CMAC AND HCMAC NEURAL NETWORK CONTROL OF 7-DOF REDUNTANT ROBOT KINEMATICS REDUNDANCY

Vladislav Řikovský[*] and Štefan Kozák[**]

[*]Institute of Control and Industrial Informatics, Slovak Technical University, 812 19 Bratislava, Slovakia vrikovsky@gmail.com.

[**]Institute of Applied Informatics, Slovak Technical University, 812 19 Bratislava, Slovakia (e-mail: stefan.kozak@stuba.sk).

Abstract

The kinematics problems of redundant robots have been investigated for many years. Plenty of different applications for robot redundancy were implemented with success. Some of them were: improvement of redundant robot manipulability, robot obstacle avoidance, robot energy consumption optimization etc. Widely used methods use conventional approaches as for example in case of manipulability enhancement is used the gradient computation. However, the computational effort of these approaches brings many difficulties when it is used with other constraints. The solution to these problems is implementation of new intelligent methods based on artificial neural networks. This paper deals with application such methods where CMAC and HCMAC (Hierarchical Cerebellar Model Arithmetic Controller) neural networks were used in redundancy control of 7 DOF redundant manipulator. And manipulability enhancement constraint was chosen as redundancy constraint. First tested neural network was conventional CMAC neural network with supervised learning. It performed well in the terms of fast learning and local generalization capability. Nevertheless, the conventional CMAC has showed enormous memory requirement. Another tested neural network for the same task was HCMAC. It is shown that HCMAC perfectly approximated the testing function with relatively fast learning.

Keywords: robot, neural network, kinematics, Cerebellar Model, CMAC network, learning methods

1 INTRODUCTION

Today, robotic manipulators are important devices in many different industries. The control of manipulator involves trajectory planning, inverse kinematics and dynamics etc. Among these control problems the kinematics control of redundant manipulator is the interest of this paper. [1] Kinematical redundant robots are of special interest due to their redundancy that can be used for additional tasks: avoiding obstacles, singularities, manipulability enhancement etc.

The inverse kinematics of redundant robots can be solved with many different approaches. One of the main algorithms widely used is the Jacobian Pseudoinverse algorithm. This algorithm allows the manipulator to satisfy the additional constraints through mapping the velocities corresponding to the additional constraints to the null space motions, while end-effector tracks the desired trajectory. Additional constraints can be either computed using conventional optimization techniques or using artificial neural networks. Between different types of neural networks CMAC neural network fits best the requirements.

CMAC neural-network was first introduced by J. Albus in 1975 [2]. With its fast learning, good generalization capability, and easy of implementation by hardware CMAC has been applied in many real-world applications such as robotic control, signal processing, pattern recognition etc. [3-5].

Previous CMAC studies have mainly focused on how to develop CMAC learning algorithms [6], improve the CMAC topology structure [7], and select learning parameters [8]. Besides the convergence property has also received considerable attention, when Wong and Sideris [16] proved that CMAC's learning always converges with arbitrary accuracy on any sets of training data. Moreover, it was proved that CMAC learning results in a least square error if the number of iteration approaches to infinity and the learning rate approaches to zero. When the system requires derivative information of input and output variables, nonconstant differentiable Gaussian basis function was applied to CMAC. Lane *et al.* [7] developed a higher order CMAC neural network by using B-Spline receptive field function in conjunction with a general CMAC weight addressing scheme.

Nevertheless, CMAC based on Albus model has two major limitations: enormous memory requirement for resolving high-dimensional problems and difficulty in adjusting memory structure parameters [15]. The memory size requirement limits application fields of CMAC in real-world applications. In many cases this problem of memory requirements can be solved with hash-coding. This method associates several hypercubes to the same weight. In such a case, although the memory requirements can be reduced, it may frustrate convergence Moreover it may also affect the speed of convergence and degenerate the behavior of convergence.

To reduce enormous memory requirement, Lin and Li [8] presented a unique learning structure composed of small two-dimensional CMACs to solve high-dimensional problems. However, their proposed structure has its limitations. Many parameters have to be obtained heuristically to get a good learning capability for nonlinear functions. This results to high dependence on parameter determination and if the architecture is not properly considered the network does not work well.

In this paper we propose HCMAC (Hierarchical CMAC) neural network. This type of neural network can effectively overcome the enormous memory requirement problem in original CMAC model, because the new structure can partition high-dimensional problems into smaller two-dimensional subproblems. Also presented herein is gradient-descent learning rule to train the proposed HCMAC model. To solve the problem of redundancy control with artificial neural networks, both CMAC and HCMAC neural networks were applied to model the manipulability enhancement function. The results of this application were then compared in several criteria: memory requirements, speed of computation, accuracy.

This paper is divided into 5 sections, where the section II describes the inverse kinematics and manipulability measure. In the section III is shown the structure of CMAC and HCMAC neural network. Section IV presents the verification results of these neural networks on modeling manipulability gradient of 7 DOF redundant manipulator model. And the last section V summarizes all achieved results.

2 REDUNDANT ROBOT KINEMATICS

A Pseudoinverse kinematics

Since the manipulator is redundant $(n > m)$, the Jacobian matrix is not square. Kinematics equation is solved by pseudoinverse of the Jacobian matrix that locally minimize the norm of joint velocities. Kinematics equation is then transformed in:

$$\dot{q}(t) = J^{+}(q(t))\dot{x}(t) \tag{1}$$

with the pseudoinverse Jacobian matrix:

$$J^{+} = J^{T}(JJ^{T})^{-1}. \tag{2}$$

Furthermore,

$$\dot{q}(t) = J^{+}(q(t))\dot{x}(t) + (I - J^{+}J)\dot{q}_a(t) \tag{3}$$

where $\dot{q}_a$ is a vector of arbitrary joint velocities projected in the null-space of J. The vector $\dot{q}_a$ specifies the additional constraints for the redundancy. For the Jacobian pseudoinverse computation is the Moore-Penrose pseudoinverse method used.

B Redundant robot manipulability

As it is mentioned before, robot joint variables are denoted by n-dimensional vector q. It is then considered the set of all realizable end-effector velocities which Euclidean norm of $\dot{q}$,

$$\|\dot{q}\| = (\dot{q}_1^2 + \dot{q}_2^2 + ... + \dot{q}_n^2)^{1/2}, \tag{4}$$

satisfies $\|\dot{q}\| \leq 1$. This set is an ellipsoid in the m-dimensional Euclidean space (*m* is the dimension of end-effector position and orientation vector). In the direction of the ellipsoid major axis the end-effector can move at high speed. On the other hand, in the direction of the minor axis the end-effector can move only at low speed. If the ellipsoid is almost a sphere, the end-effector can move in all directions uniformly. This ellipsoid is called manipulability ellipsoid.

One of the representative measures for the ability of manipulation derived from the manipulability ellipsoid is volume of the ellipsoid. This is given by $c_m w$ where c_m is constant and:

$$w = \sigma_1 \sigma_2 ... \sigma_m \tag{5}$$

where the scalars $\sigma_1, \sigma_2, \ldots, \sigma_m$ are singular values of J. Variable w is called the manipulability measure for configuration q of the manipulator. The manipulability measure w has the following properties [11]

I. $w = \sqrt{\det(J(q)J^T(q))}$ (6)

II. When $m = n$, that is when we consider non-redundant manipulators, the measure w reduces to:
$w = |\det(J(q))|$ (7)

III. Generally $w \geq 0$ holds, and $w = 0$ if and only if $rank(J(q)) \leq m$, that is the manipulator is in a singular configuration. From this fact we can regard the manipulability measure as a kind of distance of the manipulator configuration from singular ones.

The increase of the manipulability measure during the robot movement can't influence the desired end-effector movement. For this purpose vector $\dot{q}_a$ is used, because it is projected with $(I - J^{+}J)$ into the null space and it doesn't affect the end-effector motion. If the null space motion of the robot should increase the manipulability measure the vector $\dot{q}_a$ has to be equal to:

$$\dot{q}_a = \nabla H(q) \tag{8}$$

where $H(q)$ is the objective function (manipulability measure) in the optimization process, and $\nabla H(q)$ is the gradient of the manipulability measure w.

3 CMAC and HCMAC neural network

To derive the HCMAC neural network, we first briefly introduce the CMAC neural network based on different basis function.

A CMAC neural network

CMAC neural network is a look-up table neurocomputing technique to approximate any nonlinear function with fast learning and good local generalization. Part of the neural network is basis function that influences the behavior of the network. The basic type commonly used is the constant basis function that is usually implemented in association memory selection.

Before applying basis function, the input data of each state variable must be quantized into discrete regions. The number of discrete regions is termed as a resolution. Each input data can be mapped to several actual memory units via an association memory selection vector.

These mapped actual memory units are called hypercubes. That means each input data is mapped on several different layered hypercubes.

The actual output for an input state is obtained as the sum of stored contents for hypercubes covering the state. That is, the actual output of a specific input state *s* can be mathematically expressed as follows:

$$y(s) = a^T(s)w = \sum_{j=1}^{N_h} a_j(s)w_j \qquad (9)$$

where $w=[w_1, w_2,\ldots,w_{Nh}]^T$ is the vector of actual memory contents, N_h is entire memory size, $a^T=[a_1(s), a_2(s),\ldots, a_{Nh}(s)]$ is the memory selection vector.

The conventional CMAC uses a supervised learning approach to adjust the weight values during each learning cycle. Its learning rule can be described as follows:

$$w_t = w_{t-1} + \frac{\alpha}{N_e} a(s)(\hat{y}(s) - a^T(s)w_{t-1}) \qquad (10)$$

where w_t is the vector of actual memory contents at time *t*, w_{t-1} is the vector of actual memory contents at previous time *t-1*, α is learning rate, $\hat{y}(s)$ is the desired output value, and $\hat{y}(s) - a^T(s)w_{t-1}$ is the error for the input training state *s*.

The nondifferentiable property leads to some limitations when using conventional CMAC in real-world application. Therefore, the constant basis function is replaced by differentiable function as for example Gaussian basis function. Mathematical formulation of one-dimensional Gaussian basis function ϕ is described as follows:

$$\phi(s) = e^{-((s-m)/\sigma)^2} \qquad (11)$$

where *m* is a hypercube center, σ is a hypercube radius, and *s* is a specific input state. The output of CMAC neural network with Gaussian basis function with N_v dimensional problem is revised to be as follows:

$$y(s) = \sum_{j=1}^{N_h} \left[a_j(s).w_j.\left(\prod_{i=1}^{N_v} e^{-((s_i - m_{ji})/\sigma_{ji})^2} \right) \right] \qquad (12)$$

where $a_j(s)$ is the j^{th} element of association memory selection vector for a specific input space *s*, w_j is the j^{th} memory allocation of actual memory, s_i is the input value of i^{th} dimension for a specific input state *s*, m_{ji} is the corresponding hypercube center, and σ_{ji} is the corresponding hypercube radius. The CMAC with a Gaussian basis function as a nonconstant differentiable basis function is termed as GCMAC.

B HCMAC neural network

HCMAC neural network consists of two-dimensional differentiable GCMACs. Fig. 1 illustrates the smallest topology of the HCMAC neural network, indicating that each GCMAC includes two input values, and the output values of first-layer GCMACs serve as input values of the second-layer

GCMACs.

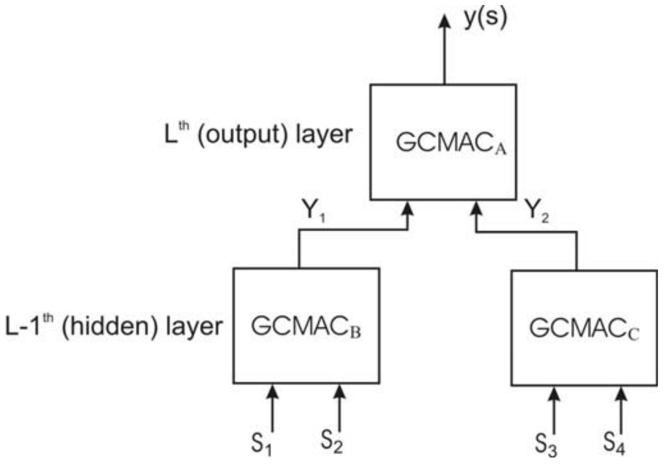

Fig. 1 Topology of HCMAC neural network structure

In Fig. 1 the structure of HCMAC contains four inputs s_i where $s_i(i=1,2,\ldots,4)$ represents i^{th} input value, $y_j(j=1,2)$ is the j^{th} output of the hidden layer, and $y(s)$ is the output of the whole HCMAC neural network for a specific input state *s*. HCMAC neural network splits problem of four input values into a problem of two two-dimensional GCMACs.

The learning method for training HCMAC neural network was gradient descent method. At first, the error cost function *E* is defined as:

$$E = \frac{1}{2}(\hat{y}(s) - y(s))^2 \tag{13}$$

where $\hat{y}(s)$ is the desired output value of HCMAC neural network for input state *s*, and *y(s)* is the actual output of the HCMAC neural network for this input state. In the GCMAC neural network equation are three parameters that have to be tuned during the learning process: weight *w*, radius σ and center *m*. First, the output layer GCMAC is trained, where the updates for the GCMAC parameters are as follows:

$$\begin{aligned}\Delta w_j &= -\frac{\alpha}{N_e}\frac{\partial E}{\partial w_j} \\ &= \frac{\alpha}{N_e}(\hat{y}(s) - y(s)).a_j(s).\prod_{i=1}^{2} e^{-((y_i - m_{ji})/\sigma_{ji})^2}\end{aligned} \tag{14}$$

where Δw_j is the updated weight value of the j^{th} actual memory in the GCMAC$_A$, w_j is the weight value of the j^{th} actual memory, α is a learning rate, and N_e is the number of mapped hypercubes for input state *s*.

$$\begin{aligned}\Delta \sigma_{ji} &= -\frac{\alpha}{N_e}\frac{\partial E}{\partial \sigma_{ji}} \\ &= \frac{\alpha}{N_e}(\hat{y}(s) - y(s)).a_j(s).\left[\prod_{i=1}^{2} e^{-((y_i - m_{ji})/\sigma_{ji})^2}\right].\frac{2(y_i - m_{ji})^2}{\sigma_{ji}^3}\end{aligned} \tag{15}$$

where $\Delta \sigma_{ji}$ is the updated radius of the i^{th} dimension for j^{th} mapped hypercube of input state *s* in the

GCMAC$_A$, σ_{ji} is radius of the i^{th} dimension for j^{th} mapped hypercube of input state *s*.

$$\begin{aligned}\Delta m_{ji} &= -\frac{\alpha}{N_e}\frac{\partial E}{\partial m_{ji}} \\ &= \frac{\alpha}{N_e}(\hat{y}(s)-y(s)).a_j(s).\left[\prod_{i=1}^{2} e^{-((y_i-m_{ji})/\sigma_{ji})^2}\right].\frac{2(y_i-m_{ji})}{\sigma_{ji}^2}\end{aligned} \tag{16}$$

where Δm_{ji} is the updated center value of the i^{th} dimension for j^{th} mapped hypercube of input state *s* in the GCMAC$_A$, m_{ji} is radius of the i^{th} dimension for j^{th} mapped hypercube of input state *s*. The parameters in GCMAC$_B$ and GCMAC$_C$ are updated according to the conventional backpropagation rule, where using the derivatives information $\partial y/\partial y_1$ and $\partial y/\partial y_2$ is the error backpropagated to GCMAC$_B$ and GCMAC$_C$.

4 EXPERIMENTS

The goal of CMAC and HCMAC neural networks in this paper is to replace the conventional manipulability gradient computation function. In order to test the control system with these neural networks, the kinematics of 7-DOF redundant robotic system model was taken. The model of the robot is shown in Fig. 2.

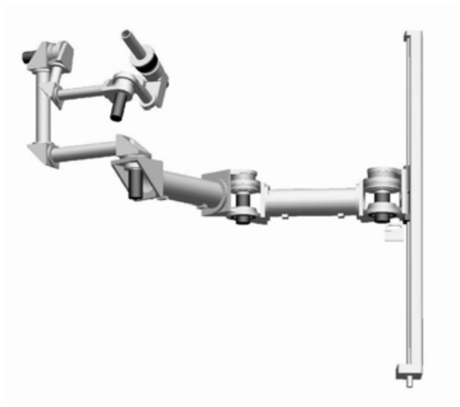

Fig. 2 Model of 7 DOF redundant manipulator

The control platform for testing neural networks was Euclidean space position control system; it can be seen in the Fig. 3. Input to the control system is the desired position and the output is actual robot end-effector position.

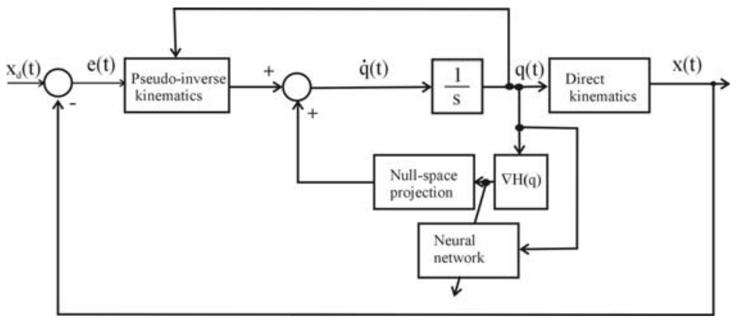

Fig. 3 Position control scheme with null-space motion and CMAC training

As can be seen from the figure the pseudoinverse kinematics (3) was used for computation the desired robot joint velocities. Apart this, the null-space motion is added for redundancy control. The optimization function for redundancy is manipulability measure. During the whole motion of the end-effector is neural network on-line trained. It learns the gradient values of manipulability measure. The neural network needs 7 input values (robot joint positions) and 7 output values. The conventional CMAC neural network allows multiple inputs and multiple outputs, where each output is computed from its own memory. On other hand, HCMAC neural network needs to implement appropriate GCMACs according to the number of inputs. It can be done either using full binary tree approach, or it can be optimized according to the input requirements. In this paper we have optimized the structure of full binary tree to get a minimum number of GCMACs. To implement 7 outputs to HCMAC structure the output GCMAC$_A$ was extended with 6 more memories to store the other 6 output values. Fig. 4 shows this structure with 7 inputs and 7 outputs.

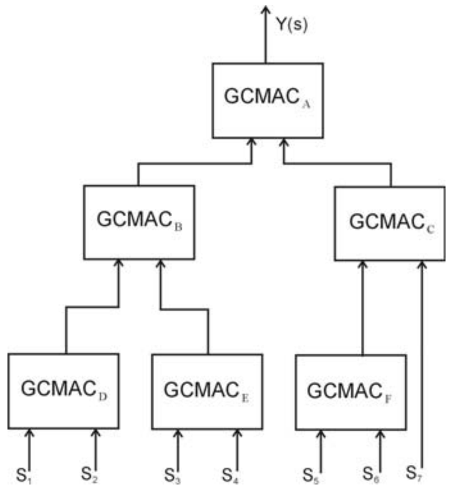

Fig. 4 Optimized HCMAC neural network structure for 7 inputs and 7 outputs.

As described earlier the control algorithm controls the robot to achieve the goal position from the starting position. During the motion are the redundant joints controlled to fulfil additional tasks. The end-effector starting position was set to (0.1455, 0.6588, 1.3896) and the goal position (0.25, 0.15, 0.5). The simulation time was set to 5 seconds.

A Simulation results

First tested neural network for manipulability modeling was the conventional CMAC neural network with Gaussian basis function. The learning process of this neural network was made online during the robot motion from starting point to finish point. After 10 training cycles CMAC replaced the gradient computation and the system was tested with this neural network. Fig. 5 shows the comparison of manipulability measure between CMAC neural network and original gradient computation. From the figure can be seen that CMAC neural network very good approximates gradient computation and therefore manipulability measure results are almost the same.

To measure the exact approximation performance an integral absolute error (IAE) was computed. It was defined as:

$$IAE = \int_0^T |w_1(t) - w_2(t)| dt \qquad (16)$$

where w_1 and w_2 are the manipulability measures for the case of CMAC and the original gradient computation.

Computed IAE in this simulation was 0.0744 what is 7.44*10^{-5} per one training sample. Time response of CMAC gradient computation and conventional gradient computation can be seen in Fig. 5. This figure shows that CMAC very good approximates all elements of the manipulability gradient vector. Each curve represents one element of gradient vector.

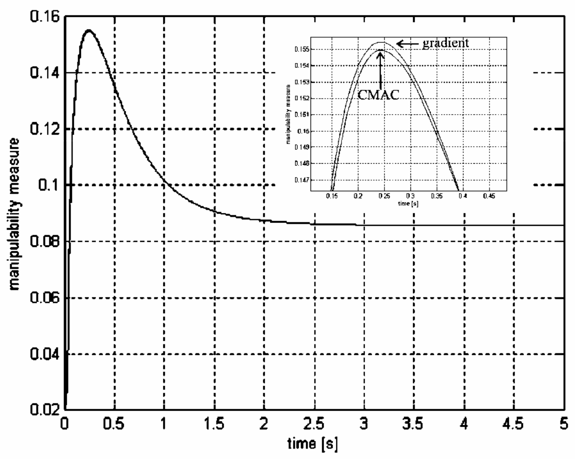

Fig. 5 Manipulability measure with CMAC gradient computation after 10 trainings

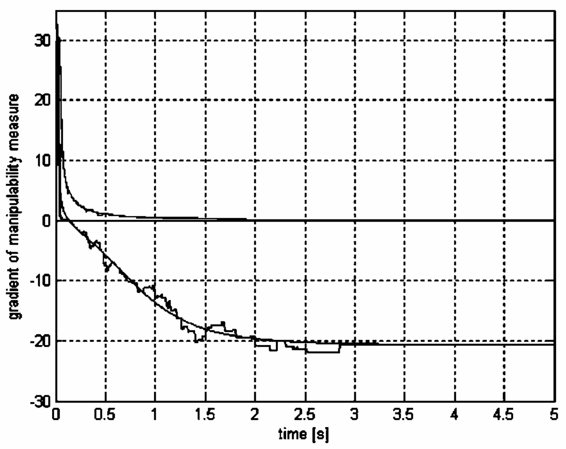

Fig. 6 Gradient computation comparison after 10 training cycles

The second tested neural network was HCMAC neural netowork. The same training samples were taken as in the case of CMAC neural network. After the training, HCMAC neural network also replaced the original gradient computation in the control scheme. Then it was compared to the conventional approach. Fig. 7 shows the results.

As can be seen from this figure HCMAC neural network approximates the manipulability measure better then CMAC neural network. The total computed IAE for this approximation was 0.0028, what is 2.8*10^{-6} per one training sample. The exact approximation of manipulability gradient vector is shown in Fig. 7. From this figure can be seen that all gradient vector elements were approximated better than in the case of CMAC. This figure also shows that HCMAC neural network output function is much smoother than CMAC neural network. That is, because of better distribution of 7 inputs between GCMACs.

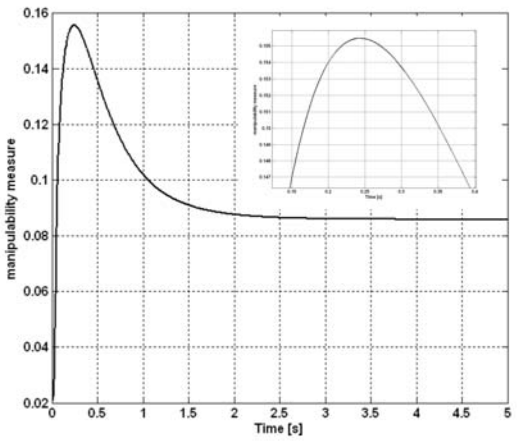

Fig. 7 manipulability measure with HCMAC gradient computation

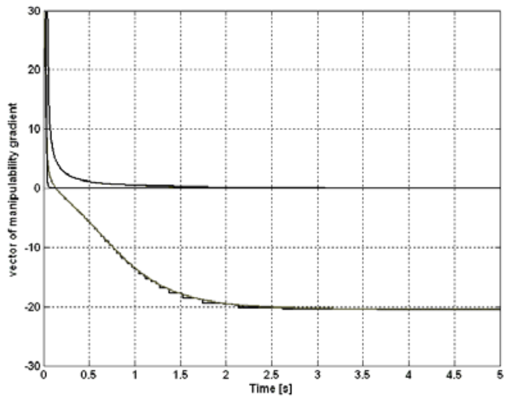

Fig. 8 Manipulability measure with CMAC gradient computation after 10 trainings

The next tested property was the speed of computation. The comparison was made between HCMAC, CMAC neural network and conventional gradient method. For this experiment were randomly chosen 100 and 1000 robot configurations for which the gradient was computed. For these configurations were computed the gradient outputs and the time of computation was measured. In table I. are presented the results of this testing.

Number of computations	Gradient in [s]	CMAC in [s]	HCMAC in [s]
100	0.1250	0.0310	0.09852
1000	1.0935	0.2580	0.86217

Tab 1. Gradient computation speed comparison

From the results can be seen that CMAC and HCMAC neural network computes the gradient faster than the original gradient method. This is an advantage in using neural network as an learnable alternative to the conventional computation.

CMAC or HCMAC neural network can be used as an alternative to the commonly used computation method. It can even compute the outputs faster than the original one. This can be effectively utilized in redundant robot kinematics control, where every spare of computation time is an advantage and can be used for other additional tasks. From the comparison of approximation properties can be seen than

HCMAC has better approximation results. Besides, HCMAC has lower memory requirements than CMAC. CMAC neural network was unable to implement with 7 inputs without using of hashing algorithm that reduced the memory requirements. But hashing algorithm brings errors in output computation and it shrinks the input space coverage.

HCMAC neural network covers the whole input space. The disadvantage of HCMAC using is that HCMAC neural network needed more computation and learning time than CMAC.

5 CONCLUSION

In this paper we proposed usage of CMAC and HCMAC neural network for manipulability gradient modeling. We tested possible replacement of the original gradient computation method. Inverse kinematics control simulation of 7 DOF redundant robot has shown, that CMAC and HCMAC can replace the original computation method with very good results. The neural networks were compared with conventional gradient method. Testing results showed us that HCMAC can better approximate gradient computation, but the computation time of one output value was higher than in the case of CMAC. Both neural networks showed that they can compute the gradient faster than the original method. Moreover, in cases where the redundant robot control system needs some spare of computation time, this can be the way how to succeed.

REFERENCES

[1] D. Crane,J. Duffy, "*Kinematic Analysis of Robot Manipulators*" Cambridge University Press, 1998.

[2] J. S. Albus, "A new approach to manipulator contro: The Cerebellar model articulation controller (CMAC)" *Trans. ASME J. Dynam. Syst., Meas. Contr*., vol. 97, no. 8, pp 220-227, 1975.

[3] J. Hu, J. Pratt, and G.Pratt, "Stable adaptive control of bipedal walking; Robot with CMAC neural networks," in *Proc. 1999 IEEE Int. Conf. Robot. Automat.,* vol. 2, 1999

[4] W. T. Miller and F.H. Glanz, "CMAC: And associative neural network alternative to backpropagation," in *Proc.IEEE,* vol. 78, pp. 1561-1567, Oct. 1990

[5] F. H. Glanz, W.T. Miller, and L.G. Kraft, "An overview of the CMAC neural network," in *Proc. 1991 IEEE Neural Networks Ocean Eng.,* 1991, pp. 301-308

[6] N.E. Cotter and T. J. Guillerm, "The CMAC and a theorem of Komogorov,".*Neural Networks*, vol. 5, pp. 221-228,1991

[7] S.H. Lane, D. A. Handelman, and J. J. Gelfand, "Theory and development of higher-order CMAC neural networks," *IEEE Contr. Syst. Mag.,* vol. 12,pp. 23-30,1992.

[8] C.S. Lin and C.K. Li, "A sum-of-product neural network (SOPNN)," *Neurocomput.,* vol. 30, pp. 273-291,2000.

[9] J. Hu and F. Pratt, "Self-organizing CMAC neural networs and adaptive dynamic control," in *Proc. 1999 IEEE Intel. Contr/Intell. Syst. Semiotics,* 1999, pp. 259-265.

[10]C.S. Lin and C.K. Li,"A memory based self generated basis function neural network," *Int. J. Neural Syst.,* vol. 9, no. 1,pp. 41-59, Feb. 1999.

[11]L. Sciavicco, B. Siciliano, "*Modeling And Control of Robot Manipulators.*", The McGraw-Hill Companies, Inc., 1996

[12]Y. Nakamura , "*Advanced Robotics: Redundancy and Optimization,.*" Addison-Wesley Publishing Company, Inc., Reading, Massachusetts. 1991

[13]Y. Li, S.H. Leong, "Kinematics Control of Redundant Manipulators Using CMAC Neural Network," *The 5th World Multiconference on Systemics, Cybernetics and Informatics (SCI2001).* pages 274-279., Orlando, USA. 2000

[14]T. Nanayakkara, K. Watanabe, K. Kiguchi, K. Izumi, ”Evolving Obstacle Avoidance Skill of a Seven-Link Redundant Manipulator Subjected to End-Effector Working Constraints,” *Proc. of IIZUKA 2000 (6th International Conference on Soft Computing)*, pages 684-689, 2000

[15]C.S. Lin amd H. Kim, “Selection of learning parameters for CMAC-based adaptive critic learning,” *IEEE Trans. Neural Networks*, vol. 6, pp. 642-647, May 1995

[16]Y.F. Wong and A. Sideris, “Learning convergence in cerebellar model articulation controller,” *IEEE Trans. Neural Networks,* vol. 3, pp. 115-121, Jan. 1992.