Adaptive Control of Hydraulic Systems with MML Inferred RBF Networks

Daniel F. Schmidt, Andrew P. Paplinski, Gordon S. Lowe
Department of Computer Science and Software Engineering
Monash University
Melbourne, Australia
Email: {Daniel.Schmidt, Andrew.Paplinski, Gordon.Lowe}@csse.monash.edu.au

***Abstract*— In this paper the problem of adaptively controlling a hydraulic system with uncertainties is considered. An adaptive controller is derived to control actuator force with unknown valve flow coefficients and fluid parameters. This is subsequently cascaded into a position controller which uses RBF networks to compensate for the effects of friction in the system. In contrast to conventional adaptive controllers, the controller is augmented with a further layer that adaptively determines the optimal architecture for the RBF networks using the Minimum Message Length costing criterion. This provides an automated method of determining when it is no longer advantageous to increase the network size. Stability results are presented, and simulation demonstrates the ability of the MML criterion to determine when a suitable fit has been achieved.**

I. Introduction

Fluid power has played an important role in most forms of heavy industry for many years now. Hydraulic actuators are capable of producing much greater forces than electric motors, and are often more compact. Subsequently, they find use in many applications ranging from robotics [10], to machine tools and mining equipment. However, hydraulically powered systems are highly nonlinear in nature, making them difficult to control with linearised models. Additionally, various uncertainties common in the hydraulic systems make the problem more difficult. This paper presents a stable, bounded adaptive control scheme that handles uncertainties in the hydraulic actuator, and uses the nonlinear approximating abilities of RBF networks to compensate for friction present the system. Additionally, and in contrast to most adaptive control schemes that use approximation networks, we do not *a priori* select a network size, but rather augment the system with a further adaptation layer that uses the Minimum Message Length criterion to select an architecture online. As far as the authors are aware, it is the first time the MML criterion has been applied to automatic control.

II. Previous Work

Various approaches to efficiently control hydraulic systems and handle the nonlinearities have been considered over time, including both adaptive and non-adaptive techniques. Variable structure control has been widely used, and provides controllers that are tolerant of inexact system modelling, though this advantage can be offset by the need to carefully tune the deadzones if good performance is to be achieved. Both linearised [1] and nonlinear [11] adaptive controllers have been developed for hydraulic systems, with the work in [11] being applied to systems using three-port valves, and adaptively identifying both fluid parameters and rigid body parameters. A nonlinear controller based around a Integrator Backstepping like approach [4] for a hydraulic cylinder has been proposed in [12] which gives good performance on a small volume, high pressure hydraulic cylinder. In this approach the hydraulic parameters and friction are identified offline. The approach presented in this paper is similar but uses online identification of parameters and RBF networks to compensate friction, with the MML criterion used to determine the required sizes of the networks.

Radial Basis Function networks have previously been applied to nonlinear, adaptive control on many occasions with good results [2], [5], [8], [9], motivated by the fact they are universal approximators over a compact region [6]. Much of the recent work on using neural networks in control has focused on deriving stable online training laws, which generally involves adjusting only the output weights of the networks and assuming the other parameters are fixed in some regular fashion. The problem then becomes one of linear regression, and stable, bounded update laws for the weights can be derived through Lyapunov analysis. Integrator Backstepping [4] can then be used to extend the results to a large class of dynamic systems. Most of these methods, however, ignore the issue of the selection of the size of networks, usually assuming the networks are picked large enough to hopefully approximate all nonlinearities reasonable well. This can lead to oversized networks being used, increasing both the computation time and the dynamic order of the system.

The Minimum Message Length [13], [14], [15] objective costing criterion is a method of evaluating the performance of models, both in terms of the model's fit to the data, and the complexity of the model. In this fashion it regulates the size of the model, yielding a model for which increases in complexity are not expected to produce much increase in performance. This also has the benefit of potentially avoiding the overfitting problem. MML has been applied successfully to various problems including RBF networks with free radius and centre parameters [7].

In this paper we concentrate on identifying hydraulic parameters for a system controlled with a five-port valve, and

compensating for friction with MML inferred RBF networks. The parameters of the body being moved by the hydraulic actuators are assumed to be known, and emphasis is put on the architecture adaptation layer of the control system.

III. Problem Statement: Fluid Power Systems

A physical system driven by a double acting fluid-powered actuator using a five-port valve conventionally connected as in [12] has the following general equations of motion

$$\dot{x}_1 = x_2 \tag{1}$$

$$\dot{x}_2 = \frac{1}{m}(F + \Theta(x_1, x_2) + h(x_2)) \tag{2}$$

$$\dot{F} = A_1\dot{p}_1 - A_2\dot{p}_2 \tag{3}$$

$$\dot{p}_1 = \frac{\beta}{V_1}(-\dot{V}_1 + q_1(u)) \tag{4}$$

$$\dot{p}_2 = \frac{\beta}{V_2}(-\dot{V}_2 + q_2(u)) \tag{5}$$

where x_1 is the position of the system, x_2 the velocity, F the force exerted by the fluid-power actuator, p_1, p_2 are the pressures in chambers 1 and 2 of the actuator, $\Theta(\cdot)$ accounts for the dynamics of the physical system, $h(\cdot)$ are the frictional forces, q_1, q_2 are the flows of fluid into chambers 1 and 2, A_1, A_2 are the areas of contact between the fluid and actuator in chambers 1 and 2, V_1, V_2 are the volumes of fluid in the chambers 1 and 2, m is the attached mass being moved, and u is the control input to the system. Given some reference trajectory, x_{1d}, the general tracking problem is to drive $x_1 \to x_{1d}$ as $t \to \infty$. Assuming the system is affine in the control, u, the flows, q_1, q_2, are given by the servo-valve equations

$$q_1 = \begin{cases} c_a\sqrt{p_s - p_1}u & \text{for } u \geq 0 \\ c_b\sqrt{p_1 - p_r}u & \text{for } u < 0 \end{cases} \tag{6}$$

$$q_2 = \begin{cases} -c_c\sqrt{p_2 - p_r}u & \text{for } u \geq 0 \\ -c_d\sqrt{p_s - p_2}u & \text{for } u < 0 \end{cases} \tag{7}$$

where β is the fluid bulk modulus, p_1, p_2 the pressures in the two bellows, p_s, p_r the supply and reservoir pressures and u is the control input to the servovalve. Using (4), (5) in (3) yields

$$\dot{F} = -\beta x_2\left(\frac{A_2^2}{V_2} + \frac{A_1^2}{V_1}\right) + v(x_1, p_1, p_2)u \tag{8}$$

where

$$v = c_1 v_1(\cdot) + c_2 v_2(\cdot) \tag{9}$$

with

$$v_1 = \begin{cases} \frac{A_1}{V_1}\sqrt{p_s - p_1} & \text{for } u \geq 0 \\ \frac{A_1}{V_1}\sqrt{p_1 - p_r} & \text{for } u < 0 \end{cases} \tag{10}$$

$$v_2 = \begin{cases} \frac{A_2}{V_2}\sqrt{p_2 - p_r} & \text{for } u \geq 0 \\ \frac{A_2}{V_2}\sqrt{p_s - p_2} & \text{for } u < 0 \end{cases} \tag{11}$$

and

$$c_1 = \begin{cases} c_a\beta & \text{for } u \geq 0 \\ c_b\beta & \text{for } u < 0 \end{cases} \tag{12}$$

$$c_2 = \begin{cases} c_c\beta & \text{for } u \geq 0 \\ c_d\beta & \text{for } u < 0 \end{cases} \tag{13}$$

$$\tag{14}$$

The variables β, c_1, c_2, and friction function $h(x_2)$ are all assumed to be *a priori* unknown and possibly time varying. This is reasonable as the bulk modulus is dependant on the quality of the fluid, which may vary over time, the valve flow coefficients depend on the construction of the valve itself, and may vary from valve to valve, and friction depends on the construction of the system. It is possible to additionally treat the other parameters, such as the stroke and volume, as *a priori* unknown but these parameters are easily measured and remain constant, and thus little advantage would be gained in doing so.

IV. Function Approximation with MML Costed RBF Networks

A. Minimum Message Length Inference

The Minimum Message Length costing criterion [13] is used to objectively cost a model, in terms of its fit to the data and its complexity. Using the MML87 approximation [15] of the Minimum Message Length (MML) inference scheme, the 'optimal' parameters, θ^*, of some model for a given set of data $\mathcal{X}$ are

$$\theta^* = \arg\min_{\theta \in D}\{-\log(h(\theta)) + \frac{1}{2}\log(F(\theta)) - \log(f(\mathcal{X}|\theta)) + K\} \tag{15}$$

where $h(\cdot)$ are the priors over the parameters, $F(\cdot)$ is the determinant of the expected Fisher Information Matrix, $f(\cdot)$ is the likelihood function of the data given the parameters, and K is some constant introduced through the approximation.

B. RBF Networks

Consider a network of m radial basis functions, with q inputs, given by

$$y = \sum_{i=1}^{m} w_i \phi(x - c_i, r_i) \tag{16}$$

where $\mathbf{w} \in \Re^m$ are the output weights, $\mathbf{r} \in \Re^{m+}$ are the basis function 'radius' parameters, $\mathbf{c} \in \Re^{m \times q}$ the basis function centres, $x \in \Re^q$ is the data input, and $\phi(\cdot)$ is some nonlinear basis function. As RBF neurons are *local approximators*, providing $\mathbf{c} \in R$, where R is a q dimensional region that encompasses the maximum and minimum values of the input vector x, an arbitrary approximation of any nonlinear function can be achieved as $m \to \infty$, assuming appropriate selection of $\mathbf{c}$ (for example, a uniform grid). Clearly this is unreasonable and we would like to be able to automate the optimal selection of m; this has the dual result of preventing overfitting in the presence of noise, and reducing computation time. We propose to use the MML costing criterion to automatically select an optimal value of m in our adaptive control system.

C. Message Length for Fixed Radius and Centre RBFs

A Message Length formulation for RBF networks with variable **c** and **r** parameters has been described in [7]. This is used as a basis to derive a message length for fixed hidden layer parameter networks, in which **c** and **r** are determined by some spacing specification.

1) Likelihood: Define the outputs of the basis functions as

$$\Phi(x) = [\phi_1(x - c_1, r_1), ..., \phi_m(x - c_m, r_m)] \tag{17}$$

Given some Gaussian noise corrupting the target with a precision of $\tau = \frac{1}{\sigma^2}$ and N input/target exemplar pairs (x_i,y_i), the likelihood function, $f(y|x, \theta)$ is given by

$$f = (2\pi)^{-\frac{N}{2}} \tau^{\frac{N}{2}} \exp\left(-\frac{\tau}{2} \sum_{n=1}^{N} (y_n - \mathbf{w}^T \Phi(x_n))^2\right) \tag{18}$$

and the negative-log likelihood is $L = -\log(f)$.

2) Fisher Information: The Fisher Information is the determinant of the expectation of the second derivatives of the likelihood with respect to the parameters, and gives a measure of the sensitivity of the model on the parameters. Given N pieces of data, the terms in the $m + 1 \times m + 1$ Fisher Matrix for our problem are

$$\left\langle \frac{\partial^2 L}{\partial \tau^2} \right\rangle = \frac{N}{2\tau^2} \tag{19}$$

$$\left\langle \frac{\partial^2 L}{\partial \tau \partial w_i} \right\rangle = \left\langle \frac{\partial^2 L}{\partial w_i \partial \tau} \right\rangle = 0 \tag{20}$$

$$\left\langle \frac{\partial^2 L}{\partial w_i \partial w_j} \right\rangle = \tau \sum_{n=1}^{N} \Phi_i(x_n) \Phi_j(x_n) \tag{21}$$

where $\langle \cdot \rangle$ denotes an expectation.

3) Priors: To find the full message length prior probability distributions must be defined over the model parameters. We use a uniform prior over the number of neurons, with some upper limit M being the reasonable upper limit of neurons that could be considered to be used, a log-uniform prior for τ, and a Gaussian prior for the weights, with fixed accuracy τ_W. Additionally, the range of the input space the network spans, given by $x_{min}, x_{max} \in \Re^q$, must also be stated so that the **c** and **r** parameters may be generated; a uniform prior over some region Q of the input space is used for these parameters. Defining $\theta = [m, \tau, \mathbf{w}, x_{min}, x_{max}]$, we have

$$h(\theta) = h(m)h(\tau)h(\mathbf{w})h(x_{min})h(x_{max}) \tag{22}$$

$$h(m) = \frac{1}{M} \tag{23}$$

$$h(\tau) = \frac{2}{(\tau_U^2 - \tau_L^2)\tau} \tag{24}$$

$$h(\mathbf{w}|\tau_W) = (2\pi)^{-\frac{m}{2}} \tau_W^{\frac{m}{2}} \exp\left(-\frac{\tau_W}{2} \sum_{n=1}^{m} w_n^2\right) \tag{25}$$

$$h(x_{min}) = h(x_{max}) = \frac{1}{Q^q} \tag{26}$$

$$\tag{27}$$

4) Message Length: The complete message length expression is given by

$$\mathcal{M}(y|x, \theta) = -log(h(\theta)) + \frac{1}{2}|F(x, \theta)| - log(f(y|x, \theta) + K \tag{28}$$

The MML point estimate of model parameters $\hat{\theta}_{MML}$ is found by setting $\frac{\partial \mathcal{M}}{\partial \theta_i} = 0$ and solving for θ_i.

5) Parameter Estimation: The estimates for the weights, **w**, are found by the parameter update laws described in the next section. Though this algorithm searches for the weights that yield the minimum mean squared error fit and not the weights that minimise the message length, the differences between the estimates are small, especially if $\frac{1}{\tau_W} \gg \|\mathbf{w}\|_\infty^2$. Given a set of weights, $\hat{\mathbf{w}}$, the MML estimate for τ, $\hat{\tau}_{MML}$ can be found explicitly by solving $\frac{\partial \mathcal{M}}{\partial \tau} = 0$ for τ and is given by

$$\hat{\tau}_{MML} = \frac{N - m}{\sum_{n=1}^{N} (y_n - \hat{\mathbf{w}} \phi(x_n))^2} \tag{29}$$

V. Nonlinear Controller

A. Force Control

The first step is to derive a stabilising controller for the actuator force, with unknown parameters β, c_1 and c_2. Rather than directly tracking the desired force signal, we require our adaptive control to instead track the output of a first-order reference model, given by

$$\dot{F}_R = K_F(F_D - F_R) \tag{30}$$

where F_D is the desired force, F_R the output of our reference model, and K_F determines the dynamic response of our reference model. Defining $z = F - F_R$ as the error between the actual force and the reference model then $\dot{z} = \dot{F} - \dot{F}_R$. We then select the control law u as

$$u = \frac{\hat{\beta} x_2 \left(\frac{A_1^2}{V_1} + \frac{A_2^2}{V_2}\right) - K_F z + \dot{F}_d}{\hat{c}_1 v_1 + \hat{c}_2 v_2} \tag{31}$$

where $\hat{(\cdot)}$ denotes an estimate. Defining $\kappa \equiv u$, the update laws for our paramater estimates are

$$\dot{\hat{\beta}} = \gamma_\beta x_2 \left(\frac{A_1^2}{V_1} + \frac{A_2^2}{V_2}\right) z \tag{32}$$

$$\dot{\hat{c}}_1 = \gamma_1 \kappa v_1 z \tag{33}$$

$$\dot{\hat{c}}_2 = \gamma_2 \kappa v_2 z \tag{34}$$

where γ_β, γ_1, $\gamma_2 > 0$ are adaptation gains.

B. Position Control

We now construct an outer position control loop to ensure that x_1 tracks some twice differentiable signal. Rather than directly tracking the reference trajectory, x_{1d}, we require our controller to track the output of a reference model, x_{1r}, given by

$$\ddot{x}_{1r} = a_1(x_{1d} - x_{1r}) - a_2 \dot{x}_{1r} \tag{35}$$

with $a_1 > 0$, $a_2 > 0$. This has the benefit that the derivatives $\dot{x}_{1r}$ and $\ddot{x}_{1r}$ are directly available for measurement from our reference model. Defining

$$\varepsilon_1 = x_1 - x_{1r} \quad (36)$$

$$\varepsilon_2 = x_2 - \dot{x}_{1r} = x_2 - x_{2r} \quad (37)$$

we propose that the desired force be

$$F_D = m\dot{x}_{2r} - K_P\varepsilon_1 - K_V\varepsilon_2 - \Theta(\cdot) + \mathbf{w}_H^T\Phi_H(\cdot) \quad (38)$$

where $\Phi_H(\cdot)$ is an RBF network used to approximate the friction function $h(\cdot)$. The parameter update law for the weights $\mathbf{w}_H$ is given by

$$\dot{\mathbf{w}}_H = -\Lambda_H\left((\varepsilon_2 + K_1\varepsilon_1)\Phi_H(|x_2|) - \sigma\mathbf{w}_H\right) \quad (39)$$

where $\Lambda_H \in \Re^{m\times m+}$ is a matrix of adaptation gains, K_1 is a constant picked as described in the next section, and $\sigma > 0$ is a small constant used to regulate the weights and prevent drifting and growth (the so called 'sigma-modification' [8]). The weights are given by

$$\mathbf{w}_H = \begin{cases} \mathbf{w}_H^+ & \text{for } x_2 > 0 \text{ or } x_2 = 0, F_{D-H} > 0 \\ \mathbf{w}_H^- & \text{for } x_2 < 0 \text{ or } x_2 = 0, F_{D-H} < 0 \\ 0 & \text{for } x_2 = 0, F_{D-H} = 0 \end{cases} \quad (40)$$

and the network by

$$\mathbf{\Phi}_H = \begin{cases} \Phi_H^+ & \text{for } x_2 > 0 \text{ or } x_2 = 0, F_{D-H} > 0 \\ \Phi_H^- & \text{for } x_2 < 0 \text{ or } x_2 = 0, F_{D-H} < 0 \\ 0 & \text{for } x_2 = 0, F_{D-H} = 0 \end{cases} \quad (41)$$

where F_{D-H} as in (38) without the friction compensation term. We use two networks depending on the sign of x_2 to avoid the inevitable discontinuity present at $x_2 = 0$ due to the nature of the static friction forces, which fixed grid RBF networks cannot easily approximate.

C. Stability

Claim: The system described by (1 - 7), controlled by control laws (31) and (38), with parameter update laws (32), (33), (34) and (39) is stable and bounded.

Proof:

Define $\tilde{(\cdot)} = \hat{(\cdot)} - (\cdot)^*$, where $(\cdot)^*$ represents the optimal value of the estimate, i.e. $\beta^* = \beta$, $c_1^* = c_1$, $c_2^* = c_2$, and

$$\mathbf{w}_H^* = \arg\min_{\mathbf{w}_H\in\Re^m}\left\{\sup_{x\in R}|h(x) - \mathbf{w}_H^T\Phi_H(x)|\right\} \quad (42)$$

where R is the region over which $\Phi_H(\cdot)$ is chosen to approximate $h(\cdot)$.

1) Force Stabilisation: We propose the following Lyapunov function

$$V_F(z,\tilde{\beta},\tilde{c}_1,\tilde{c}_2) = \frac{1}{2}z^2 + \frac{1}{2}\gamma_\beta^{-1}\tilde{\beta}^2 + \frac{1}{2}\gamma_1^{-1}\tilde{c}_1^2 + \frac{1}{2}\gamma_2^{-1}\tilde{c}_2^2 \quad (43)$$

taking the derivative of (43) along the trajectories of (8), and using (31), (32 - 34) yields

$$\dot{V}_F = -K_F z^2 \quad (44)$$

and thus $F \to F_D$ as $t \to \infty$, or $F = F_D + \Delta_F$, where $\Delta_F \to 0$ as $t \to \infty$.

2) Position Control: We treat ε_2 as a 'virtual control', and define $\zeta = \varepsilon_2 + K_1\varepsilon_1$ as the error between the ε_2 and the controller that stabilises ε_1 (i.e. a backstepping formulation [4]). The system in (ε_1, ζ) co-ordinates is then given by

$$\dot{\varepsilon}_1 = z - K_1\varepsilon_1 \quad (45)$$

$$\dot{\zeta} = \dot{\varepsilon}_2 + K_1\dot{\varepsilon}_1 \quad (46)$$

Now consider the Lyapunov function

$$V_P(\varepsilon_1,\zeta) = \frac{1}{2}\varepsilon_1^2 + \frac{1}{2}\zeta^2 + \tilde{\mathbf{w}}_H^T\Lambda_H^{-1}\tilde{\mathbf{w}}_H \quad (47)$$

Taking the derivative along trajectories of (1), (2) yields

$$\dot{V}_P = -K_1\varepsilon_1^2 + \zeta(\frac{1}{m}(F - h(x_2) + \Theta(\cdot)) - \dot{x}_{2r} \quad (48)$$

$$+K_2\dot{\varepsilon}_1 + \varepsilon_1) + (\mathbf{w}_H - \mathbf{w}_H^*)\dot{\mathbf{w}}_H \quad (49)$$

Now, we set

$$F_D = \mathbf{w}_H^T\Phi_H(\cdot) + m\dot{x}_{2r} - mK_1\dot{\varepsilon}_1 - \varepsilon_1 - mK_2\zeta - \Theta(\cdot) \quad (50)$$

From (44) we know that $F \to F_D$ as $t \to \infty$, so setting $F = F_D + \Delta_F$, where Δ_F is some disturbance that decays to 0, and using (39) in (48) we have

$$\dot{V}_P = -K_1\varepsilon_1^2 - K_2\zeta^2 + (\delta_H + \Delta_F)\zeta - \sigma\tilde{\mathbf{w}}_H^T\mathbf{w}_H \quad (51)$$

and by completion of squares

$$\dot{V}_P \le -K_1\varepsilon_1^2 - K_2\zeta^2 + (\delta_H + \Delta_F)\zeta + \frac{\sigma\|\mathbf{w}_H^*\|^2}{2} - \frac{\sigma\|\tilde{\mathbf{w}}_H\|^2}{2} \quad (52)$$

where δ_H is an error term introduced by the approximation of $h(\cdot)$ by $\mathbf{w}_H^T\Phi_H(\cdot)$. Given that Δ_F, δ_H and $\|\mathbf{w}_H^*\|$ are all bounded, we can conclude that ε_1 and ε_2 are bounded. It is important to note that (50) is the same as (38), with

$$K_V = m(K_1 + K_2) \quad (53)$$

$$K_P = m(K_1K_2 + 1) \quad (54)$$

Selecting $K_1 > 0$, $K_2 > 0$ ensures the overall system is stable and bounded.

VI. Architecture Adaptation

A standard adaptive controller consists of control and parameter update laws; the selection of architecture for any function approximation networks is done prior to implementation and is usually selected to be large enough to hopefully deal with the *a priori* unknown nonlinearities the network must approximate. This typically results in much larger networks than are required, which increases the both dynamic order of the system and computation time.

In contrast, we add another adaptation layer to the system, running at a sampling rate generally much lower than the control and parameter update laws, that selects the architecture of our networks. The controllers and parameter estimators drive our errors to some bounds, Δ, defined by the current network architecture. The architecture adaptation layer drives these bounds down to the smallest practical value, that is, the value for which increases in m have little impact on our control performance. Figure 1 shows the block diagram of the complete control system.

A. Algorithm Outline

We use the MML costing criterion to assess the performance of our RBF networks. The basic procedures is as follows:

1) Gather data from measurements
2) Cost the current network with data
3) When training has stabilised:
    a) If the current architecture is cheaper than previous, smaller architecture, increase the size of the network
    b) If current architecture is more expensive than previous, smaller architecture, return to previous size
4) Repeat ad infinitum

This simple algorithm iteratively increases the architecture size until the cost of stating the architecture outweighs the benefits gained from the extra neurons. Additionally, the 'look-ahead' depth can be set to greater than one; that is, have the algorithm check two or more increased networks sizes before making a decision in step (3). The following sections outline the procedure in greater detail.

B. Data Collection

Data is required to cost the networks. This problem, and the problem of forgetting collected data if the system cannot be assumed to be time invariant, has many solutions, each with advantages and drawbacks [3]. We use a method that is simple to implement and allows for data to be forgotten over time. Assuming a maximum of N samples is required, the input space to the networks is split into $m-1$ partitions based on the basis function centres, $\mathbf{c}$, with each partition holding up to $N/(m-1)$ samples. As measurements are read in, they are added into their appropriate partition with a probability of p_{accept}, randomly replacing one of the existing samples stored in the partition if the partition is already full. Clearly, p_{accept} determines the rate of 'forgetfullness' in the data collection. This is not necessarily an ideal parititioning of the data space - obviously it is more desirable to have larger numbers of exemplars in regions that are more nonlinear - but it is at least unbiased and ensures that there is a suitable amount of data available for each parameter we are attempting to fit (i.e. each w_i associated with a c_i). A danger of this scheme is that if system stays in some small region for a large period of time, the partition will fill completely with data samples spanning only this small region. Adding an extra mechanism to the data collection process to detect such situations and decrease p_{accept} accordingly, or increasing the number of partitions can moderate this problem.

C. Costing the Networks

The networks are costed using (28) on the current set of data. As the parameter update laws are always active as long as there is some error, and there will always be some error present due to inaccurate approximations by the networks, there must be some method of determining when training has stabilised to some region of models. To determine this, a simple procedure is used. Costing the networks occurs at some regular interval, T_S, and if the current costed network is the lowest found so far for the current architecture it is saved. If it remains the lowest cost network found over the next H_C costings, where H_C is some positive constant, it is assumed that the training has stabilised. At this point, if the lowest cost network is cheaper than the lowest cost network from the previous architecture, the network size is increased by some number of neurons. If this lowest cost network is more expensive than the cheapest from the previous architecture, the network is resized back to the previous size.

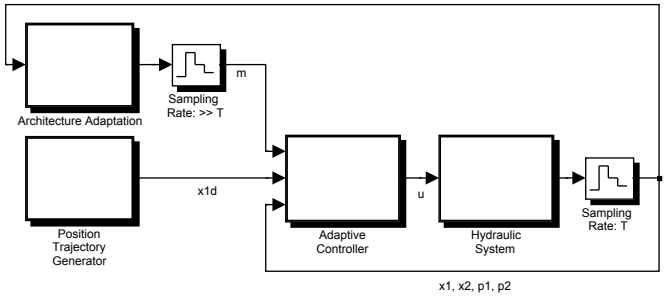

Fig. 1. Block diagram of Adaptive Control System

The MML cost depends on the amount of data used, and unless the entire state space has been traversed reasonably frequently all data collection partitons may not have been completely filled. Therefore, we cost all the networks with the current set of data. Clearly, if we expect poor initial approximations (i.e. the functions to be approximated will be highly nonlinear) it is wise to select H_C as quite large to counter the larger region of possible models the training laws converge to, though naturally this increases the time to it takes for the architecture adaptation.

D. Resizing the networks

To ensure the minimum disturbances are introduced to the controller, and that previously learned information is preserved, the networks are resized by generating the new weights as a fit of the new network to the function learned by the previous one. To this end, P exemplars are generated from each of partitions around the centres, and these are used to fit the new network weights through a linear regression. Additionally, the collected data must be repartitioned based on the centres of our new network.

VII. Simulation Results

To test the adaptive controllers and architecture selection, simulations have been performed in Matlab's™ Simulink™ package. The system to be controlled is a double acting hydraulic cylinder, with a $454kg$ attached mass and pressure supply of 1.379 MPa (200 PSI). A typical hydraulic oil ($\rho = 825kg/m^3$, $\beta = 8.3 \times 10^8$) was used as the fluid, and the valve coefficients were $c_1^+ = 0.43$, $c_1^- = 0.53$, $c_2^+ = 0.83$, $c_2^- = 0.76$. The friction functions have been chosen to be similar to those experimentally found in [12].

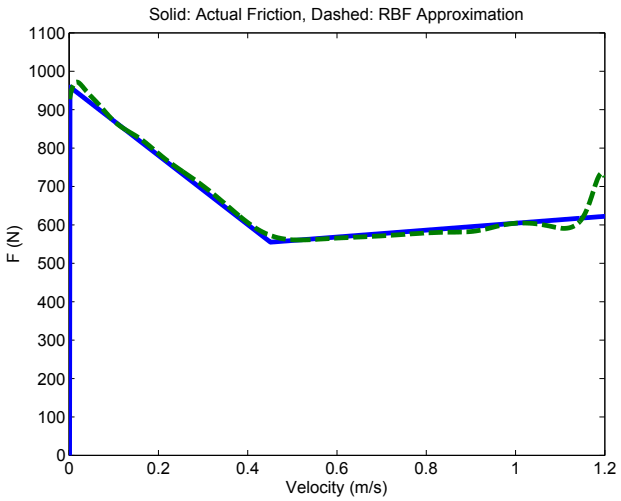

Fig. 2. Approximation of positive friction function

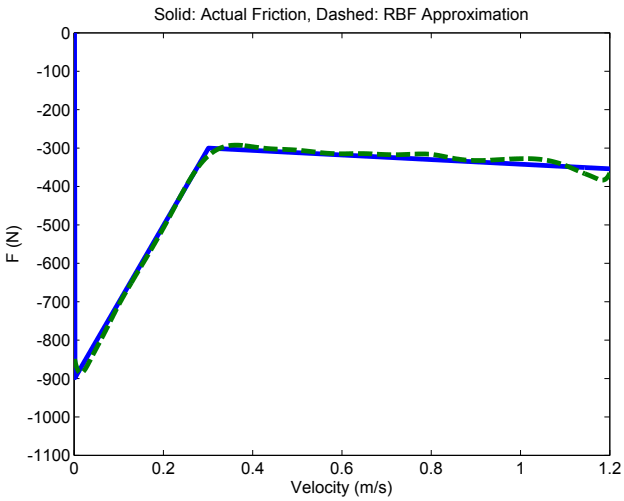

Fig. 3. Approximation of negative friction function

A. Controller parameters

The initial network size for both Φ_H^+ and Φ_H^- was five Gaussian basis functions, arranged in a uniform grid in $[0, 1.2]$ ($1.2ms^{-1}$ being the maximum velocity our system was able to achieve) with an extra basis function one radius to the left and right of 0 and 1.2. For costing the friction networks, assuming a suitable observer, $\hat{\dot{x}}_2$, for $\dot{x}_2$ exists, and measurements for p_1 and p_2 are available, the estimated friction for a some x_2 is given by

$$\hat{h}(x_2) = (p_1 A_1 - p_2 A_2) - m\hat{\dot{x}}_2 + \Delta_U \tag{55}$$

As no real, physical system is completely noise free and perfectly modelled, Δ_U is included to represent disturbances due to unmodelled dynamics, sensor and observer noise. In our simulation, this is modelled by assuming the pressure sensors and the acceleration observer are corrupted by some Gaussian white noise. Equation (55) also assumes that all other forces acting on the system, such as any external forces, are known and accounted for by $\Theta(\cdot)$. For our reference model, parameters $a_1 = 120$, $a_2 = 20$, $K_F = 200$ for our force controller, and $K_1 = 10, K_2 = 100$ for our position control loop. Costing was done with $N = 1000$ data points, and with a horizon $H_C = 5$. The controller sampling rate was $500Hz$, and the architecture adaptation sampling rate was $1Hz$. Additionally, a 'look ahead' depth of two architectures was chosen.

B. Simulation Results

The simulation was run for 60 seconds, with the system being required to track a trajectory of steps and ramps that covered most of the state space. Figures 2 and 3 show the final fits of the two networks to the friction function. The fits are good, with a larger error near $x_2 = 0$ and $x_2 > 1.1$, i.e. very low and high speeds which our trajectories did not often require. The mean squared error between the positive friction function and the fit found by the network is approximately 1.05×10^4, i.e. a standard deviation of approximately $32N$. The mean squared error between the negative friction function and the fit found by the network is approximately 716, i.e. a standard deviation of approximately $26N$. These values can be substituted into δ_H in (52) to find the approximate bounds of stability with the chosen architectures.

In Figure 4 the evolution of message lengths of the two networks over time is shown; at time $40s$ the size of network Φ_H^- is finally selected as 33 basis functions, and it returns to this architecture at $47s$ after trying two larger networks. At $50s$ the size of network Φ_H^+ is selected as 31 basis functions, and it tries two larger architectures before determining they give no decrease in message length and returning to 31 basis, whereupon simulation was halted. Finally, Figure 5 shows the performance of the position controller as it tracks a segment of the reference trajectory. The controller delivers very good performance (standard deviation $< 5.8 \times 10^{-4}m$) with the network sizes that were selected by the MML criterion.

Finally, hydraulic parameter estimates were found to be $\hat{c}_1^+ \approx 0.44$, $\hat{c}_1^- \approx 0.52$, $\hat{c}_2^+ \approx 0.83$, $\hat{c}_2^- \approx 0.79$, and $\hat{\beta} \approx 8.8 \times 10^8$ during the simulation runs. They converged very quickly on these value (within $5s$) and remained in a small region around them for the rest of the simulation run.

C. Remarks

The adaptive controller selected reasonably similar network sizes for both friction networks. Direct comparisons of the message lengths of the two networks are not particularly useful, as both are being costed on different sets of data, and a set that more completely spans the input space will test the networks generalisation capabilities much more heavily. However, the MML criterion has picked reasonable size networks that approximate both functions quite well, and give good tracking results.

One important issue to raise in regards to the adaptive architecture selection scheme is its sensitivity on a selection of a good value of p_{accept} in the collection of data (section VI-B). If p_{accept} is too high, the data used to cost the networks

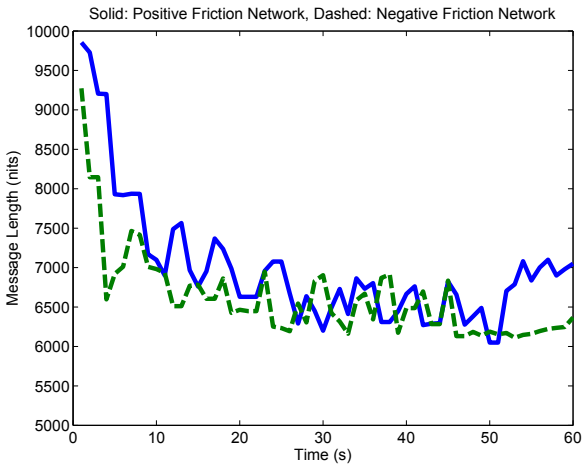

Fig. 4. Network message lengths

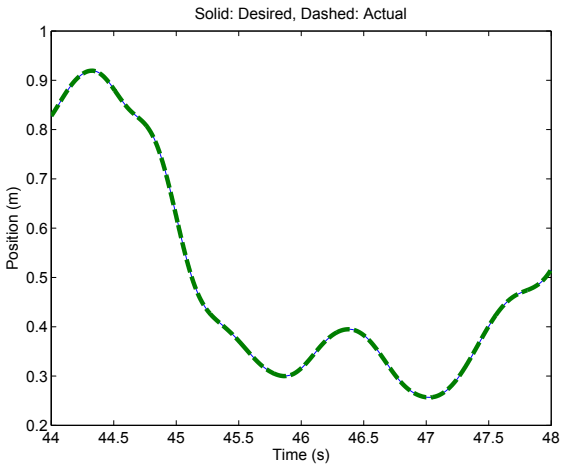

Fig. 5. Position tracking of trajectory

will always contain too much recent data, which the network has just been trained upon. This tends to result in larger architectures costing well because they have 'overfitted' to the most recent data. Selecting a small value of p_{accept} and leaving the system to run for some large length of time before switching on architecture adaptation will remove this problem, but for this to work effectively, the system, if time varying, must change quite slowly over time.

VIII. Limitations and Future Work

Further work will consist primarily of extending the MML costing of the RBF networks in an adaptive control framework; the most obvious progression is to extend the networks to learn the entire physical system's dynamics, $\Theta(\cdot)$, and to approximate the entire valve flow function in (9) with RBF networks to handle extra nonlinearities such as leakage. Further, extending the MML costing to consider systems with process noise is another important direction to pursue, as the advantages of MML model selection are most pronounced in noisy environments. Additional work using MML to find an optimal partitioning of the data space, and improving the architecture adaptation algorithm, in particular determining when training has stabilised, is also considered.

IX. Conclusion

This paper presents a stable, bounded adaptive control system for hydraulic servo systems that adaptively identifies valve flow coefficients and fluid parameters, and uses a pair of RBF networks to compensate for the effects of friction. Additionally, in contrast to conventional adaptive controllers, the system is augmented with a further architecture adaptation layer that automatically selects the size of the networks, using the Minimum Message Length objective costing criterion. Smaller networks have smaller computational requirements and lower dynamic orders, and are thus desirable, and the MML criterion provides a method of determining when the approximation is 'good' enough, and increasing the order will yield small benefits. Additionally, the MML criterion requires no 'tuning' of parameters or constants. Simulation results demonstrate the ability of the architecture adaptation layer to select a reasonable sized network to approximate a typical nonlinear friction function for our nonlinear controller.

References

[1] J. E. Bobrow, K. Lum, "Adaptive, high bandwidth control of a hydraulic actuator", *Proceedings of the 1995 American Control Conference*, pp. 71-75, American Automatic Control Council, June 1995

[2] J. Y. Choi, J. A. Farrell, "Adaptive Observer Backstepping Control Using Neural Networks", *IEEE Transactions on Neural Networks*, Vol. 12, No. 5, pp. 1103-1112, September 2001

[3] T. Hrycej, *Neurocontrol: Towards an Industrial Control Methodology*, Wiley, 1997.

[4] M. Krstic, I. Kanellakopoulos, P. Kokotovic, *Nonlinear and Adaptive Control Design*, John Wiley and Sons, 1995.

[5] Y. Li, S. Qiang, X. Zhuang, O. Kaynak, "Robust and Adaptive Backstepping Contol for Nonlinear Systems Using RBF Neural Networks", *IEEE Transactions on Neural Networks*, Vol. 15, No. 3, pp. 693-701, May 2004

[6] Y. Liao, S. Fang, H. L. W. Nuttle, "Relaxed Conditions for Radial-basis Function Networks to be Universal Approximators", *Neural Networks*, Vol. 16, No. 7, pp. 1019-1028, 2003

[7] E. Makalic, L. Allison, A. Paplinski, "MML Inference of RBF Networks for Regression", *VIII Brazilian Symposium on Neural Networks*, September-October, 2004

[8] H. D. Patino, D. Liu, "Neural Network-Based Model Reference Adaptive Control System", *IEEE Transactions on Systems, Man, and Cybernetics – Part B: Cybernetics*, Vol. 30, No. 1, pp. 198-204, February 2000

[9] R. M. Sanner, J. E. Slotine, "Gaussian Networks for Direct Adaptive Control", *IEEE Transactions on Neural Networks*, Vol. 3, No. 6, pp. 837-863, November 1992

[10] D. F. Schmidt, G. S. Lowe, A. P. Paplinski, "On the Design of a Hydraulically Actuacted Finger for Dextrous Manipulation", *Proceedings of the 2004 IEEE International Conference on Robotics and Automation*, pp. 3001-3006, May 2004

[11] M. R. Sirouspour, S. E. Salcudean, "Nonlinear Control of Hydraulic Robots", *IEEE Transactions on Robotics and Automation*, Vol. 17, No. 2, April 2001

[12] G. A. Sohl and J. E. Bobrow, "Experiments and Simulations on the Nonlinear Control of a Hydraulic Servosystem," *IEEE Transactions on Control Systems Technology*, Vol. 7, pp. 238-247, March 1999

[13] C. S. Wallace, D. M. Boulton, "An Information Measure for Classification", *Computer Journal*, 11(23), pp. 195-209, 1968

[14] C. S. Wallace, D. L. Dowe, "Minimum Message Length and Kolmogorov Complexity", *Computer Journal*, Vol 42., pp. 270-283, 1999

[15] C. S. Wallace, P. R. Freeman, "Estimation and Inference by Compact Encoding (with discussion)" *Journal of the Royal Statistics Society*, Series B, 49, pp. 240-265, 1987