $\boldsymbol{\chi} \leftarrow \boldsymbol{\chi} + a\mathbf{d}_i$
    $\boldsymbol{\zeta}_{prev} \leftarrow \boldsymbol{\zeta}$
    $j \leftarrow j + 1$
  while $j < j_{max}$ and $\|a\mathbf{d}\| \leq \epsilon_{LS}$
  $\mathbf{r}_{prev} \leftarrow \mathbf{r}$
  $\mathbf{r} \leftarrow -f'(\boldsymbol{\chi})$
  $\boldsymbol{\xi} \leftarrow \frac{\mathbf{r}^T(\mathbf{r} - \mathbf{r}_{prev})}{\mathbf{r}_{prev}^T\mathbf{r}_{prev}}$
  $\mathbf{d} \leftarrow \mathbf{r} + \boldsymbol{\xi}\mathbf{d}$
  $i \leftarrow i + 1$
end while

The secant method is then used to approximate the second derivative, f'', as a finite difference equation of the first derivative f' evaluated at two points, $a = 0$ and $a = \sigma$:

$$\frac{d^2}{da^2} \approx \frac{1}{\sigma}[f'(\boldsymbol{\chi} + \sigma\mathbf{d})]^T\mathbf{d} - [f'(\boldsymbol{\chi})]^T\mathbf{d} \tag{19}$$

where σ is a non-zero number such that the closer σ is to 0, the better the approximation of the second derivative. Substituting this expression into (18) and differentiating:

$$\frac{d}{da}f(\boldsymbol{\chi}+a\mathbf{d}) \approx [f'(\boldsymbol{\chi})]^T\mathbf{d} + \frac{a}{\sigma}\left\{[f'(\boldsymbol{\chi} + \sigma\mathbf{d})]^T\mathbf{d} - [f'(\boldsymbol{\chi})]^T\mathbf{d}\right\} \tag{20}$$

Minimization of $f(\boldsymbol{\chi}+a\mathbf{d})$ can then be achieved by equating its derivative to zero and rearranging for a:

$$a = -\sigma\frac{\boldsymbol{\zeta}}{\boldsymbol{\zeta}_{prev} - \boldsymbol{\zeta}} \tag{21}$$

where $\boldsymbol{\zeta}_{prev} = [f'(\boldsymbol{\chi} + \sigma\mathbf{d})]^T\mathbf{d}$ and $\boldsymbol{\zeta} = [f'(\boldsymbol{\chi})]^T\mathbf{d}$. This process is then repeated, with the current $\boldsymbol{\chi}$ equated to the value of $\boldsymbol{\chi} + a\mathbf{d}$ calculated in the previous iteration. The stop condition for the line search is based on the gradient approaching orthogonality with the current search direction, i.e. $f'^T\mathbf{d} \approx 0$. High tolerances for this term have been found to be too computationally inefficient for the relatively small gain in accuracy [22] that is achieved.

In order to test the effects of limiting the maximum number of line search iterations on the accuracy of the SOGP algorithm, the system was trained on data sets obtained from [13] and the MSE of prediction was monitored. In the first case, illustrated at the top of Figure 1, up to 50 line search iterations were allowed (i.e. $j_{max} = 50$), and below in the second graph, 25 iterations were allowed. Despite this difference, the final MSE of both cases is nearly identical. The only noticeable discrepancy is the initially slower rate of convergence of the joints in the case of early line search termination. Joint 1 also exhibits non-monotonic convergence initially. However, due to the lower number of allowable line search iterations, the computational burden is reduced and thus the frequency at which the overall hyperparameter updates are performed can be increased. Hence, for the simulations, less accurate, but more frequent line searches were used by terminating the search after 25 iterations.

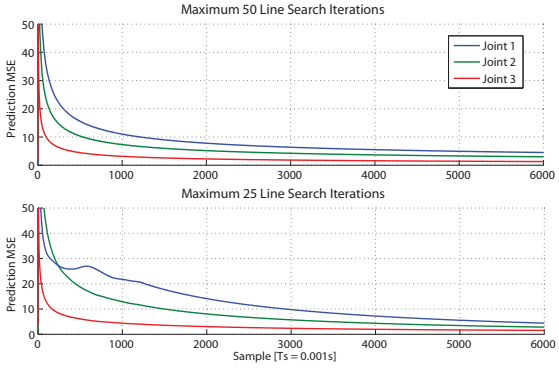

Fig. 1. Prediction MSE

Despite its improved convergence over standard gradient descent, each call to NCG optimization requires a significant number of time steps of the control loop to compute. Thus, the code is broken down into a sequence of smaller, 'atomic' code segments, so that the computation of NCG occurs over multiple controller time steps. This was done so that each code segment would have at most one call to evaluate the function derivative f'. Based on Algorithm 2, the code segments are separated into lines $1:2$, $3:5$, $6:12$ and $13:18$, with each code segment being executed over a single time step of the controller. The optimized location of the hyperparameters, represented by $\boldsymbol{\chi}$, is not used in the control loop until the entire algorithm has been executed.

V. SIMULATIONS

The proposed approach is evaluated in simulation on a 6-DOF Puma 560 robot using the Robotics Toolbox (RTB) [23] and compared to prior-knowledge initialized LWPR [24] and standard computed torque control. The open-source LWPR [8], SPGP [12] and SOGP [11] code were modified to incorporate a-priori knowledge and incremental updating as described above. In order to simulate the nonlinearities present in a physical robot, the effects of Coloumb and viscous friction were simulated with the friction constants obtained from the defaults for the Puma 560 in the RTB. Furthermore, to simulate the effects of imprecise knowledge of the inertial parameters of the robot, a 10% percent error in the inertial parameters of the a-priori knowledge was introduced. For both SPGP and SOGP, the maximum size of the sparse subset of datapoints, M, was set to 35. This number was chosen as the smallest possible value before a noticeable degradation in prediction performance was introduced.

A-priori knowledge from the full RBD model in (1), or partial knowledge from the gravity loading vector are used

to initialize each algorithm.

Tracking performance of the controllers is evaluated on a 'star-like' asterisk pattern [25]. The asterisk trajectory, shown in Figure 2, is challenging due to the high components of velocity and acceleration, and thus requires model-based control for good tracking accuracy. Inverse kinematics is used to convert this trajectory from task to joint space.

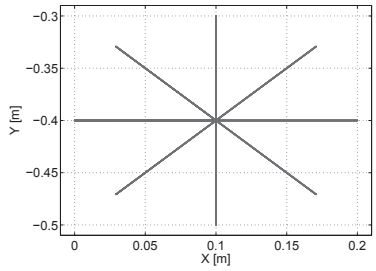

Fig. 2. 'Asterisk' trajectory

A. Results

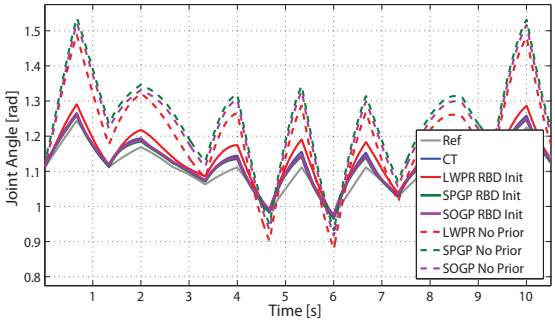

Fig. 3. Initial Tracking Performance for Joint 2 with full RBD Prior and No Prior

1) Full Knowledge of RBD: Figure 3 depicts the joint space tracking performance of the computed torque (CT) controller using a model with 10% error in the inertial and friction parameters of the system. As seen from the figure, the parameter error causes poor tracking results. The same RBD model is used to initialize both GP and LWPR models, and the resulting controllers are trained online while tracking the asterisk trajectory. Figure 3 and Table I show the joint space tracking performance after one cycle of the trajectory. The performance of the GP controllers is very similar to that of the computed torque controller, as they are initialized with the same RBD equation. On the other hand, as the LWPR model is initialized with a set of linear approximations of RBD equation, the initial performance of LWPR is not as good as the GP or the CT methods.

Figure 4 and Table I show the results after 90 seconds of additional training. Due to the high computational efficiency of LWPR, incremental updates are made at a rate of 400 Hz, while the more computationally taxing GP algorithms limit updates to a rate of 100 Hz. Hence, as time progresses, LWPR is able to accumulate more training data than the GP

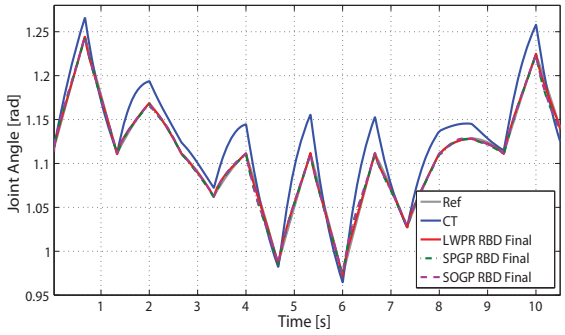

Fig. 4. Final Tracking Performance for Joints 2 with full RBD Prior

controllers, eventually performing nearly as well as SPGP and SOGP. Initially, SPGP performs better than SOGP due to the fact that SPGP uses the full $M = 35$ sparse inputs from the start, whereas SOGP is initialized with $M = 1$ and adds sparse inputs as data is incrementally processed until the limit $M = 35$ is reached. Thus, after seeing significant amounts of training data, both SOGP and SPGP perform very similarly. Lastly, given sufficient data, all three learning controllers are able to outperform the CT method by learning the nonlinear behaviour of Coloumb and viscous friction and compensating for the initial inaccurate knowledge of the RBD equation.

TABLE I
RMS TRACKING ERROR WITH FULL KNOWLEDGE (DEG)

Joint	1	2	3	4	5	6	Avg
LWPR, Init	1.25	2.26	1.74	0.65	0.75	0.80	1.24
LWPR, 90s	0.75	0.94	0.82	0.25	0.38	0.45	0.60
SPGP, Init	1.10	1.75	1.55	0.60	0.68	0.75	1.07
SPGP, 90s	0.70	0.85	0.78	0.22	0.32	0.45	0.55
SOGP, Init	1.21	1.88	1.60	0.63	0.70	0.82	1.14
SOGP, 90s	0.72	0.84	0.80	0.22	0.32	0.47	0.56

2) Partial Knowledge of RBD: Figure 5 and Table II illustrate the joint space tracking performance of the SPGP, SOGP and LWPR models when initialized with only the gravity loading vector of the RBD equation. In this case, LWPR outperforms SPGP during the first cycle. This can be attributed to the higher update rate of LWPR, allowing the model to adapt more quickly to the data it receives. After multiple cycles through the trajectory, this advantage of LWPR vanishes, as the GP models have then observed enough training data across the entire asterisk trajectory to learn a more accurate model. Figure 6 and Table II illustrate the performance after 150 seconds of training. Similarly to the case of full knowledge of the RBD, the learning controllers have all compensated for friction and clearly outperform the CT controller. However, since the system was initialized with only partial knowledge of the RBD equation, it has taken longer for both models to achieve the same tracking performance in the case of full RBD knowledge. Similarly to the case of full knowledge of RBD, SPGP initially performs better than SOGP, but after training this

discrepancy is minimized.

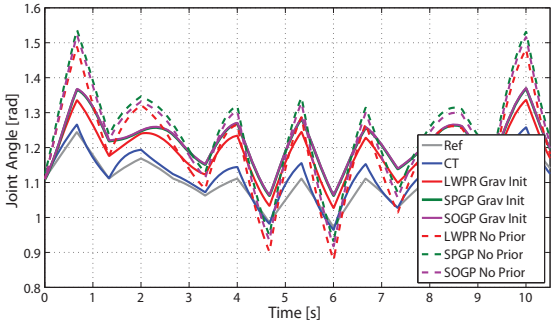

Fig. 5. Initial Tracking Performance for Joint 2 with Gravity Prior and No Prior

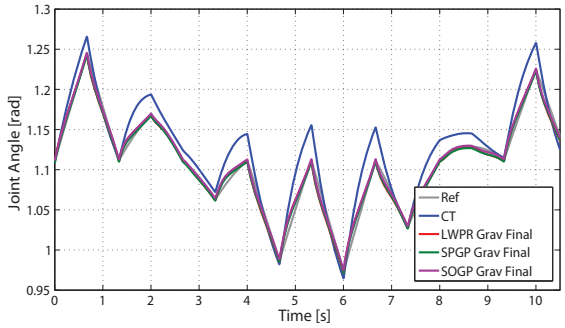

Fig. 6. Final Tracking Performance for Joints 2 with Gravity Prior

TABLE II
RMS TRACKING ERROR WITH PARTIAL KNOWLEDGE (DEG)

Joint	1	2	3	4	5	6	Avg
LWPR, Init	2.57	4.78	3.62	0.95	0.80	0.75	2.25
LWPR, 150s	0.80	0.95	0.80	0.22	0.41	0.44	0.60
SPGP, Init	0.48	6.66	5.29	0.05	0.12	0.27	2.15
SPGP, 150s	0.75	0.84	0.75	0.25	0.30	0.40	0.55
SOGP, Init	0.50	6.75	5.43	0.10	0.14	0.30	2.20
SOGP, 150s	0.77	0.83	0.76	0.25	0.32	0.39	0.55

Without the use of a-priori knowledge, learning algorithms were typically initialized with large training data sets obtained through motor babbling [8], [6], [9] in order to achieve decent tracking performance. By incorporating a-priori knowledge of the RBD equation, whether partial or full, it is shown in these results that the proposed systems are able to perform reasonably well from the start, even without undergoing such an initialization procedure.

3) Algorithm Computation Time: In order to evaluate the computational efficiency of LWPR, SPGP and SOGP, each algorithm was trained on data sets of varying size (2,4,6,8,10,12 and 14 thousand training points) obtained from simulation. Figure 7 illustrates the computation time required to compute a single torque prediction given an input $\mathbf{x}^*$ as a function of the number of training data points that the algorithm has processed. The results were obtained on a PC running Windows XP with a CPU clock speed of 2.66 GHz and 4 GB of RAM. For SOGP and SPGP, the maximum size of the sparse representation set was limited to 35.

As seen in Figure 7, LWPR has the lowest computational cost which remains relatively constant with an increasing data set size. This is due to the fact that LWPR does not explicitly store the entire training set, but rather incorporates them into local linear models. For the sparse GP techniques, SOGP clearly outperforms SPGP. This can be attributed to the online procedure of SOGP which allows data points to be processed incrementally. Although SPGP also uses a sparse representation, data is still processed in batches, and no provision for incremental updates is provided. Because of this, the computation time for a single prediction from SPGP increases much more rapidly than SOGP as the number of training points increases.

Although the simulation results show that the SPGP controller marginally outperforms the SOGP controller in terms of tracking performance (see Tables I and II), the performance gains from SPGP are not significant enough to offset the higher computational costs which increase significantly as the training data set increases. While the simulation platform was able to handle the computational requirements for SPGP while maintaining near-real time performance, it should be noted that this was only for 150 seconds of operation (corresponding to 15,000 training points). For long-term incremental learning, it is expected that the computational requirements of SPGP will prohibit real-time control.

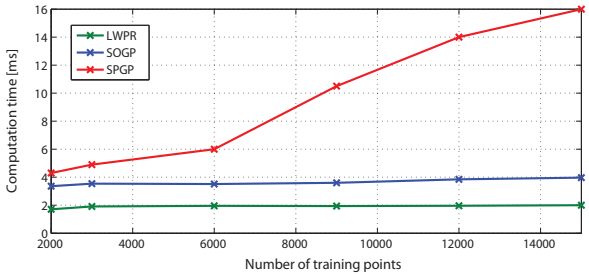

Fig. 7. Computation time required for a single prediction

VI. CONCLUSIONS

An incremental, online learning method was proposed for approximating the inverse dynamics equation while incorporating full or partial knowledge of the RBD equation. Prior knowledge was incorporated into the GP framework by setting the mean function equal to the RBD equation. The incorporation of prior knowledge improves generalization performance, as the model is able to perform comparably to computed torque control without having seen any relevant training data beforehand. Online learning with SOGP was achieved by spreading out the optimization phase of the algorithm over several timesteps, allowing the system to train itself at a rate of 10 Hz. The approach was compared to CT

control and a local learning method, LWPR. Although the greater computational efficiency of LWPR allows updates to occur at a rate of 100 Hz, after a short period of training, the performance of LWPR and SOGP was nearly identical in simulation, even though LWPR had accumulated more training data, due to the global nature of GPR techniques. Both approaches are able to compensate for the nonlinear effects of friction, as well as the initial inaccuracy in the known inertial parameters.

In future work, we intend to validate the proposed algorithm on an experimental platform, and compare to other control strategies such as adaptive control. The simulation work suggests that SOGP will yield tracking results that are very similar to LWPR, but the rate of convergence of the error will be much faster for SOGP.

REFERENCES

[1] L. Sciavicco and B. Scicliano, *Modelling and Control of Robot Manipulators*, 2nd ed. Springer, 2000.

[2] D. Nguyen-Tuong, B. Scholkopf, and J. Peters, "Sparse online model learning for robot control with support vector regression," in *IEEE/RSJ Int Conf on Intelligent Robots and Systems*, Oct. 2009, pp. 3121–3126.

[3] P. Khosla, "Categorization of parameters in the dynamic robot model," *IEEE Trans on Robotics and Automation*, vol. 5, no. 3, pp. 261 –268, Jun 1989.

[4] J. Craig, P. Hsu, and S. Sastry, "Adaptive control of mechanical manipulators," *Int Journal of Robotics Research*, vol. 6, no. 2, pp. 16–28, June 1987.

[5] R. Ortega and M. Spong, "Adaptive motion control of rigid robots: a tutorial," in *IEEE Conf on Decision and Control*, Dec 1988, pp. 1575–1584.

[6] J. Peters and S. Schaal, "Learning to Control in Operational Space," *Int Journal of Robotics Research*, vol. 27, no. 2, pp. 197–212, 2008.

[7] S. Schaal, C. Atkeson, and S. Vijayakumar, "Scalable techniques from nonparametric statistics for real time robot learning," *Appl. Intelligence*, vol. 16, pp. 49–60, 2002.

[8] S. Vijayakumar, A. D'souza, and S. Schaal, "Incremental online learning in high dimensions," *Neural Computation*, vol. 17, no. 12, pp. 2602–2634, 2005.

[9] D. Nguyen-Tuong, J. Peters, M. Seeger, B. Schlkopf, and M. Verleysen, "Learning inverse dynamics: A comparison," Tech. Rep., 2008. [Online]. Available: http://edoc.mpg.de/420029

[10] D. Nguyen-tuong, M. Seeger, and J. Peters, "Computed torque control with nonparametric regression models," in *Proceedings of the 2008 American Control Conference*, 2008, pp. 212–217.

[11] L. Csato and M. Opper, "Sparse on-line gaussian processes," *Neural Computation*, vol. 14, no. 3, pp. 641–668, 2002.

[12] E. Snelson and Z. Ghahramani, "Sparse gaussian processes using pseudo-inputs," in *Advances in Neural Information Processing Systems*. MIT press, 2006, pp. 1257–1264.

[13] J. S. de la Cruz, D. Kulić, and W. Owen, "A comparison of classical and learning controllers," in *World Congress of the International Federation of Automatic Control*, 2011, pp. 1102–1107.

[14] B. Armstrong-Hélouvry, P. Dupont, and C. Canudas de Wit, "A survey of models, analysis tools and compensation methods for the control of machines with friction," *Automatica*, vol. 30, no. 7, pp. 1083–1138, 1994.

[15] K. Ayusawa, G. Venture, and Y. Nakamura, "Identification of humanoid robots dynamics using floating-base motion dynamics," in *IEEE/RSJ Int Conf on Intelligent Robots and Systems*, Sept. 2008, pp. 2854–2859.

[16] M. Spong and M. Vidyasagar, "Robust linear compensator design for nonlinear robotic control," *Robotics and Automation, IEEE Journal of*, vol. 3, no. 4, pp. 345–351, August 1987.

[17] C. Rasmussen and C. Williams, *Gaussian Processes for Machine Learning*. The MIT Press, 2006.

[18] K. Ming, A. Chai, C. K. I. Williams, S. Klanke, and S. Vijayakumar, "Multi-task gaussian process learning of robot inverse dynamics," in *Advances in Neural Information Processing Systems*, 2009.

[19] D. Nguyen-Tuong and J. Peters, "Learning robot dynamics for computed torque control using local gaussian processes regression," in *Proceedings of the 2008 ECSIS Symposium on Learning and Adaptive Behaviors for Robotic Systems*, 2008, pp. 59–64.

[20] L. Csato, "Gaussian processes - iterative sparse approximations," Ph.D. dissertation, Aston University, 2002.

[21] D. Nguyen-Tuong and J. Peters, "Using model knowledge for learning inverse dynamics," in *IEEE Int Conf on Robotics and Automation*, 2010, pp. 2677–2682.

[22] J. R. Shewchuk, "An Introduction to the Conjugate Gradient Method Without the Agonizing Pain," Pittsburgh, PA, USA, Tech. Rep., 1994.

[23] P. Corke, "A robotics toolbox for MATLAB," *IEEE Robotics and Automation Magazine*, vol. 3, no. 1, pp. 24–32, Mar. 1996.

[24] J. S. de la Cruz, D. Kulić, and W. Owen, "Online incremental learning of inverse dynamics incorporating prior knowledge," in *International Conference on Autonomous and Intelligent Systems*, 2011.

[25] J. Nakanishi, R. Cory, M. Mistry, J. Peters, and S. Schaal, "Operational Space Control: A Theoretical and Empirical Comparison," *Int Journal of Robotics Research*, vol. 27, no. 6, pp. 737–757, 2008.