2015 IEEE Conference on Control Applications (CCA)
Part of 2015 IEEE Multi-Conference on Systems and Control
September 21-23, 2015. Sydney, Australia

Reinforcement learning neural network (RLNN) based adaptive control of fine hand motion rehabilitation robot

Xianwei Huang[1] and Fazel Naghdy[2], Haiping Du[2], Golshah Naghdy[2], Catherine Todd[2]

Abstract— **Recent neural science research suggests that a robotic device can be an effective tool to deliver the repetitive movement training that is needed to trigger neuroplasticity in the brain following neurologic injuries such as stroke and spinal cord injury (SCI). In such scenario, adaptive control of the robotic device to provide assistance as needed along the intended motion trajectory with exact amount of force intensity, though complex, is a more effective approach. A critic-actor based reinforcement learning neural network (RLNN) control method is explored to provide adaptive control during post-stroke fine hand motion rehabilitation training. The effectiveness of the method is verified through computer simulation and implementation on a hand rehabilitation robotic device. Results suggest that the control system can fulfil the assist-as-needed (AAN) control with high performance and reliability. The method demonstrates potential to encourage active participation of the patient in the rehabilitation process and to improve the efficiency of the process.**

I. INTRODUCTION

Enhancement and restoration of impaired fine motor skills caused by stroke is a complex process. This is due to the nature of fine motion that consists of small, precise, coordinated movements of the fingers and requires integration of muscular, skeletal and neurological functions. The focus of the previous studies of upper extremity therapy has been mainly on proximal parts: the shoulder, wrist, and elbow. Whereas, fine motor skills are of great importance in performing delicate activities of daily life (ADLs) such as eating, drinking and handwriting.

In this study, an innovative end-effecter rehabilitation robotic device named Amadeo (Tyromotion GmbH, Graz, Austria)[1], [2] designed for rehabilitation of fine hand functions is deployed. Aamdeo has 5 degrees of freedom (DOF) and is fully capable of providing position-based continuing passive mode (CPM) training, as well as active assistive modes with visual and audio feedback during computerized interactive games emphasizing flexion and extension of each finger. During training, the device is attached to the finger tips using a small magnetic disc and cohesive tape for connection with the robot. The finger slides can transfer bend or stretch movements to all the fingers at once or independently as shown in figure 1.

The major limitation of the Amadeo standard training protocol is that it provides position control with constant assistive force intensity regardless of the actual need of the patient. In clinical practice, the constant assistive force intensity can be excessive at some point during the movement of the finger. As a result, the patient may end up with just following the finger slides passively, and this makes the training task less challenging and thus compromise the training effectiveness. In order to achieve optimal training results and to reduce the required training sessions and associated costs, dynamic assistive force intensity, known as adaptive assist-as-needed (AAN)[3] control seems a desirable solution.

Fig. 1. Hand training with Amadeo

This study proposes a new AAN control method that provides extension/flexion assistance force by dynamically changing the intensity of the applied force as needed by a patient during training. The remainder part of the paper is structured as follows. The control strategy, especially high-level RL control theoretical base is explained in section 2. Details on the actor-critic based RLNN controller design is given in section 3. Section 4 covers validation method using computer simulation and experimental work. Finally, conclusion is presented in section 5.

II. CONTROL STRATEGY

The control strategy consists of the following steps : (1) Amadeo device stays in free mode and continuously checks if the position of any of the sliding actuators is changed as the result of the force applied by the subject. As soon as a position change is detected, the algorithm tracks the movement of the corresponding finger and predicts the next position of the finger and the intensity of the force applied by the finger. (2) If there is no change of position, the detection of force on any of the actuators means that the subject is trying to move the slides but the force is not adequate. The algorithm generates the amount of force required to move the slides. If no force detected, this implies that the subject is not able to apply any force on the slide. Hence, the algorithm generates just enough force to compensate the friction force

*This work is supported by China Scholarship Council (CSC)

[1]Xianwei Huang is with School of Electrical, Computer, and Telecommunications Engineering, University of Wollongong, NSW, 2522, Australia xh962@uowmail.edu.au

[2]Fazel Naghdy, Haiping Du, Golshah Naghdy and Catherine Todd are with School of Electrical, Computer, and Telecommunications Engineering, University of Wollongong, NSW, 2522, Australia

to actuate the slides. (3) If the force intensity applied by the subject is adequate to move the slides, the subject operates independent from the control algorithm.

A. Low level control

The proposed control strategy is implemented through a hierarchical control system consisting of a low level PID controller and a high level actor-critic based reinforcement learning neural network (RLNN) controller. The PID controller is designed to provide a stable force feedback control. The RLNN block regulates the desired force intensity and adapts the actor and critic neural network weights in order to compensate for the force input under nonlinear dynamic of the system and uncertainties. Fig. 2 shows the control structure for real-time implementation of the algorithm.

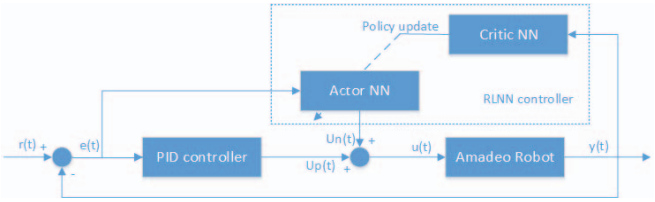

Fig. 2. Structure of RLNN and PID based control scheme

In Fig 2, the position and velocity signal are the input, denoted by r(t). y(t) representing position and force is the output of Amadeo robot. Up(t) and Un(t) are the output of the PID controller and RLNN controller respectively. Force control is achieved by the closed loop PID controller. Compared with low level force control, the high level force tracking and prediction is a more complicated task.

B. High level control

In the high level control, Amadeo acts as an agent in RL and learns from its interaction with the subject (environment) in order to provide assistive force intensity and range of motion (ROM) according to the needs of a subject (different case severity and capabilities). According to the agent-environment based RL structure [4], in this case, the action is the desired position and velocity for each finger slide, the state is the force and position of each finger, and the reward is the error between the predicted and actual output at current time step becomes smaller than that of the previous time step.

The prediction of the output is estimated through the model-free temporal difference (TD) learning method, which is a prediction method used for solving the reinforcement learning problem [5]. TD (λ) algorithm learns by reducing the error between estimates made by the agent at different times. The common approach in TD(λ) algorithm is the n-step reward estimation:

$$R_t^{(n)} = r_{t+1} + \gamma r_{t+2} + \cdots + \gamma^{n-1} r_{t+n} + \gamma^t V_t(s_{t+n}), \quad (1)$$

where γ is the discount factor, λ refers to the trace decay parameter, with $0 \leq \lambda \leq 1$. The estimate averages all the n-step estimates, each of which is weighted proportional to λ^{n-1}. The backup is toward the λ-return, defined by:

$$R_t^{\lambda} = (1-\lambda) \sum_{n=1}^{\infty} \lambda^{n-1} R_t^{(n)} \quad (2)$$

The actor-critic based RL algorithm is a part of the TD learning method of the RLNN controller. During the learning process, the critic NN generates an effective reinforcement signal and sends it to the actor NN, which learns and executes the behavioural output. The policy evaluation step is performed by observing the results of applying current actions and the state value function of each state is estimated from the reinforcement feedback [6]. The reward obtained from the transition is used to update predictions of the current state value and the preference of selection of the action next time step. The actor-critic method can be represented schematically as Fig. 3.

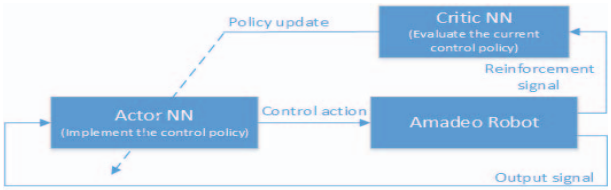

Fig. 3. Reinforcement learning scheme using critic-actor algorithm

Critic-actor algorithm makes a TD-update to the critic's value function V at state s_t is defined by:

$$V(s_t) := V(s_t) + \alpha_c(r_t + \gamma V(s_{t+1}) - V(s_t)), \quad (3)$$

where α_c is the learning rate for the critic NN. Similarily, α_a stands for the learning rate of the actor NN, the policy values of the actor NN at time t, indicating the tendency to select each action a_t when at each state s_t, are updated by:

$$P(s_t, a_t) := P(s_t, a_t) + \alpha_a(r_t + \gamma V(s_{t+1}) - V(s_t)), \quad (4)$$

The reinforcement at time step t is denoted by current reward r_t, and $\overline{V_t}$ is the correct prediction that is equal to the discounted sum of all future reinforcement signals, γ is the discounting factor. Namely:

$$\overline{V_t} = r_t + \sum_{i=1}^{\infty} \gamma^i r_{t+i} \quad (5)$$

The actor-critic based TD learning method works by selecting an action from the current policy, the critic NN and the actor NN are tuned online using the output of Amdeo, the weights of one NN are held constant while the weights of the other are tuned until convergence. This procedure is repeated until both NN have converged.

III. RLNN CONTROLLER DESIGN

A. NN function approximation

In RLNN controller, the weights of the neural network are used as function approximation to the value function. The

back propagation procedure is used to make the TD update according to the following equation for every output unit:

$$\Delta w_t = a(Y_{t+1} - Y_t) \sum_{k=1}^{t} \lambda^{t-k} \nabla_w Y_k, \tag{6}$$

where w_t is the vector of neural network weights being tuned, Y_t is the prediction for the output at time step t,$\nabla_w Y_k$ is a set of partial derivatives for each component of the weights w, α is the standard learning rate, and λ is the factor controlling how much future estimates affect the current update [7].

B. The critic NN

The critic NN generates a scalar evaluation signal which is used to tune the actor NN. The critic itself consists of an NN which approximates an evaluation function based on performance measurement. The tracking error $e(t)$ can be considered as an instantaneous utility function of the plant performance. The reinforcement signal R is defined as:

$$R = W_C^T \sigma(V_C^T e_n) + \psi, \tag{7}$$

where σ is the nonlinear activation function, W_C is the number of neurons in hidden layer of the critic NN, and e_n is the input to the critic NN, ψ is the auxiliary term. The critic NN weight update rule is a gradient-based adaption given by:

$$w_c(t+1) = w_c(t) + \Delta w_c(t) \tag{8}$$

$$\Delta w_c(t) = a_c(t) \left[-\frac{\partial E_c(t)}{\partial w_c(t)} \right], \tag{9}$$

where a_c is the learning rate of the critic NN at time t, which usually decreases with time to a small value, w_c is the weight vector of the critic NN, E_c is the objective function to be minimized by the critic NN. The output of the critic NN, the function approximates the discounted total reward.

$$R(t) = r(t+1) + a \sum_{t+1}^{k} r, \tag{10}$$

where $R(t)$ is the future reward-to-go value at time t, $(0 < \alpha < 1)$ is the discount factor. The binary reinforcement signal r is provided from the external environment and may be as simple as either 0 or -1 corresponding to success or failure respectively.

C. The actor NN

The principle in adapting the actor NN is to indirectly back propagate the error between the desired ultimate objective, denoted by E_A, and the approximate function from the critic NN. In the actor network, the state measurements are used as inputs to create a control as the output of the network. The weight estimation error for the actor NN is defined as:

$$\hat{e}_a(t) = w_a - \hat{w}_a(t) \tag{11}$$

The critic NN and the actor NN are tuned sequentially, with the weights of the other NN being kept constant. The weights in the actor NN are updated similarly to the ones in the critic NN.

$$w_a(t+1) = w_a(t) + \Delta w_a(t) \tag{12}$$

$$\Delta w_a(t) = a_a(t) \left[-\frac{\partial E_a(t)}{\partial w_a(t)} \right], \tag{13}$$

where $a_a > 0$ is the learning rate of the actor NN at time t, which usually decreases with time to a small value, w_a is the weight vector of the actor NN, E_a is the objective function to be minimized by the actor NN.

D. NN Learning

RLNN controller learning is to find an effective way to modify network connection weight matrix or network structure, thus to make sure that the control signal output of the NN controller follows the desired system output [8]. The on-line learning involves two major components in the learning system, namely the actor NN and the critic NN. In the following section, we devise the learning algorithms and elaborate on how learning takes place in each of the two modules. Both the actor NN and the critic NN are nonlinear multilayer feed-forward networks with one input layer, one hidden layer and one output layer in each network.

In the critic network, the approximating function output can be deduced from:

$$J(t) = \sum_{i=1}^{Nth} w_{ci}^2(t) Pi(t), \tag{14}$$

where Pi is the corresponding output of the hidden node,

$$Pi(t) = \frac{1 - \exp^{-qi(t)}}{1 + \exp^{-qi(t)}}, \tag{15}$$

the $i-th$ hidden node input qi can be deduced by:

$$qi(t) = \sum_{j=1}^{n+1} w_{cij}^{(1)}(t) x_j(t), \tag{16}$$

where x_j is the input of a connected j-th hidden node. The neurons in each layer are interconnected with their connections by weights. During the online training phase, the connection weights are adjusted while the input parameters and corresponding outputs are presented to the network for online learning. By applying the chain rule, the adaptation of the critic network is summarized as follows [9].
For the critic NN hidden to output layer:

$$\Delta w_c^{(2)}(t) = \alpha e_c(t) Pi(t) \tag{17}$$

For the critic NN input to hidden layer:

$$\Delta w_{cij}^{(1)}(t) = \alpha e_c(t) w_{ci}^{(2)}(t) \left[\tfrac{1}{2}(1 - p_i^2(t)) \right] x_j(t), \tag{18}$$

where $w_{ci}^{(2)}(t)$ stands for the weighting factor of the j-th neuron to the i-th neuron of the critic NN hidden layer, hidden layer neuron activation function takes positive and negative symmetry Sigmoid function.

In the actor NN, which is implemented by a feedforward network similar to the critic NN except that the inputs are the measured states and the output is the action $u(t)$. For the actor NN input to hidden layer:

$$\Delta w_{aij}^{(1)}(t) = e_a(t) \left[\tfrac{1}{2}(1 - u^2(t)) \right] w_{ai}^2(t) \left[\tfrac{1}{2}(1 - g_i^2(t) \right] x_j(t) \sum_{i=1}^{Nh} \left[w_{ci}^{(2)}(t) \tfrac{1}{2}(1 - p_i^2(t)) w_{ci}^{(1)}(t) \right], \tag{19}$$

for the actor NN hidden to output layer:

$$\Delta w_{ai}^{(2)}(t) = e_a(t)\left[\tfrac{1}{2}(1-u^2(t))\right] g_i(t) \sum_{i=1}^{Nh}\left[w_{ci}^{(2)}(t)\tfrac{1}{2}(1-p_i^2(t))w_{ci}^{(1)}(t)\right] \quad (20)$$

The above equations are implemented in Matlab code in section 4.2. The actor NN and the critic NN are both randomly initialized in their weights/parameters. Once a system state is observed, an action will be subsequently produced based on the parameters in the actor NN. During the training procedure, the weights of the network are periodically saved and tested, until no more improvement is observed [7].

IV. Simulation and Implementation

A. Simulation results

The simulation and online learning of the control system is developed using MATLAB/SIMULINK neural network toolbox, the program is written for the neural networks model based on TD (λ) algorithm. Firstly, we build the simulation model of Amadeo robot as shown in Fig.4. There are three variables in the Amadeo plant simulation model: input signals are position and velocity, while position and force signal are the output of the device. The constant friction is 2N according to the data sheet of the device.

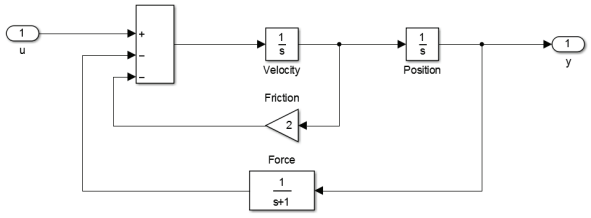

Fig. 4. Amadeo robot simulation model

In this RLNN controller, there is only one hidden layer and 10 neurons in both the critic NN and the actor NN, thus to ensure the low computing complexity and real-time property. The critic and actor NN are trained using back propagation as discussed in section 3, the reference input r is changed to a new value over the interval [-1, 1] stochastically. The time step t is set to 0.01s after several simulations for each variant to determine the optimal value of the RL parameters. During the online learning, the initial parameters of the Amadeo robot are listed as follows: position =0 (ranges from 0 to 100), and velocity=1 (ranges from 0 to 10). The learning process is repeated until it no longer improves the RLNN controller. The training error against the number of iteration is shown in Fig. 5.

It can be seen from Fig. 5 that the critic-actor modelled algorithm training error has been dropping with the increase of iterations, which means that difference between desired values and output values becomes smaller. The critic NN and the actor NN of the RLNN controller have the same structure and training parameters, with 10 neurons in the hidden layer, 1000 epochs, 0.05 learning rate and error value of 0.00005.

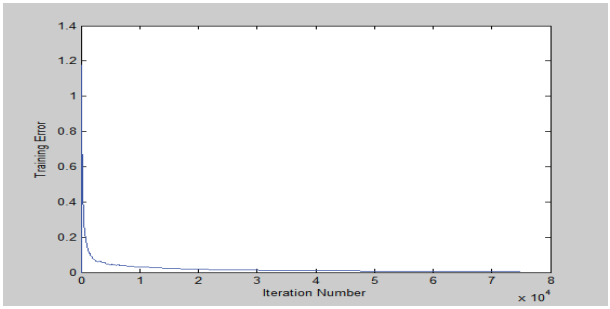

Fig. 5. Training errors vs iteration with 10 neurons in the hidden layer

Once the learning process is finished, the resultant policies with its corresponding parameters are updated. Fig 6 shows the critic NN control policy update difference and reward difference along with the iteration numbers.

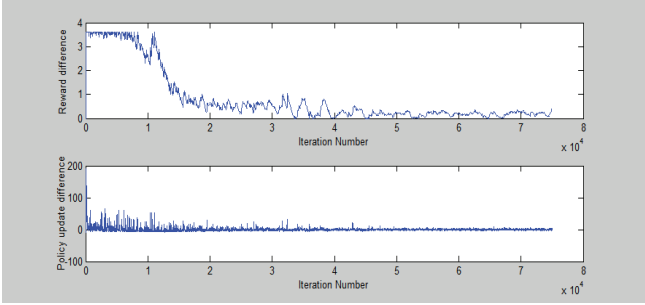

Fig. 6. Policy update difference and reward difference

As it can be seen from Fig 6, the x-axis shows the training epochs/iteration number and y-axis shows the reward difference and policy update difference. At the end of 20,000 iterations, the reward difference becomes less than one and the policy update difference becomes small enough after 20,000 iterations. This indicates that the updated control policy is gradually fitting the critic and actor NN. The comparison of the output of NN controller, PID controller and Amadeo output is shown in Fig 7.

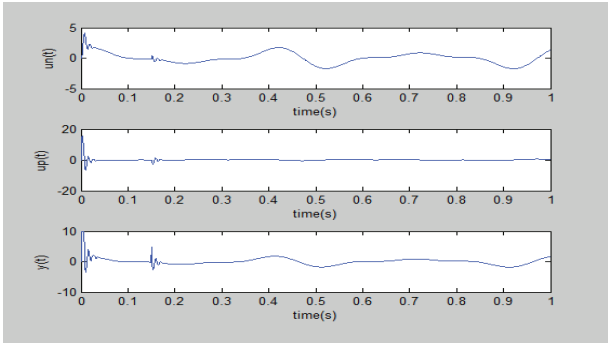

Fig. 7. comparison of the output of RLNN controller un(t), PID controller up(t)and Amadeo output y(t)

The performance of the RLNN controller step response is compared against the PID controller. The parameter of the PID controller is set to 200, 5 and 80 for proportional,

integral and derivative terms respectively. They are obtained through optimisation of PID controller to produce the best response for the system. The comparison between the responses of the RLNN and PID controllers are shown in Figure 8.

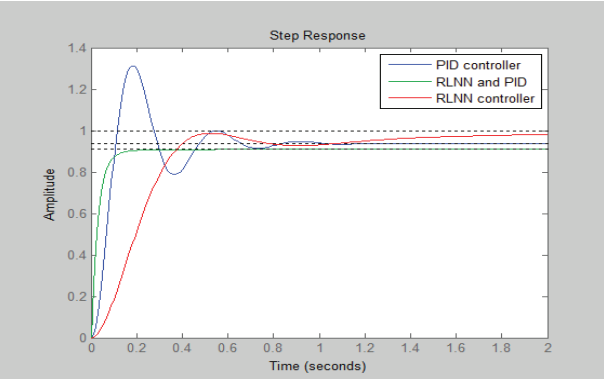

Fig. 8. Step responses of different controllers

In the step response experiment, the response of RLNN has no overshoot though slower than the PID controller. This is what is needed in Amadeo to prevent any excessive force on the subject's fingers.

B. Implementation results

Tyromotion GmbH, the maker of Amadeo provides a development environment based on LabVIEW that receives and executes external control signals in the form of position and velocity from a host machine and provides the current position and force produced by the actuators through a UDP communication protocol. The TD (λ) based RLNN control algorithm is coded in matlab using MATLAB Script and embedded in the LabVIEW environment, which provides accessibility to call custom-defined .m functions in LabVIEW files. This structure enables easy updating and revision of the control algorithm. The MATLAB Script in LabVIEW environment is shown in Fig.9.

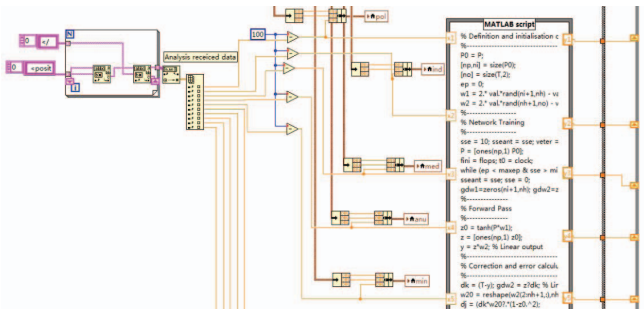

Fig. 9. Screenshot of RLNN MATLAB Script in LabVIEW environment

The front panel of the RLNN online learning LabVIEW program mainly consists of desired force intensity and the actual output as well as the error between them. The learning rate option enables input the learning rate manually or self-adaptive to generate the learning rate automatically within the pre-set Max and Min value. Thus, the learning rate for updating the hidden and output weights can be changed easily. The pre-set error value is the stopping criterion for online learning. The online learning virtual instrument (VI) is performed as shown in Fig. 10.

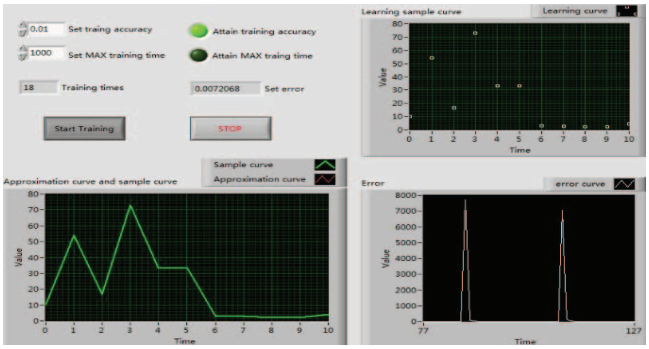

Fig. 10. RLNN controller online learning sub-VI

As shown, the pre-set training accuracy is attained at 6 seconds after the start of training. Furthermore, by using the online work of the virtual LabVIEW instrumentation, it is possible to choose the optimal values of the weight and biases matrix to obtain a smaller error after a number of iterations, as well as the number of neurons in each layer and the biases, weights, input matrix and the teaching gain.

The complete control system front panel is shown in Fig. 11. The real-time positon and force values of each finger can be found in bar chart on the left half, while the graph on the right half denotes the changing position and force information of the middle finger as an example. The sensitivity of the assistive force can be easily tuned by adjusting the five knobs for five fingers respectively, which makes the system suitable for the usage of different patients. The system is also featured with adjustable settings of easy switch between active and passive training mode.

C. Performance analysis

To validate the effectiveness of the proposed control method in full grasping/extending ROM and force intensity, the following tests are performed on a healthy subject with complete control ability of his hands and fingers. An initial set of position and velocity and the desired endpoint location were given as input to the network. Then, the current state force and position feedback at the end of each time step is used to update the inputs of the critic NN. Real-time assessment of force strength during full extension and grasping is shown in Fig.12.

As shown, the force intensity varies between -20 to 20N, where negative denotes the movement of extending and positive represents grasping. It can be seen that the actual output of force curve complies with the pattern of the estimated force in both directions, which means the RLNN controller is able to track and estimate moving trajectory of the finger. Also, the error between them turns to be smaller after 11 seconds, resulted from the online learning achieved by the RLNN controller. The provided assistive force is

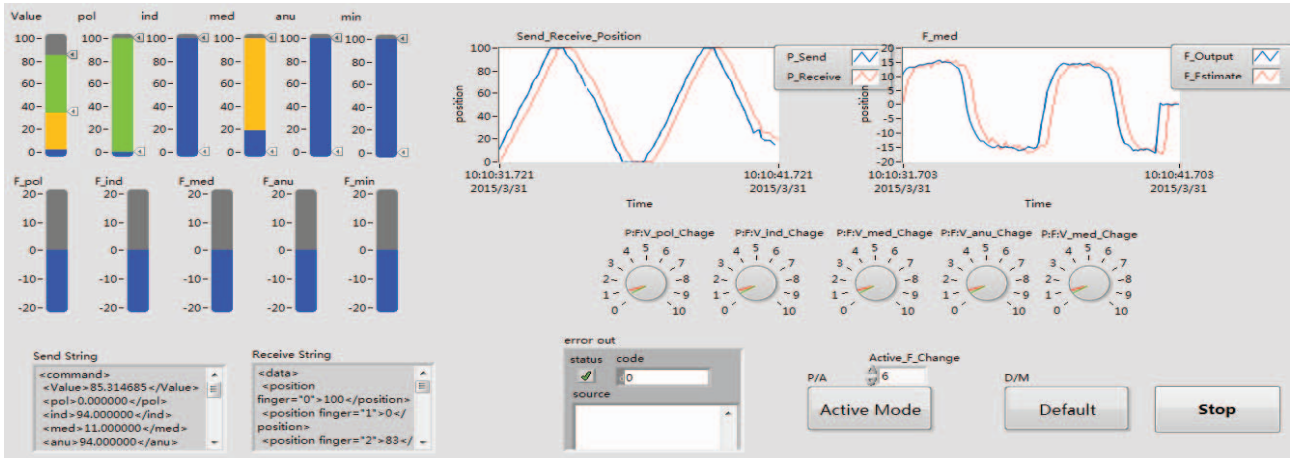

Fig. 11. LabVIEW front panel of the complete control system

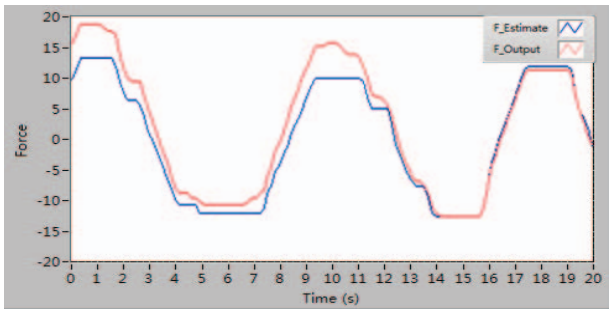

Fig. 12. Estimated and output force curve

roughly the same intensity of that is needed after the online learning process.

In implementation, according to the ROM of each finger, this position value ranges from 0 to 100, where grasping end (near end) is denoted 100 and extending end (far end) is 0. The result is shown in Fig. 13.

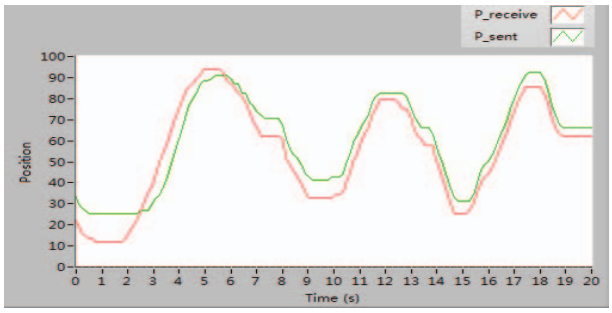

Fig. 13. Send and receive Amadeo output postion

In Figure 13, the output position of the control system is denoted as P_sent, while the actual output of Amadeo robot pattern is depicted as P_receive. The largest difference between these two values occurs around 1second after the start of the experiment. It can also be noticed that, compared with the total ROM of 80, the error suggests the proposed control method is able to assist the subject's movement in the intended trajectory within a timely manner.

V. Conclusion and Future Work

The study demonstrated that TD (λ) critic-actor architecture based RLNN controller can fulfil the control requirements of Amadeo robot. However, since the approach relies on low-resolution training information, a RLNN controller may require a large amount of online learning and experience to show significant improvement.

Meanwhile, extensive clinical experiments should be conducted on post-stroke patients before a firm conclusion can be drawn. Moreover, clinical evaluation and analysis on active participation, grasping/extending force, ROM, max frequency, duration time etc. should also be carried out.

References

[1] P. Sale, V. Lombardi, and M. Franceschini, "Hand robotics rehabilitation: Feasibility and preliminary results of a robotic treatment in patients with hemiparesis," *Stroke Research and Treatment*, vol. 2012, pp. 1–5, 2012.

[2] C. H. Hwang, J. W. Seong, and D. S. Son, "Individual finger synchronized robot-assisted hand rehabilitation in subacute to chronic stroke: a prospective randomized clinical trial of efficacy," *Clinical Rehabilitation*, vol. 26, no. 8, pp. 696–704, 2012.

[3] J. H. Daniel K Zondervan, Lorena Palafox, "The resonating arm exerciser: design and pilot testing of a mechanically passive rehabilitation device that mimics robotic active assistance," *Journal of NeuroEngineering and Rehabilitation*, vol. 10, no. 39, 2013.

[4] L. P. Kaelbling, M. L. Littman, and A. W. Moore, "Reinforcement learning: a survey," *J. Artif. Int. Res.*, vol. 4, no. 1, pp. 237–285, 1996.

[5] P. Dayan, "The convergence of td(λ) for general λ," *Machine Learning*, vol. 8, no. 3-4, pp. 341–362, 1992.

[6] F. L. Lewis, D. Vrabie, and K. G. Vamvoudakis, "Reinforcement learning and feedback control: Using natural decision methods to design optimal adaptive controllers," *Control Systems, IEEE*, vol. 32, no. 6, pp. 76–105, 2012.

[7] N. Papahristou and I. Refanidis, *Training Neural Networks to Play Backgammon Variants Using Reinforcement Learning*, ser. Lecture Notes in Computer Science. Springer Berlin Heidelberg, 2011, vol. 6624, book section 12, pp. 113–122.

[8] Y.-C. Chen and C.-C. Teng, "A model reference control structure using a fuzzy neural network," *Fuzzy Sets and Systems*, vol. 73, no. 3, pp. 291–312, 1995.

[9] S. Jennie and W. Yu-tsung, "Online learning control by association and reinforcement," *Neural Networks, IEEE Transactions on*, vol. 12, no. 2, pp. 264–276, 2001.