RBF Network Pruning Techniques for Adaptive Learning Controllers

Serge Gale*, Siri Vestheim*, Jan Tommy Gravdahl*, Sigurd Fjerdingen** and Ingrid Schjølberg**

Abstract— **This paper presents two simple and efficient methods for pruning a Radial Basis Network (RBF) used in an adaptive controller architecture for a robotic manipulator. The methods presented in this paper are *Weight Magnitude Pruning* (WMP) and *Node Output Pruning* (NOP). The above pruning methods are simulated on a trajectory tracking task of a three degree of freedom robotic manipulator arm. The RBF based inverse dynamics controller is presented with a task of learning the inverse dynamics of the plant in a closed loop control. Simulation study shows that implementation of an inverse dynamics control law in such manner makes the controller more robust towards uncertainties and disturbances. Pruning RBF network improves controller performance in the case of modelling errors and reduces computational costs, thus making such controller more suitable for implementation.**

I. INTRODUCTION

Over the last decade robotic manipulators have become an integral part of any industry where safety, quality, precision and efficiency are key factors. As manipulators became more common the subject of control of robotic systems became an active research field. Previous research in control highlighted the importance of model based controllers [1]. Performance of such regulators has been found to be superior to non-model based controllers [5]; however the precision of model based controllers highly dependent on the accuracy of the model at hand. It is not always possible however to accurately model a dynamic system and to determine values of all of the parameters involved due to the mechanics of a robotic manipulator itself and surrounding environment containing a great deal of uncertainties and disturbances. In order to address these issues it is necessary to look further than standard controllers. This paper presents adaptive neural controller as a solution to the problem above.

Adaptive neural controllers are able to cope well with uncertainties and modelling errors [16], [8], [5]. Out of various computational intelligence tools currently available Artificial Neural Networks (ANN) have proven to be well suited for adaptive control of mathematically ill-defined systems exposed to structured and unstructured uncertainties [10]. There are many different types of ANNs available; however the focus of this paper is on a feed forward type of network; Radial Basis Function (RFB) networks. Specifically RBF type network is linear in the parameters and is therefore relatively easy to analyse using available stability analysis tools. Secondly, an interesting and useful feature for control engineering application of RBF networks is stated in universal approximation theorem, which states that given sufficient number of processing nodes such a network is able to approximate any continuous real function to any degree of accuracy [8]. It has also been reported that RBF based networks are more computationally efficient compared to multilayer perceptron (MLP) networks in a number of control applications [13]. However, the chase for greater accuracy leads to one of the major problems with all ANN, specifically known as curse of dimensionality [3]. Thus for a given problem most ANNs are redundant i.e. contain more computational nodes than necessary for reconstruction of a final function, however redundancy nodes are helpful during training. Therefore the dilemma is clear, having too larger network will lead to a greater accuracy but poor overall performance due to computational costs and over-fitting issues. When used in control application of robotic manipulators it is vital to have minimal size network for a given problem. However, solution to the described problem presents itself in the form of dynamically growing and shrinking network structures. This paper focuses on pruning method for a given artificial neural network controller.

There are in general two ways to create a network of appropriate size, 1) growing and 2) pruning. Growing starts with a small or empty neural network and adds neurons until some threshold for the approximation error is met. The second method starts with an oversized network and removes the weights and/or neurones that not are necessary for the network. Concept of pruning for artificial neural networks is based on what naturally happens in a biological brain when connections between neurones are destroyed. Specifically for human beings some synapses are naturally removed to increase brains ability to generalise around the age of 20. The remaining connections in the brain have their strength increased [7].

The focus of this paper is to investigate and develop a simple and efficient pruning methods for an RBF type network. The application case study is a ABB IRB140 manipulator arm with a task of trajectory tracking control.

Most of the developed pruning algorithms have been concerned with finding optimal network size with best possible approximation and generalisation criterion. However, many pruning schemes are very time consuming as every attempt is made to avoid removing most contributing nodes. An example would be a pruning method proposed in [18] which is based on complexity.

Some pruning schemes utilise two different networks in a performance check called Cross Validation in order to identify each nodes contribution [9]. Both sets are pruned and the performance is analysed for each set before and after pruning. In [14] this performance check is used for a pruning scheme based on local sensitivity for the parameters in the

*Department of Engineering Cybernetics, NTNU, NORWAY
**SINTEF ICT Applied Cybernetics, NORWAY

network.

Sensitivity or saliency based pruning schemes are perhaps the most common group of pruning schemes. In this group a well known method of *Optimal Brain Surgeon* (OBS) [7] can be found. The *Optimal Brain Damage* (OBD) proposed in [4] states that the weight with the smallest saliency will generate the smallest error variation when removed. Both OBD and OBS start with a network that has already been trained to converge to a local minimum and then try to minimise a cost function by setting one of the weights to zero. OBD makes an assumption that the associated Hessian is diagonally dominant. However in [7] it was found that Hessian in fact was far from diagonally dominant which resulted in OBD for some cases removing the wrong weights. Hence this assumption is not made in OBS and the saliency is found by solving the minimisation problem with a Lagrangian multiplier. The weight with the lowest saliency then is removed. From the same minimisation problem the optimal weight change is also found and this is used to update all the remaining weights. However, the challenge is that it is necessary to calculate the inverse Hessian for each weight that is pruned adding extra computational stress. Pruning continues until there are no more weights that can be removed without causing a large increase in the cost function.

None of the described pruning schemes, which are concerned with finding the optimal network are concerned with computational cost of the pruning algorithm or how computationally demanding the final network will be. In the case of real time control of a robotic manipulator it is crucial that pruning happens fast and with minimal computational cost since this will be done online while controlling a manipulator. In this case it is probably better to start with a rather small network that perhaps will not give the best possible estimations but can give a satisfying approximation fast enough to be used in the controller.

This paper presents two simple pruning methods that are aimed at fast removal of multiple units and offers a network with satisfying approximation and generalisation performance. The first pruning method is based on *Weight Powers Method* [6] but greatly enhances it into a WMP scheme. The second proposed pruning technique is based on a definition from [2]. This is now put into a new setting in a NOP scheme.

In the next section the two proposed pruning methods WMP and NOP are described in detail. Section 3 focuses on the design of a controller for ABB IRB140 manipulator. The proposed regulator is a hybrid combination of an adaptive learning technique with inverse dynamics controller similar to [5] and [12]. The learning part of the controller is achieved through utilisation of an RBF network that is assigned the task of learning system dynamics.

The designed learning controller with the proposed pruning methods is simulated on trajectory tracking control task for the first three joints of the manipulator in Section 4.

II. PRUNING METHODS

Both of the pruning methods presented in this paper are developed for RBF type networks. The unique features of such networks are unity input weights and a single hidden layer, where adjustable weights are associated with the connection between hidden and output neurons. An interesting feature of an RBF type network presented in light of this paper is that due to its unique structure and features, removing a node is equivalent to removing a weight between hidden and output layer neuron, however this is not the case for a RBF network with multiple outputs. A single output network can be presented in a more familiar way as

$$f_{nn}(x) = \sum_{i=1}^{N} w_i a_i(\|x-\mu_i\|), \tag{1}$$

where w_i is the weight associated with the ith connection between the nodes in hidden and output layers, $a_i(\|x-\mu_i\|)$ is activation function (2) output of ith node in hidden layer in response to an excitation provided by input x and $||\cdot||$ is the L_2 norm.

The activation function is the *Gaussian* function given for node i as

$$a_i(||x-\mu_i||) = k_g e^{\frac{||x-\mu_i||}{\sigma^2}}, \tag{2}$$

where k_g is a positive constant, x is the input to the network, σ is the width of the function and μ_i is the centre of the function. The width is fixed the whole time and set to be the distance to the next unit.

It is possible to see from (1) that output of the network is a linear combination of all of the nodes output in the hidden layer. The output depends on two factors 1) weight magnitude $|w|$ and 2) activation function output magnitude $|a(x)|$. It is therefore clear to see that if either of the above factors are $\{(|w|,|a(x)|) \in \mathbb{R}|\ 0 < (|w|,|a(x)|) << 1\}$ then the contribution of the node to the overall output of the network is rather small and can thus be removed.

A. Weight Magnitude Pruning (WMP)

This pruning method is based on a scheme presented in [6], which relies on the fact that small magnitude weights can be successfully removed from the network without negatively affecting overall performance. The novelty of the presented method is in threshold selection mechanism for network optimisation and based on selecting a certain percentage of $||w||$ vector norm. The norm of the weights vector is a measurement of the total strength of the networks weights. As the weights of the network converge to their close to optimal values during training the value of the 2-norm converges to its steady state value $\bar{w}$. Pruning some of the nodes results in remaining ones having their weights increased in magnitude as to match the weight that were deleted.

$$P_{w,th} = \frac{||w||}{100} P_{w,\%}, \tag{3}$$

where $||w||$ is the norm of the weights vector and $P_{w,\%}$ is the pruning threshold. The pruning threshold is the only value that has to be chosen and is given as a percentage

of the weights norm. Weights with magnitude lower than set threshold are then removed. The individual value of weights change mostly at the start of the training and slows down gradually toward the end. Hence the norm of the weights vector grows fast at the start of the training and slowly converges to $\bar{w}$. The weights do not have to converge to their final values before the pruning can begin. While some weights are being removed in certain neighbourhoods it is possible that some key neuron weights become more and more important to the output of the network while previously their presence did not constitute much due to surrounding weights being removed. Since the remaining weights have their magnitude strengthened after pruning a growing threshold has been introduced to overcome such issue. Therefore modifying equation (3) to the following:

$$P_{w,th} = \frac{||w||}{100}(NP_{w,\%}), \tag{4}$$

where N is incremented every iteration. Due to localisation principle [8] weights belonging to neurons in the area far from the input space will have little change in magnitude. Hence such weights are most suitable for WMP method. In addition if the network has densely populated input space with nodes such method will assist in improving network generalisation performance.

B. Node Output Pruning (NOP)

The second contributing factor which affects final network output is the magnitude of the response of the hidden layer nodes activation function to a given input pattern. Since the choice of the activation function is Gaussian, the magnitude of the response depends on the distance from the centre of the function (2). Thus input coinciding with the centres of the activation function will give the strongest response, while nodes in the remote areas will have little or no effect on the network output. Therefore remote nodes can be pruned. This paper proposes a pruning method based on the activation function output ratio [2] given as:

$$o_i^j = |\frac{a_i(||j-\mu_i||)}{a_{max}}|, \tag{5}$$

where j is the input pattern, $a_i(||j-\mu_i||)$ is the activation function for the i_{th} neuron, μ_i is the centre of the activation function of the i_{th} node and a_{max} is the maximum activation function output for a given input. Activation function output ratio is here referred to as node/neuron output to shorten the writing. The novelty of this method is to sum up all the node output ratios for each neuron and remove those with the smallest value. The activation function output ratio sum is defined as:

$$os_i = \sum_{j=1}^{T} o_i^j, \tag{6}$$

where T is the number of previous inputs and o_i^j is given in (5). By specifying a threshold as a percentage, the actual threshold value $P_{o,th}$ is given as:

$$P_{o,th} = \frac{os_{max}}{100}P_{o,\%}, \tag{7}$$

where os_{max} is the highest output ratio sum and $P_{o,th}$ is the pruning threshold to be specified. Under this method, nodes in remote areas from the input are removed, giving a smaller network.

C. Mixed Method Pruning (MMP)

The implementation of a learning adaptive controller in this paper is constructed in such way that each element of inertia, centrifugal, gravity and friction forces matrices are presented with a standalone RBF network, similar to [5]. This results in a number of smaller networks with lower input dimensionality, which makes it more suitable for real time implementation on actual systems. The MMP consists of combination of WMP and NOP as well as application of different threshold on the same network.

III. CONTROL DESIGN

The controller is designed for ABB IRB140 manipulator, model of which will later be used in simulation. However only the first three joints are considered for trajectory tracking purpose. Tracking error is found by using the *Sum of Squared Error* (SSE) function for all three joints while the network approximation error is found by the *Root Mean Square Error* (RMSE) function.

A. Manipulator Model

Consider a well knows dynamic equation of a robotic manipulator with friction and disturbance terms included:

$$M(q)\ddot{q} + C(q,\dot{q})\dot{q} + G(q) + F(\dot{q}) + d = \tau, \tag{8}$$

where $M(q)$ is inertia matrix, $C(q,\dot{q})$ is centrifugal matrix, $G(q)$ is gravitation matrix, $F(\dot{q})$ is friction matrix and d is a constant disturbance term. $q,\dot{q},\ddot{q}$ are joint position, velocity and acceleration respectively.

B. Controller and Desired Trajectory

An inverse dynamics controller is used as in [12]. Representing each matrix component via RBFNN the above equation can be rewritten as:

$$\tau = \hat{M}(q)(\ddot{q}_d + K_d(\dot{q}_d - \dot{q}) + K_p(q_d - q)) + \hat{C}(q,\dot{q})\dot{q} + \hat{G}(q). \tag{9}$$

Stability proof of such regulator based on multiple RBF is given in [15].

C. RBF Model of a Manipulator

Mathematical representation of an RBFNN is given in (1). Using vector notation $\mathbf{w}$ is defined as:

$$\mathbf{w}^T = [\ w_1 \quad \dots \quad w_N\ ]^T, \tag{10}$$

where w_i is a weight connecting a hidden node i to the output neuron. Initially all of the coefficients are set to zero. In the same way it is possible to represent hidden layer activation functions in a vector form, with each element defined in (1).

Similar to [5] each of the elements in dynamic matrices are represented by a separate RBFNN. This will result in input dimension reduction, which is essential in order to prevent curse of dimensionality.

The estimated inertia matrix $\hat{M}$ is defined as:

$$\hat{M} = W_M^T \bullet A_M(x), \tag{11}$$

where W_M matrix defined as:

$$W_M^T \triangleq \begin{bmatrix} \mathbf{w}_{m11}^T & \cdots & \mathbf{w}_{m1N}^T \\ \vdots & \ddots & \vdots \\ \mathbf{w}_{mN1}^T & \cdots & \mathbf{w}_{mNN}^T \end{bmatrix} \tag{12}$$

each element of the matrix in (12) is a column vector defined in (10). Similarly $A_M(x)$ matrix is a matrix of activation functions of constructed RBFNN for an input pattern x and is defined similar to (12). Correspondingly matrices C and G can be represented using matrices constructed of elements based on RBFNN.

Elements of the dynamic equation of a manipulator have some unique properties which can be utilised in order to construct optimal neural network based controller structure. Manipulator inertia matrix ($M(q)$) is three by three elements and is symmetric (i.e. $m_{12} = m_{21}$, $m_{13} = m_{31}$, $m_{23} = m_{32}$), thus only six elements out of nine for a three degree of freedom manipulator are necessary to be approximated, which dramatically reduces computational load on the system. Similarly elements c_{33} of the $C(q,\dot{q})$ matrix and element g_1 of the $G(q)$ matrix are zero and do not need to be estimated using an RBFNN.

IV. SIMULATION RESULTS AND DISCUSSION

Three simulations with inverse dynamics controller are carried out. First simulation assumes frictionless and disturbance free manipulator. The second simulation has friction and disturbance terms included in the manipulator dynamic equation. Inverse dynamics controller requires an accurate model of a plant under control even small inaccuracies in model can lead to instability [17]. Table I shows initial networks configuration and size. Further, *Resource Allocation Networks Extended Kalman Filter (RANEKF)* [11] simulation with growing RBFNN carried out in order to compare the performance from growing networks to pruned networks.

Initial RBF Networks		
Network	*Network Size*	*Input Config*
$\hat{m}_{11}$	81	q_2, q_3
$\hat{m}_{12}$	81	q_2, q_3
$\hat{m}_{13}$	81	q_2, q_3
$\hat{m}_{22}$	9	q_3
$\hat{m}_{23}$	9	q_3
$\hat{c}_{11}$	729	$q_2, q_3, (\dot{q}_2 + \dot{q}_3)$
$\hat{c}_{12}$	729	$q_2, q_3, (\dot{q}_2 + \dot{q}_2 + \dot{q}_3)$
$\hat{c}_{13}$	729	$q_2, q_3, (\dot{q}_2 + \dot{q}_2 + \dot{q}_3)$
$\hat{c}_{21}$	729	$\dot{q}_1, q_2, q_3$
$\hat{c}_{22}$	81	$q_3, \dot{q}_3$
$\hat{c}_{23}$	81	$q_3, (\dot{q}_2 + \dot{q}_3)$
$\hat{c}_{31}$	81	$(\dot{q}_1 + \lambda q_2), q_3$
$\hat{c}_{32}$	81	$\dot{q}_2, q_3$
$\hat{g}_1$	81	q_2, q_3
$\hat{g}_2$	81	q_2, q_3
Sum	**3663**	-

Friction and disturbance terms are implemented as $F = [1.2\dot{q}_1 \ 1.4\dot{q}_2 \ 0.8\dot{q}_3]^T$ and $d = [3 \ 5 \ 4]^T$.

Derivative and Proportional gain values are set to: $K_p = diag(350, 450, 450)$ and $K_d = diag(2.5, 4, 2.5)$

The desired trajectory is specified for each joint as:

$$q_{1d}(t) = 0.5sin(t) \quad q_{2d}(t) = 0.7sin(t) \quad q_{3d}(t) = 0.3sin(t). \tag{13}$$

A. Pruning of Networks

Before the networks may be pruned it is crucial that they have experienced enough training. With WMP all of the weights have to be updated to avoid removing wrong ones. The manipulator must have completed one half period of the first sine wave before the networks can be pruned. For NOP it is necessary that the different areas of the input space all are visited since the pruning is done based on the distance from the units to the inputs. Therefore it is necessary for the manipulator to complete one whole period of a sine wave, that is 2π simulation seconds and approximately 4400 iterations in Simulink. In addition to the simulation time criteria the RMS estimation error over a moving window of 400 Simulink iterations is used. The requirement is that it has to be smaller than 0.01 before a network may be pruned. The exception to this is the network $g2$ which can be pruned when the approximation error is less than 0.05 over the last 400 iterations.

All the networks are initially created as shown in Table I and then pruned with different methods. For WMP method a starting threshold of 5% has been used which can grow up to 5 times. Threshold for NOP has been taken as 30%. Due to the threshold being specified as a percentage the same threshold is used for all of the networks even if they are quite different. Looking at the networks more separately however will make it possible to obtain better result. Thus in MMP different pruning techniques are used to estimate dynamic matrices of the manipulator under study. Networks belonging to $\hat{M}$ are pruned with 7% weight based pruning threshold, while a 30% threshold for NOP method for the $\hat{C}$ is used and 5% for the WMP for the two networks belonging to $\hat{G}$.

B. Obtained Networks After Pruning

Table II presents data on pruned networks using methods presented earlier. The approximation error shown in the table is given for the last 300 iterations of the simulation for each of the network, thus indicating performance of the networks after pruning. From the data it is clear that presented pruning methods significantly reduce the number of hidden neuron in each of the networks. WMP method produced final network with least amount of neurons in the hidden layer, while MMP and NOP produce almost the same result.

C. Results for the Case Without Friction and Disturbance

Table III presents implementation results of the inverse dynamics controller based on suggested methodologies. Here method, total number of hidden units for all the networks, the tracking error for the whole simulation and the approximation error for all of the networks over last 300 iterations are presented. The latter is shown for both simulation times of 6π and 20π seconds.

Resulting Networks After Pruning						
	Weight Magn.		Node Output		Mixed Method	
NN	*NN Size*	*Approx. Error [RMSE]*	*NN Size*	*Approx. Error [RMSE]*	*NN Size*	*Approx. Error [RMSE]*
$\hat{m}_{11}$	6	0.0067127	5	0.0061364	4	0.0054748
$\hat{m}_{12}$	6	0.0005721	5	0.00050466	4	0.00034923
$\hat{m}_{13}$	2	0.001048	5	0.00075247	2	0.0010479
$\hat{m}_{22}$	3	0.004117	2	0.0065574	3	0.0041168
$\hat{m}_{23}$	3	0.001939	2	0.0028382	3	0.0019389
$\hat{c}_{11}$	3	0.0083142	17	0.0016613	17	0.0016601
$\hat{c}_{12}$	13	0.0023642	22	0.0037684	22	0.0037649
$\hat{c}_{13}$	8	0.0072465	22	0.00073191	22	0.00073047
$\hat{c}_{21}$	3	0.0072956	11	0.00084954	11	0.00084958
$\hat{c}_{22}$	2	0.0061949	4	0.0012086	4	0.0012063
$\hat{c}_{23}$	3	0.0006785	7	0.0010091	7	0.0010054
$\hat{c}_{31}$	2	0.0069534	6	0.0043594	6	0.0043596
$\hat{c}_{32}$	2	0.0014019	6	0.0011061	6	0.0011059
$\hat{g}_1$	10	0.23334	5	0.23923	10	0.23334
$\hat{g}_2$	6	0.0021934	5	0.0095059	6	0.002195
Sum	**74**	**0.29037**	**126**	**0.28021**	**129**	**0.26314**

As can be seen from Table III implementing the ideal model of the manipulator in the inverse dynamics controller gives the best tracking result as expected. It can also be seen that an unpruned network gives better tracking result compare to pruned one. Out of the proposed pruning techniques mixed pruning method gives the smallest tracking error. RANEKFs have the largest tracking error which probably is due to the poorer approximation ability at the beginning of the simulation when not enough neurons has been added.

The unpruned network also gives smaller approximation error in total. Again the MMP gives the best result in comparison to other pruning techniques. Here RANEKFs are far superior compared to the other RBF networks with a much smaller estimation error. However the approximation error increases when the simulation length is increased for the RANEKFs. Also the networks did not stop to add neurons and after 20π simulation time at which point the total number of hidden neurons in the RANEKFs was 147. Thus it seems like the RANEKFs keeps growing and starts to over fit the simulation data as the time goes. All of the other networks had the same sizes after the longer simulation and their estimations all improved.

The actual time it took to complete the simulation was much longer when RANEKFs were implemented. It then took 7.20 minutes while the simulation with weight based pruning took 4.10 minutes to complete. The simulation hardware is Intel Core 2 Duo CPU with 2GB RAM. Both NOP and MMP used 4.20 minutes for the whole simulation. Not pruning the networks gave an actual time of 4.50 minutes.

Trajectory Tracking and Network Approximation Errors		
Method	*Sum of Hidden Units*	*Tracking Error [SSE]*
Correct Model	-	0.031897
Unpruned	3672	0.032
Weight Magnitude	74	0.03201
Neuron Output	126	0.032002
Mixed Methods	129	0.032001
RANEKF	96	0.032094

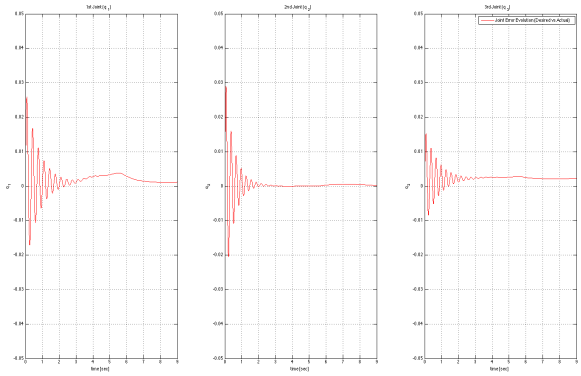

Fig. 1. Joints Error Evolution

Figure 1 shows desired and actual trajectories for each of the joints under inverse dynamics controller with 5% threshold with WMP. It can be seen from the figure that actual trajectory is closely following the desired one.

By looking at the results presented in Table III it is possible to conclude that using RMSE as a measure of performance for such a system is not suitable. While RANKEF network shows best RMSE results it fails to perform as well in actual tracking control task.

D. Results for the Case With Friction and Disturbance

For more realistic simulation it is necessary to include friction and disturbance terms in to the manipulator dynamic equation. These terms are not added to the model of the manipulator used for training of RBFNN, this is aimed at testing networks ability to generalise.

Results from simulations that lasted 6π seconds and includes friction and disturbance are shown in Table IV. This table shows the method that is used to implement the system dynamics in the controller, the total number of hidden units in the final networks, the tracking error for all three joints for the whole simulation and also the tracking error for only the last 2π simulation time. During the whole simulation the manipulator completes 3 full sine wave periods, the tracking error of only the last period is analysed to see how the networks performs in tracking control after being pruned.

The final networks obtained from the NOP and MMP techniques produces networks with different sizes compared to the frictionless and disturbance free model. For example the two networks $\hat{c}_{12}$ and $\hat{c}_{13}$ has one less neuron in the hidden layer. This is due to the change of inputs and thus the activation function ratios are no longer the same as in the previous situation.

Best tracking performance for both the whole simulation and the last part only is obtained using adaptive neural controller with MMP pruned RBF networks. All of the pruned networks give more accurate tracking than the other methods. Worst tracking for the whole simulation come from RANKEFs implemented network. In the final period of the trajectory they do however obtain better tracking then the now incorrect model. RANEKFs are much smaller than

for the case without friction and disturbance. A plot of the desired joint trajectories together with the actual joint trajectories for the case with friction and disturbance can be seen in Figure 2. Here mixed pruning method of the RBF networks in the learning inverse dynamics controller has been used, in this case the tracking result is very good.

Disturbance and Friction Trajectory Tracking and Network Approximation Errors			
Method	*Sum of Hidden Units*	*Tracking Error Whole Sim. [SSE]*	*Track. Error Last 2π Time [SSE]*
Incorrect Model	-	0.052887	0.0072445
Unpruned Networks	3672	0.052977	0.0072385
Weight Magnitude	74	0.052898	0.0072129
Neuron Output	124	0.052975	0.0072379
Mixed Methods	127	0.052563	0.0071018
RANEKF	78	0.053116	0.007243

V. CONCLUDING REMARKS

A. Conclusion

Application of suggested novel pruning methods according to the results have dramatically reduced the size of each network while maintaining good level of controller performance. The reduction in network size results in reduced computational costs of the controller which makes them attractive for a real time controller implementation. The tracking performance may not be ideal case however the results presented in this paper indicate success of the suggested pruning methods, which is the main focus of this paper.

B. Future Work

Future work includes the study of the effects of input signal combination on network performance and complexity. Further work will be focused on study of regularisation of pruned RBF networks to treat ill posed mathematical system, as it was highlighted earlier in this paper. To further extend the case study the suggested control technique will be tested on tracking control of a Universal Robot with 6DOF.

ACKNOWLEDGMENT

This work is performed in Next Generation Robotics for Norwegian Industry. The project is funded by the Research Council of Norway, Statoil AS, Hydro AS, Tronrud Engineering AS, SbSeating AS, Glen Dimplex Nordic AS and RobotNorge AS. SINTEF and the Norwegian University of Science and Technology are research partners.

REFERENCES

[1] C. H. An, C. G. Atkeson, and J. M. Hollerbach. Model-based control of a direct drive arm, part 2: Control. *IEEE Int. Conf. on Robotics and Automation*, 3:1386–1391, April 1988.

[2] S. Arisariyawong and S. Charoenseang. Dynamic self-organized learning for optimizing the complexity growth of radial basis function neural networks. *IEEE Int. Conf. on Industrial Technology*, 1:655–660, 2002.

[3] V.S. Cherkassky, J.H. Friedman, H. Wechsler, and North Atlantic Treaty Organization. Scientific Affairs Division. *From statistics to neural networks: theory and pattern recognition applications*, chapter An Overview of Computational Learning and Function Approximation. NATO ASI series: Computer and systems sciences. Springer-Verlag, 1994.

[4] Y. L. Cun, J. S. Denker, and S. A. Solla. Optimal brain damage. *Proc. of the Neural Information Processing Systems*, pages 598–605, 1990.

[5] S.S. Ge, T.H. Lee, and C.J. Harris. *Adaptive Neural Network Control of Robotic Manipulators*. World Scientific Publishing Co, Singapore, 1998.

[6] M. Hagiwara. Removal of hidden units and weights for back propagation network. *Proc. of Int. Joint Conf. on Neural Networks*, 1:351–354, October 1993.

[7] B. Hassibi, D. G. Stork, and G. J. Wolff. Optimal brain surgeon and general network pruning. *IEEE Int. Conf. on Neural Networks*, pages 293–299, April 1993.

[8] S. Haykin. *Neural Networks – A Comprehensive Foundation*. Pearson - Prentice Hall, ninth reprint 2005 edition, 1999.

[9] T. Q. Huynh and R. Setiono. Effective nn pruning using cross-validation. *Proc. IEEE Int. Joint Conf. on Neural Networks*, 2:972–977, August 2005.

[10] L. C. Jain and Clarence W. De Silva. *Intelligent adaptive control : industrial applications / edited by Lakhmi C. Jain, Clarence W. de Silva*. CRC Press, Boca Raton, Fla. :, 1999.

[11] V. Kadirkamanathan and M. Niranjan. A function estimation approach to sequential learning with neural networks. *Neural Computation*, 5:954–975, 1993.

[12] M. Vidyasagar M. W. Spong, S. Hutchinson. *Robot Modeling and Control*. John Wiley and Sons, INC., first edition.

[13] J. Park, R.G. Harley, and G.K. Venayagamoorthy. Comparison of mlp and rbf neural networks using deviation signals for on-line identification of a synchronous generator. *IEEE Power Engineering Society Winter Meeting*, 1:274–279, 2002.

[14] D. Sabo and X. Yu. A new pruning algorithm for neural network dimension analysis. *IEEE In. Joint Conf. on Neural Networks*, pages 3313–3318, June 2008.

[15] S. Vestheim. Pruning of rbf networks in robot manipulator learning control. Master's thesis, Department of Technical Cybernetics, Norwegian University of Science and Technology, 2012.

[16] C. Wang and D. J. Hill. Learning from neural control. *IEEE Transactions on Neural Networks*, 17(1):130–146, January 2006.

[17] R. Ortega & T. Yu. Theoretical results on robustness of direct adaptive controllers: a survey. In *10th IFAC World Congress*, pages 26–31, Munich, 1987.

[18] Z. Zhang and J. Qiao. A novel pruning algorithm for feedforward neural network based on neural complexity. *Int. Conf. on Intelligent Control and Information Processing*, pages 406–410, August 2010.