Learning Inverse Dynamics for Redundant Manipulator Control

Joseph Sun de la Cruz, Dana Kulic
Department of Electrical and Computer Engineering
University of Waterloo
Waterloo, ON, Canada
{jsundela, dkulic}@uwaterloo.ca

William Owen
Department of Mechanical and Mechatronics Engineering
University of Waterloo
Waterloo, ON, Canada
bowen@uwaterloo.ca

***Abstract*—High performance control of robotic systems, including the new generation of humanoid, assistive and entertainment robots, requires adequate knowledge of the dynamics of the system. This can be problematic in the presence of modeling uncertainties as the performance of classical, model-based controllers is highly dependant upon accurate knowledge of the system. In addition, future robotic systems such as humanoids are likely to be redundant, requiring a mechanism for redundancy resolution when performing lower degree-of-freedom tasks. In this paper, a learning approach to estimating the inverse dynamic equations is presented. Locally Weighted Projection Regression (LWPR) is used to learn the inverse dynamics of a manipulator in both joint and task space and the resulting controllers are used to drive a 3 and 4 DOF robot in simulation. The performance of the learning controllers is compared to a traditional model based control method and is also shown to be a viable control method for a redundant system.**

***Index Terms*—robotics, learning, control, redundancy resolution**

I. Introduction

The development of robotics has yielded many types of control schemes for robotic manipulation. Motion control schemes based on independent joint position tracking control are most prevalent in industrial robots, enabling them to perform simple tasks such as repetitive pick-and-place motions. An alternative formulation is model-based tracking control, where knowledge of the system dynamics is included in the control loop. This can present numerous advantages such as compliance, increased performance during high-speed movements, reduced energy consumption and improved tracking accuracy [1] as compared to methods which do not utilize knowledge of the dynamics of the system. However, the performance of model-based control is highly dependant upon having an accurate representation of the robot's dynamics, which includes knowledge of inertial parameters such as link mass, centre of mass and moments of inertia, and friction parameters. Obtaining such a model is a complex task which involves the modeling of nonlinear and highly coupled behaviour, including physical processes such as backlash and friction which are not well understood or difficult to model. Thus, assumptions are often made to simplify the modeling process, leading to inaccuracies in the model. Furthermore, uncertainties in the physical parameters of a system may be introduced from significant discrepancies between the manufacturer data and the actual system [2]. Changes to operating conditions can also cause the structure of the system model to change, thus resulting in degraded performance. Recently, machine learning approaches have been proposed to model the inverse dynamics of a manipulator. Learning approaches do not assume a model structure, but build a model based on the training data, treating the inverse dynamics as a function estimation problem. The approach can be divided into two broad classes [3], global and local methods. Global methods generally attempt to fit the nonlinear function globally through input space expansion and linear combinations of the expanded inputs. Gaussian Process Learning (GPR)[4] and Support Vector Regression (SVR) [1] are examples of global methods. Until recently, GPR and SVR methods have primarily been used for offline batch data analysis due to high computational requirements [3]. Recent approaches to approximating SVR through sparsification [5] allow for online, global incremental learning. Local learning consists of methods which fit the nonlinear function with spatially localized linear models in the original input space and automatically adjust the number of local models and their locality to account for the nonlinearities of the estimated function.

Locally Weighted Projection Regression (LWPR) is an example of a local learning method which has been applied in many instances to learn the inverse dynamics equation for control [3],[6],[7] due to its ability to achieve online, incremental learning by employing simple local linear models. Local dimensionality reduction is also achieved, making the algorithm capable of control of highly redundant systems [3]. Recent studies comparing these learning methods [8] show that while SVR and GPR can potentially yield higher accuracy, LWPR is better in terms of computational cost and is thus highly suitable for real time learning. These features make this approach very promising for on-line learning of robot inverse dynamics, as the approach can be used to model the relationship between the robot kinematics and torques without exact knowledge of the model structure, and can be used to adapt on-line to changes in the structure, such as changes in mass due to picking up an unknown load.

For humanoid and other redundant robots, a key issue is also redundancy resolution. Highly redundant systems typically have more degrees of freedom than are demanded by

978-1-4244-7107-2/10/$26.00

the task, the controller should be able to allocate degrees of freedom appropriately, and possibly allow the system to perform secondary tasks. For learning the inverse dynamics, redundant degrees of freedom pose an additional challenge, as there is no longer a unique mapping between the input and the output variables, as an infinite combination of input variables can achieve the desired output. This paper focuses on demonstrating the ability of LWPR to learn the inverse dynamics relationship for redundant systems. We propose a redundancy resolution mechanism based on minimizing the torque output as first introduced by Hollerback and Suh [9] for model based control. Unlike [9], we propose an approach which can be implemented without a-priori knowledge of the dynamic model of the manipulator through the use of LWPR. The remainder of this paper is organized as follows. In Section II, the dynamic model of a rigid body manipulator and model based control strategies are reviewed. Section III overviews the LWPR algorithm and learning inverse dynamics. Section IV presents the simulations and results. Finally, in Section V, the conclusions and next steps are discussed.

II. Background

A. Manipulator Dynamics

The dynamic equation of a manipulator characterizes the relationship between its motion (position, velocity and acceleration) and the forces that cause these motions. The closed-form solution to this relationship is obtained through the Lagrangian equation [10] and results in

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

where $\mathbf{q}$ is the nx1 vector of generalized coordinates consisting of the n joint angles for an n-degree of freedom (DOF) manipulator, $\mathbf{M}(\mathbf{q})$ is the nxn inertia matrix, which is symmetric and positive definite for all values of $\mathbf{q}$, $\mathbf{C}(\mathbf{q}, \dot{\mathbf{q}})$ is the nx1 centripetal and Coriolis force vector, $\mathbf{G}(\mathbf{q})$ is the nx1 gravity loading vector and $\boldsymbol{\tau}$ is the nx1 torque vector. Equation (1) represents the dynamic structure of the manipulator, but does not include additional torque components caused by friction, backlash and actuator dynamics. If accounted for, these components are modeled as additional terms in (1).

B. Task Space Dynamics

Manipulation tasks are typically described in terms of the trajectory of the end effector, whereas the actuator control signals to achieve this trajectory are specified in terms of the generalized joint coordinates. From this point of view, two approaches to the control of robot manipulators exist: joint space and task space techniques. As the manipulator equation in (1) is derived in terms of the generalized joint coordinates $\mathbf{q}$ and its derivatives, it can be applied to achieve joint space control techniques. To achieve task space control, the joint space manipulator equation in (1) can be rewritten as a function of the mx1 vectors of task space position $\mathbf{x}$, velocity $\dot{\mathbf{x}}$ and acceleration $\ddot{\mathbf{x}}$ as follows

$$\boldsymbol{\Lambda}(\mathbf{x})\ddot{\mathbf{x}} + \boldsymbol{\mu}(\mathbf{x}, \dot{\mathbf{x}}) + \mathbf{p}(\mathbf{x}) = \boldsymbol{\Gamma} \tag{2}$$

where

$$\boldsymbol{\Lambda}(x) = \boldsymbol{J}^{-\boldsymbol{T}}(q)\boldsymbol{M}(q)\boldsymbol{J}^{-\mathbf{1}}(q) \tag{3}$$

$$\boldsymbol{\mu}(x, \dot{x}) = \boldsymbol{J}^{-\boldsymbol{T}}(q)\boldsymbol{C}(q, \dot{q})\boldsymbol{J}^{-\mathbf{1}}(q) - \boldsymbol{\Lambda}(x)\boldsymbol{J}(q)\dot{q} \tag{4}$$

$$\boldsymbol{p}(x) = \boldsymbol{J}^{-\boldsymbol{T}}(q)\boldsymbol{G}(q) \tag{5}$$

where $\mathbf{q}$ is the nx1 vector of generalized manipulator coordinates, $\boldsymbol{\Lambda}$ is the nxn task space inertia matrix, $\boldsymbol{\mu}$ is the nx1 task space Coriolis/centripetal vector, $\mathbf{p}$ is the nx1 task space gravity vector, $\boldsymbol{\Gamma}$ is the nx1 vector of forces and torques in the task space and $\mathbf{J}(\mathbf{q})$ is the 6xn Jacobian matrix, defined by the differential kinematic equation relating task space and joint space velocities

$$\dot{\boldsymbol{x}} = \boldsymbol{J}(q)\dot{\boldsymbol{q}} \tag{6}$$

and its derivative relating the corresponding accelerations

$$\ddot{\boldsymbol{x}} = \boldsymbol{J}(q)\ddot{\boldsymbol{q}} + \dot{\boldsymbol{J}}(q)\dot{\boldsymbol{q}} \tag{7}$$

C. Computed Torque Control

Model-based controllers, or inverse dynamic controllers, are a broad class of controllers which apply the joint space dynamic equation (1) or the task space equation (2) to cancel the nonlinear and coupling effects of the manipulator. A common example of this is the computed torque control approach [11],[10] in which the control signal $\mathbf{u}$ is composed of the computed torque signal, $\mathbf{u_{CT}}$, which is set to the torque determined directly from the inverse of the dynamic equations (1) or (2). This term globally linearizes and decouples the system, and thus a linear controller can be applied for the feedback term, $\mathbf{u_{FB}}$, which stabilizes the system and provides disturbance rejection. Typically a PD scheme is used such that

$$\mathbf{u_{FB}} = \mathbf{k_p}\mathbf{e} + \mathbf{k_d}\dot{\mathbf{e}} \tag{8}$$

where $\mathbf{k_p}$ and $\mathbf{k_d}$ are proportional and derivative gain matrices, and $\mathbf{e} = \mathbf{x_d} - \mathbf{x}$ is the tracking error for task space control or $\mathbf{e} = \mathbf{q_d} - \mathbf{q}$ for joint space control. Thus, the overall control signal $\mathbf{u}$ is given by

$$\mathbf{u} = \mathbf{u_{FB}} + \mathbf{u_{CT}} \tag{9}$$

Desirable performance of the computed torque approach is contingent upon the assumption that the values of the parameters in the dynamic model (1),(2) match the actual parameters of the physical system. Otherwise, imperfect compensation of the nonlinearities and coupling occurs.

D. Task Space Control

Manipulator control requires the transformation of end effector motion, $\mathbf{x}$ to the corresponding joint motion, $\mathbf{q}$ through the solution of the inverse kinematics problem. In the case of joint space control, this is often done through iterative numerical methods, or in closed form methods involving analytical geometry [12] or though differential kinematics (6),(7) [13]. In such cases, desired trajectories are specified in task space and then converted to the equivalent motion in joint space. Joint

angles are then used in as the feedback signal. With task space control, Cartesian positions of the end effector are fed back in the control loop, and the inverse kinematics problem is solved directly in the control loop through the use of differential kinematics [14],[15],[16].

For redundant systems (i.e. systems with $n > m$), redundancy resolution at the velocity level is achieved by rearranging (6) to solve for $\dot{\mathbf{x}}$ in the following

$$\dot{\mathbf{q}}_\mathbf{d} = \mathbf{J}^\dagger \dot{\mathbf{x}}_\mathbf{d} + (\mathbf{I} - \mathbf{J}^\dagger \mathbf{J})\xi_\mathbf{1} \tag{10}$$

where $\mathbf{J}^\dagger$ is a pseudo-inverse of the Jacobian $\mathbf{J}$ and $\boldsymbol{\xi}_1$ is an arbitrary vector which controls the desired velocity behavior in the null space. At the acceleration level, (7) is rearranged to solve for $\ddot{\mathbf{x}}$ in the following

$$\ddot{\mathbf{q}}_\mathbf{d} = \mathbf{J}^\dagger(\ddot{\mathbf{x}}_\mathbf{d} - \dot{\mathbf{J}}\dot{\mathbf{q}}) + (\mathbf{I} - \mathbf{J}^\dagger \mathbf{J})\boldsymbol{\xi}_\mathbf{2} \tag{11}$$

where $\boldsymbol{\xi}_2$, similar to $\boldsymbol{\xi}_1$, is an arbitrary vector which controls the desired velocity behavior in the null space. By equating (11) to the analytical time derivative of (10) [16], $\boldsymbol{\xi}_\mathbf{2}$ can be calculated as

$$\boldsymbol{\xi}_\mathbf{2} = \dot{\mathbf{J}}^\dagger \mathbf{J}(\dot{\mathbf{q}} - \boldsymbol{\xi}_\mathbf{1}) + \dot{\boldsymbol{\xi}}_\mathbf{1} \tag{12}$$

Liegeois [17] determined that a position-dependant cost function, $g(\mathbf{q})$ can be minimized by equating $\boldsymbol{\xi}_1$ to the gradient of g in the following

$$\boldsymbol{\xi}_\mathbf{1} = k\frac{\partial g}{\partial q} \tag{13}$$

Many task space control schemes have been derived based on the task space dynamic equation in (2) [14],[16]. One particular class of these controllers is the resolved acceleration type [15], which account for the desired acceleration of the manipulator by using a control signal given by

$$\ddot{\mathbf{x}}_\mathbf{r} = \ddot{\mathbf{x}}_\mathbf{d} + \mathbf{K}_\mathbf{d}(\dot{\mathbf{x}}_\mathbf{d} - \dot{\mathbf{x}}) + \mathbf{K}_\mathbf{p}(\mathbf{x}_\mathbf{d} - \mathbf{x}) \tag{14}$$

where $\ddot{\mathbf{x}}_\mathbf{r}$ is the reference acceleration, $\ddot{\mathbf{x}}_\mathbf{d}$, $\dot{\mathbf{x}}_\mathbf{d}$ and $\mathbf{x}_\mathbf{d}$ are the desired task space accelerations, velocities and positions, and $\mathbf{K}_\mathbf{p}$ and $\mathbf{K}_\mathbf{d}$ are PD feedback gain matrices.

III. Learning Inverse Dynamics

While the computed torque control approach requires accurate knowledge of the structure of the dynamic model of the manipulator, the learning approach requires no a-priori knowledge of the dynamic model. Instead, the model can be obtained directly using measured data [1]. This allows unknown or typically unmodeled nonlinearities such as friction and backlash to be accounted for. In order to be practical for manipulator control, learning algorithms must process continuous streams of large sets of training data to update the model and predict outputs fast enough for real-time control. Locally Weighted Projection Regression (LWPR) achieves these objectives though the use of various techniques in nonparametric statistics [18].

A. Locally Weighted Projection Regression

The main idea behind the application of LWPR is to approximate the nonlinear inverse dynamics equation (1) with a set of piecewise local linear models based on the training data that the algorithm receives. Formally stated, this approach assumes a standard regression model of the form

$$\mathbf{y} = \mathbf{f}(\mathbf{X}) + \boldsymbol{\varepsilon} \tag{15}$$

where $\mathbf{X}$ is the input vector, $\mathbf{y}$ the output vector, and $\boldsymbol{\varepsilon}$ a zero-mean random noise term. Given a data point $\mathbf{X_c}$, a subset of data close to $\mathbf{X_c}$, with the appropriately chosen measure of closeness, or region of validity, a linear model can be fit to the subset of data such that

$$\mathbf{y} = \boldsymbol{\beta}^\mathbf{T}\mathbf{X} + \boldsymbol{\varepsilon} \tag{16}$$

where $\boldsymbol{\beta}$ are the set of parameters of the hyperplane that describe $\mathbf{y}$. The region of validity, termed the receptive field [3] is given by

$$w_k = exp(-\frac{1}{2}(\mathbf{X} - \mathbf{X_c})^\mathbf{T}\mathbf{D_k}(\mathbf{X} - \mathbf{X_c})) \tag{17}$$

where w_k determines the weight of the kth local linear model, $\mathbf{X_c}$ is the centre of the kth linear model, $\mathbf{D_k}$ corresponds to a positive semidefinite distance metric which determines the size of the receptive field. Given a query point $\mathbf{X}$, LWPR calculates a predicted output

$$\hat{\mathbf{y}}(X) = \sum_{k=1}^{K} w_k \hat{\mathbf{y}}_k / \sum_{k=1}^{K} w_k \tag{18}$$

where K is the number of linear models, $\hat{\mathbf{y}}_k$ is the prediction of the kth local linear model given by (16) which is weighed by w_k associated with its receptive field. Thus, the prediction $\hat{\mathbf{y}}(\mathbf{X})$ is the weighted sum of all the predictions of the local models, where the models having receptive fields centered closest to the query point are most significant to the prediction.

1) Partial Least Squares: Determining the set of parameters, β of the hyperplane is time consuming in the presence of high-dimensional input data, which is a characteristic of large numbers of DOF in redundant robotic systems. LWPR assumes that the data can be characterized by local low-dimension distributions, and attempts to reduce the dimensionality of the input space $\mathbf{X}$ by searching for the latent variables which represent the fundamental relationships between the input and output data. In order to achieve this, Partial Least Squares regression (PLS) is used. PLS fits linear models using a set of univariate regressions along selected projections in input space which are chosen according to the correlation between input and output data [18]. This ensures that irrelevant input dimensions are excluded from the data set, meaning that the input data $\mathbf{X}$ and output data $\mathbf{y}$ are reduced to the residual sets $\mathbf{X_{res}}$ and $\mathbf{y_{res}}$, on which subsequent projections are performed. This ensures that the next direction of projection is orthogonal to the previous [18]. The mean squared error (MSE) of the prediction after each projection is tracked, and

projections are added until the change in MSE becomes smaller than a certain threshold.

2) Adjustment of Receptive Fields: The distance metric, $\mathbf{D_k}$, (17) which dictates the size and shape of its corresponding receptive field, can be learned for each local model through stochastic gradient descent given by

$$\mathbf{m^{n+1}} = \mathbf{m^n} - \boldsymbol{\alpha}\frac{\partial \mathbf{J_{cost}}}{\partial \mathbf{m}} \tag{19}$$

where $\boldsymbol{\alpha}$ is the learning rate for gradient descent, $\mathbf{D} = \mathbf{m^T m}$ and $\mathbf{J_{cost}}$ is a penalized leave-one-out cross-validation cost function which addresses the issue of over-fitting of the data [19]. Receptive fields are created if for a given training data point, no existing receptive field possesses a weight w_i (17) that is greater than a threshold value of w_{gen}, which is a tunable parameter. The closer w_{gen} is set to one, the more overlap there will be between local models. Conversely, if two local models produce a weight greater than a threshold w_{prune}, the model whose receptive field is smaller is pruned.

B. Learning in Joint Space

The problem of learning the inverse dynamics relationship in the joint space can be described as the map from joint positions, velocities and accelerations to torques

$$(\mathbf{q}, \dot{\mathbf{q}}, \ddot{\mathbf{q}}_\mathbf{d}) \mapsto \mathbf{u} \tag{20}$$

where $\mathbf{u}$ is the control signal in the form of an nx1 torque vector, and $\mathbf{q}$ is the nx1 vector of generalized coordinates. This means that the mapping has an input dimensionality of $3n$ and an output dimensionality of n. Redundancy is dealt with outside of the control loop in the generation of the desired joint space trajectory.

C. Learning in Task Space

In the task space, the inverse dynamics relationship relates task space position, velocity and accelerations to the joint space torques. However, In the case of redundant systems, the dimension of $\mathbf{u}$ is greater than that of $\mathbf{x}$, meaning that there is an infinite number of joint configurations $\mathbf{q}$ which can yield the same task position $\mathbf{x}$. Thus, for proper localization of learning, $\mathbf{q}$ must be included in the mapping for learning task space control. Learning the inverse dynamics in task space is equivalent to learning the mapping from

$$(\mathbf{q}, \dot{\mathbf{q}}, \ddot{\mathbf{x}}_\mathbf{d}) \mapsto \mathbf{u} \tag{21}$$

The addition of $\mathbf{q}$ to the mapping is critical to resolve the problem of non-convexity [20]. This property states that the motor commands $\mathbf{u}$ which achieve the same desired acceleration $\ddot{\mathbf{x}}_\mathbf{d}$ may not be associated with a unique configuration of the robot. Thus, global averaging over all the learned models will yield invalid solutions to the inverse dynamics problem. However, spatial localization of the learned task can be achieved within the vicinity of a particular pose by including $\mathbf{q}$ in the mapping (21). Thus, when LWPR performs global averaging over all its local linear models, the learned mapping forms a convex data set.

D. Dealing with Redundancy

In order to deal with redundant systems, i.e. systems in which $n > m$, a method of selecting a particular solution out of the infinite number of solutions must be devised.

With humanoid and anthropomorphic manipulators, redundancy resolution is often achieved by specifying a desired posture for the robot and defining the cost function (13) accordingly [16],[7],[20]. This approach to redundancy resolution is typically used when there are a large number of redundant degrees of freedom relative to the task. Other methods of redundancy resolution commonly used with model based controllers involve the minimization of energy or the avoidance of joint limits or obstacles [9]. In order to test the ability of LWPR to handle redundant manipulators, the criteria of minimum torque [9] was selected by defining the cost function as

$$g(q) = \boldsymbol{\tau^T \tau} \tag{22}$$

where $\boldsymbol{\tau}$ is the nx1 torque vector and $\boldsymbol{\tau}^T$ is its transpose, which are functions of q as seen in (1). As in [17], let the gradient of $g(q)$ be $\mathbf{H}$ in the following

$$\mathbf{H} = \mathbf{W}\frac{\partial \mathbf{g}}{\partial \mathbf{q}} \tag{23}$$

where $\mathbf{H}$ is a vector which points in the direction of the greatest rate of change of g and W is a weighting matrix. $\mathbf{H}$ is then input into the LWPR model along with the kinematic variables during training. Thus, the mapping to be learned is

$$(\boldsymbol{q}, \dot{\boldsymbol{q}}, \ddot{\boldsymbol{x}}_\boldsymbol{d}, \boldsymbol{H}) \mapsto \boldsymbol{u} \tag{24}$$

where the addition of $\mathbf{H}$ minimizes the cost function in (22).

IV. Simulation

In order to test the performance of LWPR, both redundant and non-redundant manipulators were simulated in Matlab using Peter Corke's Robotics Toolbox [21] and the open source Matlab LWPR code [3]. The simulations were carried out on a 3 DOF and 4 DOF manipulator. Figure 8 trajectories were used to test the trajectory following capabilities of the learned inverse dynamics controller. The overall trajectory is described by the task space equations

$$\begin{aligned} x_d &= 0.1sin(kt) + 0.1 \\ y_d &= 0.1cos(kt/2) + 0.1 \\ z_d &= 0 \end{aligned} \tag{25}$$

where $\mathbf{p_d} = [x_d \ y_d \ z_d]^T$ are the desired end effector task space positions and k is proportional to the frequency of the trajectory. A frequency of approximately 0.25 Hz ($k = 3$) was used to test the tracking capabilities of the controller. The LWPR controller was trained for 60s while tracking the 'figure 8' trajectory at 0.25Hz with a poorly tuned PD controller, after which training was stopped and an attempt to track the desired trajectory was made. The system was then allowed to continue to learn for an additional 120s. The durations of training were

determined by observing the mean squared error (MSE) of the predicted torques from the LWPR controller. Training was generally stopped when the observed MSE had asymptotically decreased to a low value.

A. LWPR Tuning and Initialization

Although LWPR incorporates many algorithms which enable the system to automatically adjust its parameters for optimal performance, initial values of these parameters can significantly impact the convergence of the predictions. The initial value for the distance metric $\mathbf{D}$ (17) dictates how large a receptive field is upon initialization. Too small a value of $\mathbf{D}$ (corresponding to large receptive fields) tends to delay convergence while a larger value of $\mathbf{D}$ results in overfitting of the data [3]. Next, the normalization constant, $norm_{in}$, is used to normalize the inputs of the system according to the expected range of each input dimension. Values for the expected range of each input were obtained during the training phase which will be described below. The initial value of the learning rate α(19), which is for the gradient descent process, determines the rate at which MSE of predictions converge. Too high a value leads to instability and too low a value leads to a slow convergence. Although this value is automatically updated, it was found that its initial value has a significant impact upon the initial rate of convergence of the algorithm. These parameters were generally tuned through a trial-and-error process which involved monitoring the MSE of the predicted values during the training phase. The initial performance of the LWPR controller is also highly dependant upon the data sets that are used to train the LWPR model. Because LWPR is a local learning approach, it must be trained locally in the region(s) of input space that the manipulator will be operating in. In order to train the model, a low-gain PD controller was used to track the desired trajectories while the LWPR model obtained training data by observing joint torques and the resulting end effector movement, as described by the mapping in (21).

B. Task Space

Figure 1 and Table I shows the tracking results for the 3 DOF case after the initial 60 seconds of training and the subsequent performance after an additional 120s of training. It can be seen that only after three minutes of learning, the performance of the LWPR controller approaches that of the resolved acceleration (RA) controller which assumes perfect knowledge of the dynamic model.

C. Task Space with Redundancy

In order to assess the effectiveness of the redundancy optimization, both the LWPR and RA (10-14) controller and were simulated on a 4 DOF robot using the torque minimization criteria in (23). Figure 2 and Table I illustrate that the initial 60s of training was insufficient to produce accurate tracking results, as the RMS error in tracking is much higher than the corresponding error of the 3 DOF case after the same amount of learning. However, after training for a total of 240 seconds,

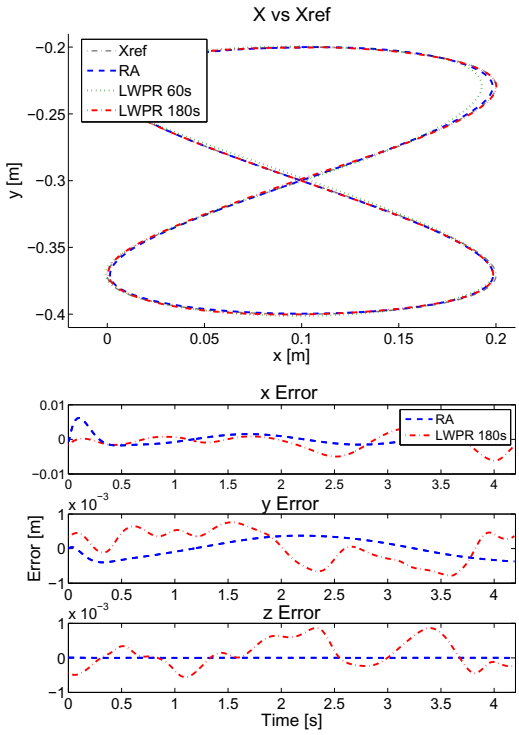

Fig. 1. Simulation 1 - LWPR Tracking Error for 3 DOF after 180s

the LWPR model for the redundant system produces results closer to that of the 3 DOF with 180 seconds of training. The longer training times for the 4 DOF system are expected since the dimension of the mapping to be learned is larger due to the additional joint and the gradient term for redundancy resolution, thus requiring more time for the algorithm to acquire training data and incrementally improve its predictions. The slightly larger error of the RA controller compared to the 3 DOF result is likely caused by the redundancy resolution in the null space interfering with the task space [16].

Figure 3 plots the cost function (22) for both the RA and LWPR controller after 240s of training. The LWPR controller does a reasonable job of producing the same torque optimization results as the RA controller as the major features of the graphs are similar and close in magnitude. However, due to the piecewise local linear models of LWPR, the controller changes with each prediction and hence the cost function produced by LWPR is not as smooth as with RA.

TABLE I
RMS TRACKING ERROR OF CONTROLLERS (MM)

	3 DOF	4 DOF
Resolved Acceleration	1.79	1.86
LWPR, 60s Training	2.83	14.36
LWPR, 180s Training	1.90	2.87
LWPR, 240s Training	\	1.97

V. CONCLUSION AND FUTURE WORK

LWPR was shown to be a viable method of estimating the nonlinear inverse dynamics relationship of robot manipulators,

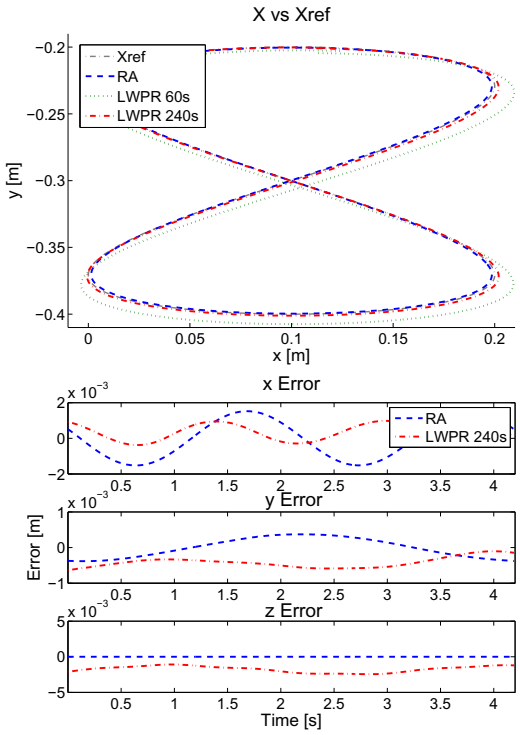

Fig. 2. Simulation 2 - LWPR Tracking Error for 4 DOF after 240s

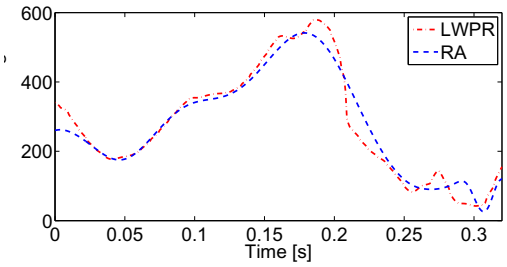

Fig. 3. Cost Function - LWPR vs RA for 4 DOF

in order to be used as the computed torque term of an inverse dynamics controller. This approach was able to handle learning the mapping of the task space control scheme with the addition of redundancy, proving that it is a viable approach to control the newer generation of humanoid, assistive and entertainment robots which possess high DOFs.

The approach of learning inverse dynamics poses several advantages over the classical modeling approach, by reducing sensitivity to modeling and modeling structure errors, and allowing for online adaptation to changes in the model structure or parameters. The main issues with this approach lie in the need for significant tuning and poor performance in untrained areas. The former issue is not as constraining, as the algorithm already deals with automatic tuning of critical parameters which yield satisfactory performance in most cases. However, the latter issue must be addressed in order for this approach to be viable in the human environment, where safety is of utmost importance.

Future studies will address the need for evaluation on real-world robots by testing the system's ability to cope with uncertainties such as friction, sensor noise, payload changes and perturbations to the plant model. These results will be compared to classical model-based approaches. Secondly, methods of incorporating a-priori knowledge of the dynamic model of the system will be investigated.

REFERENCES

[1] D. Nguyen-Tuong, B. Scholkopf, and J. Peters, "Sparse online model learning for robot control with support vector regression," pp. 3121–3126, Oct. 2009.

[2] K. Ayusawa, G. Venture, and Y. Nakamura, "Identification of humanoid robots dynamics using floating-base motion dynamics," pp. 2854–2859, Sept. 2008.

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

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

[5] D. Nguyen-Tuong, B. Scholkopf, and J. Peters, "Sparse online model learning for robot control with support vector regression," pp. 3121–3126, Oct. 2009.

[6] C. Atkeson, "Using locally weighted regression for robot learning," pp. 958–963 vol.2, Apr 1991.

[7] A. D'Souza, S. Vijayakumar, and S. Schaal, "Learning inverse kinematics," pp. 298–303 vol.1, 2001.

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

[9] J. Hollerbach and K. Suh, "Redundancy resolution of manipulators through torque optimization," *Robotics and Automation, IEEE Journal of*, vol. 3, no. 4, pp. 308 –316, august 1987.

[10] L. Sciavicco and B. Scicliano, *Modelling and Control of Robot Manipulators*. Springer, 1996.

[11] J. Craig, P. Hsu, and S. Sastry, "Adaptive control of mechanical manipulators," pp. 190–195, Apr 1986.

[12] T. Asfour and R. Dillmann, "Human-like motion of a humanoid robot arm based on a closed-form solution of the inverse kinematics problem," pp. 1407–1412 vol.2, Oct. 2003.

[13] G. Tevatia and S. Schaal, "Inverse kinematics for humanoid robots," pp. 294–299 vol.1, 2000.

[14] O. Khatib, "A unified approach for motion and force control of robot manipulators: The operational space formulation," *Robotics and Automation, IEEE Journal of*, vol. 3, no. 1, pp. 43–53, February 1987.

[15] F. Caccavale, C. Natale, B. Siciliano, and L. Villani, "Resolved-acceleration control of robot manipulators: A critical review with experiments," *Robotica*, vol. 16, no. 5, pp. 565–573, 1998.

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

[17] A. Liegeois, "Automatic supervisory control of the configuration and behavior of multibody mechanisms," *Systems, Man and Cybernetics, IEEE Transactions on*, vol. 7, no. 12, pp. 868 –871, dec. 1977.

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

[19] S. Schaal and C. G. Atkeson, "Constructive incremental learning from only local information," *Neural Computation*, vol. 10, no. 8, pp. 2047–2084, 1998.

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

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