2013 IEEE International Conference on Robotics and Automation (ICRA)
Karlsruhe, Germany, May 6-10, 2013

Locally Weighted Learning Model Predictive Control for Nonlinear and Time Varying Dynamics

Christopher Lehnert and Gordon Wyeth

Abstract— This paper proposes an online learning control system that uses the strategy of Model Predictive Control (MPC) in a model based locally weighted learning framework. The new approach, named Locally Weighted Learning Model Predictive Control (LWL-MPC), is proposed as a solution to learn to control robotic systems with nonlinear and time varying dynamics. This paper demonstrates the capability of LWL-MPC to perform online learning while controlling the joint trajectories of a low cost, three degree of freedom elastic joint robot. The learning performance is investigated in both an initial learning phase, and when the system dynamics change due to a heavy object added to the tool point. The experiment on the real elastic joint robot is presented and LWL-MPC is shown to successfully learn to control the system with and without the object. The results highlight the capability of the learning control system to accommodate the lack of mechanical consistency and linearity in a low cost robot arm.

I. Introduction

Giving a robot the ability to learn its own control system allows a robot to account for its own nonlinear and changing dynamics experienced in the real world. Elastic Joint Robots (EJR) can experience large changes in dynamics, causing controller instability when they interact with their environment, such as manipulating heavy objects. EJR's are generally difficult to learn to control due to their elastic properties preventing standard model learning techniques from being used, such as learning computed torque control. Further challenges are found when low cost implementations are introduced, problems such as stiction and deadzones from geared DC motors, and nonlinear forces from non-ideal springs and bearings.

Model based learning techniques are becoming increasingly popular in robotics, as they have shown to be a useful technique for learning dynamic models for control of complex and nonlinear systems. The majority of model learning for robot control techniques has been applied to rigid body dynamics using a computed torque control method. However, EJR's are not strictly rigid body systems, which makes learning inverse dynamics increasingly difficult. De Luca and Book outline that feedback linearization via computed torque control is not straightforward for EJR's and in some cases does not satisfy conditions for exact linearization [1]. This is further supported in [2] where it is shown that directly learning the inverse dynamics fails to produce a stable control method for a single Degree of Freedom (DoF) elastic joint. Therefore, more advanced control methods are required in order to use model learning techniques to control EJR's and other systems that cannot be controlled using a direct inverse dynamics model.

The authors are with the CyPhy Laboratory, School of Electrical Engineering and Computer Science, Queensland University of Technology, Australia (c.lehnert@qut.edu.au, gordon.wyeth@qut.edu.au).

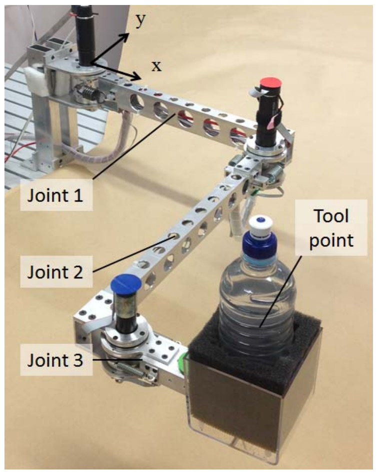


Figure 1. Three DoF Elastic Joint Robot arm consisting of three Series Elastic Actuators (SEA) connected in a planar configuration.

Model Predictive Control is a successful method that has been applied to a variety of industrial and real world robotic systems. MPC uses a forward dynamics model combined with an optimization method to produce an optimal feedback control law for each control cycle. The advantages of MPC are its ability to explicitly handle system constraints and delays while being computationally feasible as an online control system. As MPC is based on a predictive forward model it can be easily integrated into a model based learning technique, combining the advantages of its ability to learn its own model while computing an optimal feedback control law.

This paper describes the method of LWL-MPC which uses locally weighted learning in combination with model predictive control to produce an incremental and online system which is demonstrated to learn and control a three DoF elastic joint robot. Section II gives a background of similar approaches for robot control, and an overview of model learning techniques for robot control. The locally weighted learning technique of Receptive Field Weighted Regression (RFWR) is presented in Section III. Learning linear MPC is then outlined for a time varying discrete system. The linear method is extended to the nonlinear

LWL-MPC approach in Section IV. The LWL-MPC method is then evaluated experimentally on a real EJR in Section V followed by the results in Section VI. Lastly a discussion of the proposed system and results are given in Section VII.

II. Background

Model learning techniques have been previously used with MPC. Most techniques learn a global nonlinear function which is combined with nonlinear MPC to control the system. Different model learning methods have been used, such as Fuzzy Models in Fuzzy Multi-Model predictive control (FMMPC) [3]. This method was demonstrated on a thermoplastic injection moulding plant and had difficulty learning to control due to initialization problems of the fuzzy models. Another approach used Locally Weighted Projection Regression (LWPR) to approximate a nonlinear function and was used for predictive force control in assistive beating heart surgery [4].

Most of these systems have large computational costs in order to find the solution to the nonlinear MPC problem. In our proposed technique this cost is reduced by computing multiple local linear MPC's inside a locally weighted framework. Local model techniques have also been previously used in the adaptive control literature [5], which uses radial basis functions to separate local models. These techniques, as with most adaptive control strategies, rely on an explicit definition of the system model and a stringent switching strategy.

A general framework for a robust and stable Learning Based Model Predictive Control strategy has also been developed [6]. The method relies on a good initial estimate of the model in order to estimate stability margins. This work has been applied to a real quadrotor, where a learning rule is used to estimate un-modelled dynamics [7].

Other optimal control techniques have been combined with model learning methods such as Iterative Linear Quadratic Gaussian with Learned Dynamics (ILQG-LD) [8]. ILQG-LD iteratively computes local optimal control policies for a specified end goal state based on a learned dynamics model using LWPR. ILQG-LD has been applied to control a single Variable Stiffness Actuator (VSA) [9] and on multiple simulated robotic arms. The disadvantage with this method is the computational cost of iteratively computing the path and control law to the end goal until convergence is found.

Another model learning method Sparse Incremental Gaussian Process Regression (SI-GPR) was developed and demonstrated to learn and adapt to changing dynamics on a 7-DoF rigid body robot arm [10]. This method used a learned dynamics model as a feed forward term and used a PD feedback controller to stabilize the system.

III. Model Learning for Robot Control

Model learning algorithms approximate functions of different model types, such as forward or inverse models from actual system data. The model is typically generated using a supervised nonparametric regression method. The advantage of using model learning techniques is that the nonlinear effects of the system associated with training data are inherently built into the nonparametric model. A more in-depth review of model learning techniques can be found within recent surveys [11],[12]. One popular model learning method is Locally Weighted Regression (LWR), which reduces the computational cost by approximating nonlinear functions with multiple locally linear models.

A. Locally Weighted Regression

There are a few variants of the LWR algorithm, but the body of the work can be found in [13]. An iterative version has been developed titled Receptive Field Weighted Regression (RFWR) [14] which incrementally adds and updates local linear models to produce a computationally efficient online regression technique. A Partial Least Squares (PLS) solution was then incorporated as a dimensionality reduction technique for high dimensional models in LWPR [15]. In this paper the RFWR algorithm was investigated as the PLS solution in LWPR makes the local models incompatible with model predictive control in its current form. The RFWR estimates a nonlinear function of an output vector $\mathbf{s}$ as

$$\mathbf{s} = f(\mathbf{z}) + \boldsymbol{\varepsilon}, \tag{1}$$

where $\mathbf{z}$ is the input vector, and $\boldsymbol{\varepsilon}$ is a zero mean noise term. The RFWR algorithm estimates this nonlinear function by introducing locally linear models across the input space forming an output prediction based on a weighted average of local predictions. The weights of an input, w_i, are computed for each local model using a kernel function. Typically the Gaussian kernel

$$w_i(\mathbf{z}) = \exp\left(-0.5(\mathbf{z}-\mathbf{c}_i)^T \mathbf{D}_i (\mathbf{z}-\mathbf{c}_i)\right), \tag{2}$$

is used, where $\mathbf{z}$ is the input vector, $\mathbf{c}_i$ is the centre of the kernel of the i^{th} local model, and $\mathbf{D}_i$ determines the shape and size of the Gaussian kernel. The function is then approximated using a Weighted Recursive Least Squares (WRLS) rule for each locally linear model. The estimated function is approximated by a weighted average of each local model in the following form

$$f(\mathbf{z}) = \frac{1}{\mathbf{W}} \sum_{i=1}^{I} w_i(\mathbf{z}) \Phi_i(\mathbf{z}), \quad \mathbf{W} := \sum_{i=1}^{I} w_i(\mathbf{z}), \tag{3}$$

where $\mathbf{W}$ is the sum of the all weights acting as the normalising factor, and Φ_i is the local linear model in the form

$$\Phi_i(\mathbf{z}) = \boldsymbol{\beta}_i^T (\mathbf{z} - \bar{\mathbf{z}}) + \boldsymbol{\beta}_i^0, \tag{4}$$

where $\boldsymbol{\beta}_i^T$ is the gradient of the linear model and $\boldsymbol{\beta}_i^0$ is the mean. The shapes of the kernels for each model are then updated to minimise the prediction error using a gradient descent update rule. An advantage of using a recursive algorithm for model learning is that it has the ability to forget previous sample points. This allows the learning algorithm to adapt to time varying parameters in the model. This is achieved by introducing a forgetting factor, $\lambda \in [0,1]$, in the WRLS which down-weights previous training data. The forgetting factor acts as an exponential decay with time constant approximate to $1/1-\lambda$. Choosing the value of the forgetting factor for different time varying systems is still an open problem and more detail about the effect of the forgetting factor can be found in [16].

IV. Locally Weighted Learning Model Predictive Control

Previous work has been done which introduces the idea of LWL-MPC and applies the method to an invariant system [17]. An extension of the approach formulated in previous work is outlined in this section in order incorporate time variant systems and overall performance improvements. This section firstly outlines the method of linear MPC for a state space model. It is then shown how this approach can be incorporated into locally weighted regression for nonlinear systems. A more in depth overview of MPC techniques can be found in [18].

A. Learning Linear Model Predictive Control

A linear time variant system is modelled as a discrete state space system defined as

$$\begin{aligned} \mathbf{x}_{k+1} &= \mathbf{A}(k)\mathbf{x}_k + \mathbf{B}(k)\mathbf{u}_k + \mathbf{K}\mathbf{e}_k \\ \mathbf{y}_k &= \mathbf{C}(k)\mathbf{x}_k + \mathbf{D}(k)\mathbf{u}_k + \mathbf{e}_k \end{aligned}, \qquad (5)$$

where $\mathbf{u}_k \in \mathbb{R}^q$, $\mathbf{y}_k \in \mathbb{R}^l$, and $\mathbf{x}_k \in \mathbb{R}^m$, are vectors of input, output and state, respectively. The matrices $\mathbf{A}(k) \in \mathbb{R}^{m\times m}$, $\mathbf{B}(k) \in \mathbb{R}^{m\times q}$, $\mathbf{C}(k) \in \mathbb{R}^{l\times m}$, and $\mathbf{D}(k) \in \mathbb{R}^{l\times q}$ are time varying discrete state space system matrices and $\mathbf{K} \in \mathbb{R}^{l\times l}$ represents the Kalman gain of the process white noise $\mathbf{e}_k$. MPC uses the future outputs and inputs defined as

$$\mathbf{y}_k^f := \begin{bmatrix} \mathbf{y}_k & \mathbf{y}_{k+1} & \cdots & \mathbf{y}_{k+n} \end{bmatrix}^T, \text{ and } \mathbf{u}_k^f := \begin{bmatrix} \mathbf{u}_k & \mathbf{u}_{k+1} & \cdots & \mathbf{u}_{k+n} \end{bmatrix}^T,$$

where n denotes the horizon window from the current time k to n steps later. The past inputs, $\mathbf{u}_k^p$ and outputs, $\mathbf{y}_k^p$ can be similarly defined representing a history of inputs and outputs from k–n to the current time step k.

The future outputs with respect to the system matrices can then be described as a set of input output equations from k to n in the form of

$$\mathbf{y}_k^f = \mathbf{\Gamma}_n \mathbf{x}_k + \mathbf{H}_n^d \mathbf{u}_k^f + \mathbf{H}_n^s \mathbf{e}_k^f, \qquad (6)$$

where

$$\mathbf{\Gamma}_n = \begin{bmatrix} \mathbf{C} & \mathbf{CA} & \cdots & \mathbf{CA}^n \end{bmatrix}^T,$$

$$\mathbf{H}_n^d = \begin{bmatrix} \mathbf{D} & 0 & \cdots & 0 \\ \mathbf{CB} & \mathbf{D} & \cdots & 0 \\ \vdots & \vdots & \ddots & 0 \\ \mathbf{CA}^{n-1}\mathbf{B} & \mathbf{CA}^{n-2}\mathbf{B} & \cdots & \mathbf{D} \end{bmatrix}, \ \mathbf{H}_n^s = \begin{bmatrix} \mathbf{K} & 0 & \cdots & 0 \\ \mathbf{CB} & \mathbf{K} & \cdots & 0 \\ \vdots & \vdots & \ddots & 0 \\ \mathbf{CA}^{n-1}\mathbf{B} & \mathbf{CA}^{n-2}\mathbf{B} & \cdots & \mathbf{K} \end{bmatrix}.$$

For this system, $\mathbf{\Gamma}_n \in \mathbb{R}^{(l\times n)\times m}$ is the extended observability matrix, $\mathbf{H}_n^d \in \mathbb{R}^{(l\times n)\times(q\times n)}$ and $\mathbf{H}_n^s \in \mathbb{R}^{(l\times n)\times(q\times n)}$ are lower block triangular Toeplitz matrix. This method is based on the assumption that the system is fully observable and measurements of the output and state of the system can be measured. Under this assumption the linear model can simply be estimated using a Least Squares algorithm. Furthermore, by buffering a history of samples from k–n to k the model can be computed recursively and online as each sample point is measured. An estimate of the output can then be formulated given the query point $[\mathbf{x}_q, \mathbf{u}_q^f]$

$$\hat{\mathbf{y}}_k^f = \hat{\mathbf{\Gamma}}_n \mathbf{x}_q + \hat{\mathbf{H}}_n^d \mathbf{u}_q^f. \qquad (7)$$

where $\hat{\mathbf{\Gamma}}_n$, and $\hat{\mathbf{H}}_n^d$ are the estimated model parameters.

Given this estimated linear model, MPC is formulated by solving the optimization problem that minimizes the control performance and control effort with respect to the model using the cost function

$$\underset{\mathbf{u}_k^f}{\arg\min}\, J = (\hat{\mathbf{y}}_k^f - \mathbf{y}_{des}^f)^T \mathbf{W}_y (\hat{\mathbf{y}}_k^f - \mathbf{y}_{des}^f) + \mathbf{u}_k^f \mathbf{W}_u \mathbf{u}_k^f, \qquad (8)$$

where, $\mathbf{y}_{des}^f$ is the sequence of future desired outputs, $\mathbf{W}_y \in \mathbb{R}^{(n\times l)\times(n\times l)}$ and $\mathbf{W}_u \in \mathbb{R}^{(n\times q)\times(n\times q)}$ are diagonal matrices of weights that balance the tradeoff between control performance and control effort. Substituting (7) into (8) and removing terms independent of the input (as later the cost function is minimised with respect to the input), the cost function in quadratic form becomes

$$J = \frac{1}{2}\mathbf{u}_k^{f^T} \mathbf{\Psi} \mathbf{u}_k^f + \mathbf{u}_k^{f^T} \mathbf{\Omega}, \qquad (9)$$

where

$$\mathbf{\Psi} := \hat{\mathbf{H}}_n^{d^T} \mathbf{W}_y \hat{\mathbf{H}}_n^d + \mathbf{W}_u, \text{ and } \mathbf{\Omega} := \hat{\mathbf{H}}_n^{d^T} \mathbf{W}_y (\hat{\mathbf{\Gamma}}_n \mathbf{x}_k - \mathbf{y}_{des}^f).$$

Equation (9) can be solved using a Quadratic Programming (QP) method subject to constraints, such as upper and lower bounds of inputs and outputs. For the unconstrained case the solution can be computed using linear algebra. Taking the derivative of the cost function, setting it to zero and rearranging with respect to the input gives

$$\mathbf{u}_k^f = -\mathbf{\Psi}^{-1}\mathbf{\Omega}, \qquad (10)$$

and substituting for system parameters becomes

$$\mathbf{u}_k^f = (\hat{\mathbf{H}}_n^{d^T} \mathbf{W}_y \hat{\mathbf{H}}_n^d + \mathbf{W}_u)^{-1} \hat{\mathbf{H}}_n^{d^T} \mathbf{W}_y (\mathbf{y}_{des}^f - \hat{\mathbf{\Gamma}}_n \mathbf{x}_k). \qquad (11)$$

A. Learning Non-linear Model Predictive Control

For non-linear systems the problem can be separated into multiple locally linear controllers by simultaneously learning linear forward models and compute a local control using (11) within each linear region.

Given the nonlinear and time varying discrete state space equation

$$\begin{aligned}\mathbf{x}_{k+1} &= f(\mathbf{x}_k,\mathbf{u}_k,k)\\ \mathbf{y}_k &= g(\mathbf{x}_k,\mathbf{u}_k,k)\end{aligned}, \tag{12}$$

RFWR can be used to approximate the nonlinear function of the output in the form of (3)

$$\hat{\mathbf{y}}_k^f = \frac{1}{\mathbf{W}}\sum_{i=1}^{I} w_i(\mathbf{x}_k,\mathbf{u}_k)\Pi_i(\mathbf{x}_k,\mathbf{u}_k^f), \tag{13}$$

where the linear model $\mathbf{\Phi}_i$ has been substituted for Π_i defined as

$$\Pi_i(\mathbf{x}_k,\mathbf{u}_k^f) = [\hat{\Gamma}_N^i\mathbf{x}_k + \mathbf{H}_N^i\mathbf{u}_k^f] .$$

Under the assumption that the system is locally linear the feedback control law can be computed from (11) to estimate a sequence of local control inputs $\hat{\mathbf{u}}_{k,i}^f$. The estimated next input $\hat{\mathbf{u}}_{k+1}^i$ is then taken out of this sequence and combined as a weighted average across all models producing an estimate of the next control input as

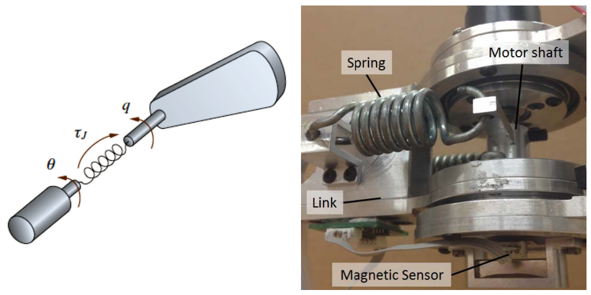
$$\hat{\mathbf{u}}_{k+1} = \frac{1}{\mathbf{W}}\sum_{i=1}^{I} w_i(\mathbf{x}_k,\mathbf{u}_k)\hat{\mathbf{u}}_{k+1}^i. \tag{14}$$

It was found that the performance of each local predictive controller is attributed to the selection of $\mathbf{W}_u$. The smaller the $\mathbf{W}_u$ term the more control effort is given. However, if the estimated parameters $\hat{\mathbf{\Gamma}}_n$, and $\hat{\mathbf{H}}_n^d$ have poor condition or are inaccurate then the inverse in (11) is poorly conditioned producing incorrect input predictions. It was empirically found that the kernel weight, w_i, can be used to improve the conditioning of the local controllers by inversely weighting the cost of control effort with the distance to the current local region. The weighted cost for each model is defined as

$$\mathbf{W}_u^i = \frac{\alpha}{w_i} I_n , \tag{15}$$

where α is a constant defining an initial weighted cost for control effort.

Using the RFWR algorithm the models can be learned online by using the past input, outputs and state which is delayed from k–n to k. The forgetting factor is then used to allow the system to approximate the time varying parameters of the nonlinear system. Every control step each local controller estimates a new input and is then averaged based on how close the local controller is to the current state and input of the system. The final LWL-MPC system is defined in Fig. 3.



(a) Concept Model of an SEA (b) SEA joint on robot arm

Figure 2. Diagram of Series Elastic Actuator (a) Concept model of SEA shows the motor angle θ is connected in series to the joint angle q through the spring element. (b) Shows the implementation of the SEA on the real robot arm, highlighting all the SEA elements.

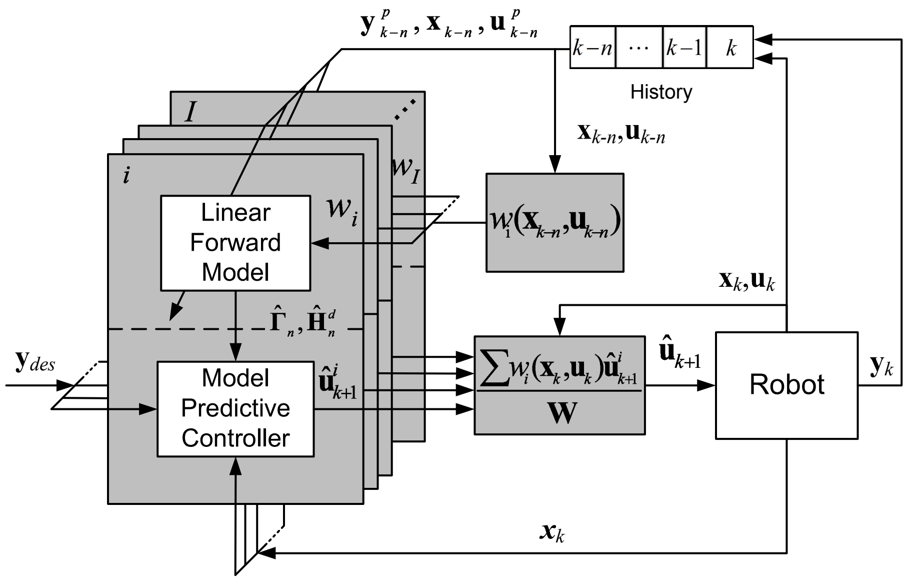

$\mathbf{y}_{k-n}^p, \mathbf{x}_{k-n}, \mathbf{u}_{k-n}^p$
$k-n$ ⋯ $k-1$ k
History
$\mathbf{x}_{k-n},\mathbf{u}_{k-n}$
I
w_I
i
Linear Forward Model
w_i
$w_i(\mathbf{x}_{k-n},\mathbf{u}_{k-n})$
$\mathbf{x}_k,\mathbf{u}_k$
$\hat{\Gamma}_n, \hat{\mathbf{H}}_n^d$
$\mathbf{y}_{des}$
Model Predictive Controller
$\hat{\mathbf{u}}_{k+1}^i$
$\frac{\sum w_i(\mathbf{x}_k,\mathbf{u}_k)\hat{\mathbf{u}}_{k+1}^i}{\mathbf{W}}$
$\hat{\mathbf{u}}_{k+1}$
Robot
$\mathbf{y}_k$
x_k

Figure 3. Illustration of LWL-MPC scheme. The system incorporates a linear forward model and a shared local controller. Inputs are computed by a weighted average of estimated inputs from each controller. The forward models are updated from a history of past inputs, states and outputs.

V. Experiments

The online learning and adaption of LWL-MPC was investigated by applying the system to the real three DoF EJR shown in Fig. 1. The aim of this experiment is to validate LWL-MPC's ability to learn significant nonlinear and time varying dynamics using only motor voltages as inputs and state feedback with no prior knowledge of the system. The greatest challenge in the experiment is to learn joint position control of the arm while the system experiences a significant change in dynamics by placing an object on the tool point of the robot arm shown in Fig. 1.

B. Experimental Setup and Procedure

The experiment can be separated into two main steps: the training phase which generates an initial model of the system; and the control phase where the robot is given a desired trajectory and performs online learning and control. For this experiment the state of the robot $\mathbf{x} := [\dot{\boldsymbol{\theta}} \quad \boldsymbol{\tau}_j \quad \mathbf{q} \quad \dot{\mathbf{q}}]$ has twelve dimensions, a vector of motor velocities $\dot{\boldsymbol{\theta}}$, joint torques $\boldsymbol{\tau}_j$, joint positions $\mathbf{q}$, and joint velocities $\dot{\mathbf{q}}$, for each DoF. The joint torques and joint positions are determined using a magnetic encoder (highlighted in Fig. 2) that measures the spring deflection of each joint, which can be converted to torques and added to the motor position to estimate joint position. The inputs to the system are motor voltages $\mathbf{u} := [v_1 \quad v_2 \quad v_3]$, for each joint and the output of the system are the joint positions $\mathbf{y} := [q_1 \quad q_2 \quad q_3]$.

The window size for this system was experimentally selected as 30, producing stable control and good performance for a small horizon window. Implementing the LWL-MPC system in hardware was achieved by separating the method into three SISO systems for each joint, where each joint is given the full state. This allowed the computation to be separated among three threads for each joint but further speedups could be achieved by implementing multiple threads for each local model and controller. With the current implementation of the algorithm a control rate of 100Hz was achieved on a 3.4GHz i7 based PC.

The learning rates and initial parameters for the RFWR algorithm were experimentally chosen producing stable learning while performing online updates. The forgetting factor, λ, for this task was empirically chosen such that good performance was achieved while adapting to the sudden change in dynamics.

The parameters chosen for the experiment are shown in Table I.

TABLE I
LWL-MPC Parameter List

Parameter	Value	Description
λ	0.9996	Forgetting factor
γ	5×10^{7}	Learning rate
$\mathbf{D}_0$	30	Initial Gaussian distance metric
α	1×10^{-3}	Initial weighted cost

For this experiment a desired trajectory of a figure of eight was chosen and computed using the inverse kinematics of the robot arm. The trajectory was then given to the LWL-MPC as joint positions. The figure of eight was executed with a four second cycle time in order to produce significant differences in nonlinear dynamics with and without the object.

1) Training Phase

A PD controller based on joint position feedback was chosen to produce the initial training data on the real robot arm. In other learning control methods the training phase typically involves performing motor babbling and can also be applied to this system. However, using a simple PD controller instead of motor babbling was selected for two reasons: firstly, it produces initial training data, where the state of the system should be close to the solution of the desired trajectory; and secondly, it produces a base line for comparing the final performance of the LWL-MPC system. Approximately halfway through the training phase an object is placed on the end effector evaluating the model free controller's ability to handle changes to system dynamics. For this experiment to test learning of changing dynamics the training data given to LWL-MPC for offline learning only involved sample points without the object.

2) Control Phase

After sufficient offline training the initialized LWL-MPC system is applied to the real robot arm. After 20 iterations of learning to control the initial system the object from the training phase is then placed on the tool point. 20 iterations of the figure of eight are then performed while adapting to the change in dynamics. Once this is achieved the object is removed and a final set of 20 iterations is performed in order to readapt to the initial system.

VI. Results

A. Training Results

The training phase was executed for a total of 20 iterations of the figure of eight, with and without the object, taking 80 seconds to complete. The training data generated a total of 2000 sample points, half were used for training. The offline training produced a total 14 local models for the initial system.

Figure 4 shows a plot of the resulting end effector position for the PD controller with and without an object. The results show that even for a model free controller the change in dynamics causes a change in control performance.

It was found that stable gains of the PD controller produced a large delay and increasing the gain the system became unstable. This is related to the elastic properties of each joint having complex poles (defining the resonance of the joint) which are close to the real axis. Increasing the gain or introducing integration to reduce the steady state error pushes the closed loop poles over to the real axis. Therefore, in order to achieve better performance further system analysis to identify the open loop complex poles for each joint would be required. After sufficient system identification is complete an advanced controller would then be required to cancel each complex pole. This generally is a complicated and time consuming task [19].

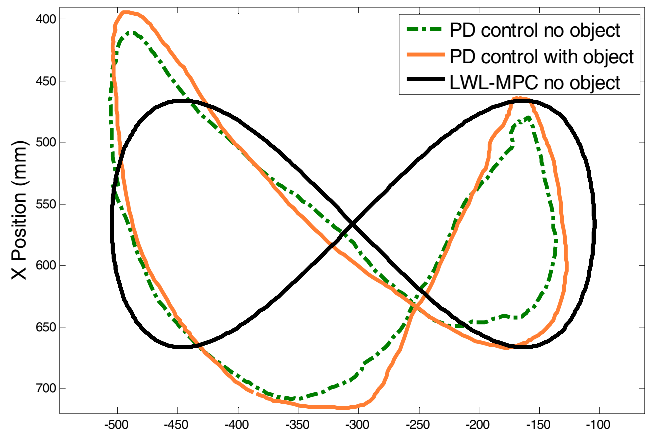
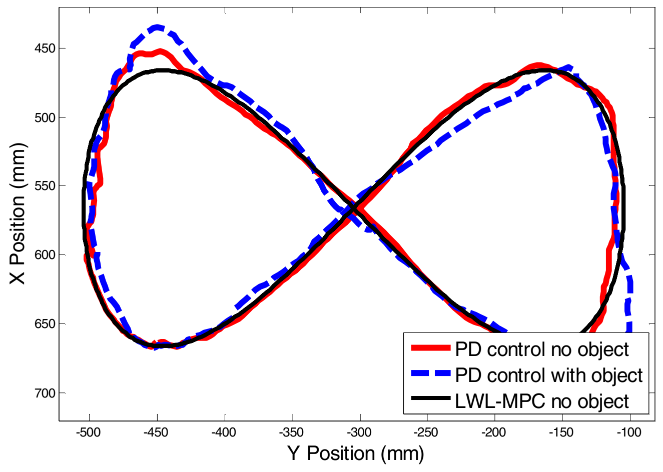


Figure 4. Actual end effector position for the PD controller with and without object versus the desired figure of eight for the training phase.

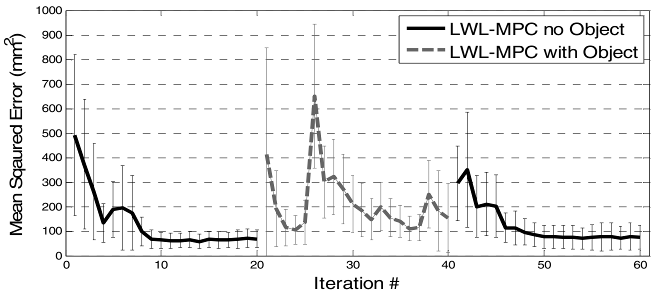
B. *Control Results*

The controller was executed for a total of 4 minutes which included a total of 60 iterations of the figure of eight. The performance of the controller was recorded during the experiment, including overhead video footage at 25 frames per second to produce a reasonable ground truth of the end effector. A total of 154 local models where produced to learn the robot arm while adapting to changing dynamics.

LWL-MPC successfully learned to control and adapt to the change in dynamics when the object was placed on the tool point and subsequently removed. Fig. 5a illustrates the performance of the controller during the last iterations after adapting to the dynamics with and without an object. This figure shows that the controller produces a significant improvement when compared to the PD controller shown in Fig. 4. Furthermore, it validates that the system can adapt and produce a similar control performance when a change in the systems dynamics occurs.



(a) End effector trajectory of arm using LWL-MPC



(b) Learning curve over the control phase

Figure 5. LWL-MPC control performance (a) Actual versus desired joint positions for last iteration of control phase with and without object (b) Plot of the mean squared error between the desired and actual end effector positions per iteration.

Fig. 5b shows the learning curves across the 60 iterations. It can be seen that when the system undergoes an abrupt change in dynamics the performance initially decreases. However, the model learning algorithm is able to adapt to the change, slowly improving the controller performance as

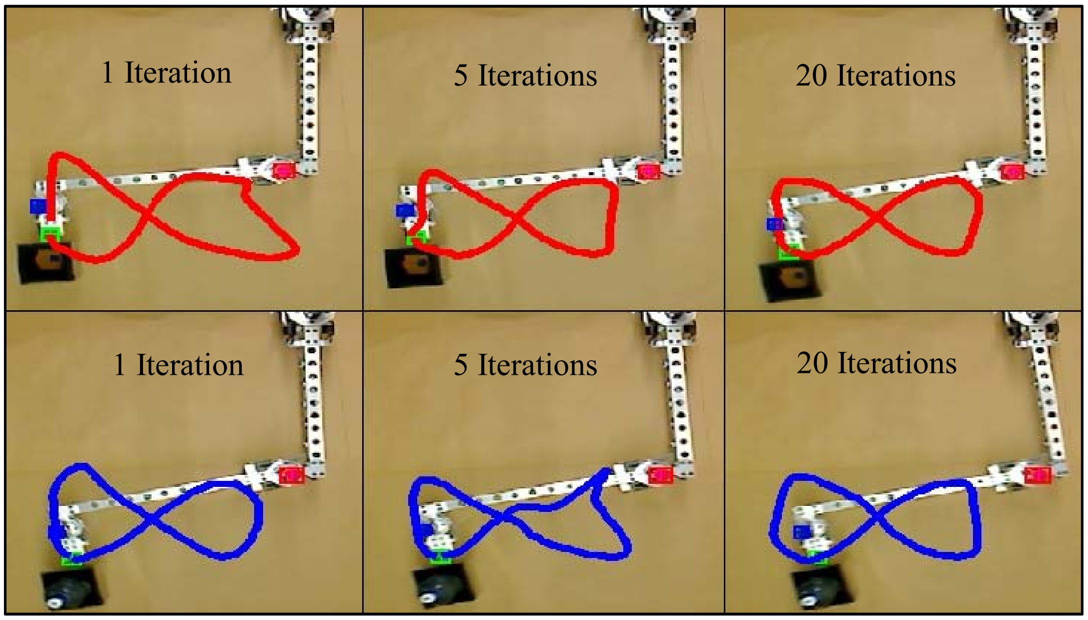


Figure 6. Sequence of images with end effector traces overlayed. The top row with red line corresponds to learning without the object, and the bottom row with blue line corresponds to learning with the object.

the previous system is forgotten. Furthermore, the learning curves show that once a new model is adapted it has the ability to readapt to the original system and produce similar control performance. A video recording of the experiment was also used as an estimate of ground truth to further validate the results. A trace of the end effector from the video frames was generated by post-processing the image and performing colour tracking of coloured dots placed on the robot arm. Fig. 6 shows the resulting tool point figure of eight traces of the experiment, comparing the initial performance up to the last iteration with and without an object.

VII. Discussion and Future Work

A. Discussion

The results show a marked improvement in control performance for the EJR when LWL-MPC is applied compared to the PD controller. The results show that even if the dynamics change a similar performance can be achieved after a brief period of adaptation.

The LWL-MPC system is widely applicable in robotic applications. In this experiment the system rapidly learned the raw motor voltages required to control a compliant multi-degree of freedom robot arm. It achieved this result despite the low cost design of the elastic joint arm, which involves real world problems such as high stiction and deadzones from geared DC motors and non-ideal springs that experience fatigue and nonlinear friction from bearings.

The underlying performance of LWL-MPC was found to depend on the density of models in the RFWR algorithm, and control effort determined by the MPC cost function. The only caveat to LWL-MPC deployment in other robotic applications is the initial selection of hyper parameters in the learning algorithm and MPC parameters, such as cost function weights and horizon window size.

B. Future Work

Improving the control performance will be investigated by analysing parameters such as the weights defined in the cost function and selection of the horizon window to produce better performance. This could involve determining the parameters online using statistical information gathered from the RFWR to modify control costs with respect to uncertainty of the models. Improvements could also be made by recursively updating the MPC control law in (11), instead of directly computing the inverse every control cycle.

Further work is also required to analyse the stability and robustness of the controller, including developing methods to ensure convergence of the learning system while performing stable control. Work has been done in this area and could be used to extend LWL-MPC to ensure stability and robustness [6].

LWL-MPC has the ability to extend its use to more complicated control tasks. One extension that will be investigated is the ability to build local models of a system which incorporate task or context dependent variables where underlying changes in system dynamics occurs. This would improve the system by giving it the ability to store multiple representations of its dynamics, removing the requirement to readapt when context dependent system changes occur.

References

[1] A. De Luca and W. Book, "Robots with flexible elements", Handbook of Robotics, B. Siciliano and O. Khatib, Eds., Springer, 287–319, 2008

[2] C. Lehnert and G. Wyeth, "Adding a Receding Horizon to Locally Weighted Regression for Learning Robot Control", In International Conference on Intelligent Robots and Systems, San Francisco, California, 2011

[3] M. Li, Y. Yang, F. Gao and F. Wang, "Fuzzy multi-model based adaptive predictive control and its application to thermoplastic injection molding", The Canadian Journal of Chemical Engineering, 79(2):263-272, 2001

[4] J. M. Florez, D. Bellot and G. Morel, "LWPR-model based predictive force control for serial comanipulation in beating heart surgery", IEEE/ASME International Conference on Advanced Intelligent Mechatronics, pages 320-326, 2011

[5] G.W. Irwin and S. Townsend, "Predictive control using multiple model networks", pages 5/1-5/7, 1999

[6] A. Aswani, H. Gonzalez, S.S. Sastry and C. Tomlin, "Provably safe and robust learning-based model predictive control", Arxiv preprint arXiv:1107.2487, 2011

[7] A. Aswani, P. Bouffard and C. Tomlin, "Extensions of Learning-Based Model Predictive Control for Real-Time Application to a Quadrotor Helicopter", Proc. American Control Conference (ACC), pages (Montreal, Canada), 2012

[8] W. Li and E. Todorov, "Iterative linearization methods for approximately optimal control and estimation of non-linear stochastic system", International Journal of Control, 80(9):1439-1453, 2007

[9] D. Mitrovic, S. Klanke and S. Vijayakumar, "Learning impedance control of antagonistic systems based on stochastic optimization principles", The International Journal of Robotics Research, 30(5):556-573, 2011

[10] D. Nguyen-Tuong and J. Peters, "Incremental online sparsification for model learning in real-time robot control", Neurocomputing, 2011

[11] D. Nguyen-Tuong and J. Peters, "Model learning for robot control: a survey", Cognitive Processing, 1-22, 2011

[12] Olivier Sigaud, Camille Salaün and Vincent Padois, "On-line regression algorithms for learning mechanical models of robots: A survey", Robotics and Autonomous Systems, 2011

[13] C.G. Atkeson, A.W. Moore and S. Schaal, "Locally weighted learning for control", Artificial intelligence review, 11(1):75-113, 1997

[14] S. Schaal and C.G. Atkeson, "Constructive incremental learning from only local information", Neural Computation, 10(8):2047-2084, 1998

[15] S. Vijayakumar and S. Schaal, "Locally weighted projection regression: An O (n) algorithm for incremental real time learning in high dimensional space", pages 288–293, 2000

[16] L. Ljung and T. Söderström, "Theory and practice of recursive identification", 1983

[17] C. Lehnert and G. Wyeth, "Locally Weighted Learning Model Predictive Control for Elastic Joint Robots", In Australasian Conference on Robotics and Automation, Wellington, New Zealand, 2012

[18] E.F. Camacho and C. Bordons, Model predictive control. Springer Verlag, 2004

[19] G. Wyeth, "Demonstrating the safety and performance of a velocity sourced series elastic actuator", pages 3642-3647, 2008