2017

Intelligent Learning Control System Design Based on Adaptive Dynamic Programming

Naresh Malla
South Dakota State University

INTELLIGENT LEARNING CONTROL SYSTEM DESIGN BASED ON ADAPTIVE DYNAMIC PROGRAMMING

BY

NARESH MALLA

A thesis submitted in partial fulfillment of the requirements for the

Master of Science

Major in Electrical Engineering

South Dakota State University

2017

INTELLIGENT LEARNING CONTROL SYSTEM DESIGN BASED ON ADAPTIVE DYNAMIC PROGRAMMING

NARESH MALLA

This thesis is approved as a creditable and independent investigation by a candidate for the Master of Science in Electrical Engineering degree and is acceptable for meeting the thesis requirements for this degree. Acceptance of this thesis does not imply that the conclusions reached by the candidates are necessarily the conclusions of the major department.

Zhen Ni, Ph.D.
Thesis Advisor Date

Steven Hietpas, Ph.D.
Head, Electrical Engineering and Computer Science Date

Dean, Graduate School Date

ACKNOWLEDGEMENTS

I would first like to express my gratitude to my thesis advisor, Dr. Zhen Ni for the inspiring guidance and support during the thesis tenure. Without his continuous support and feedback, I would not have been able to stand in this position in a short span of time. I would like to take this opportunity to thank Dr. Reinaldo Tonkoski for his problem-solving suggestions, continuous support, and encouragement during the joint project. I would like to thank Dr. Timothy M. Hansen, Dr. Robert Fourney, and Mr. Jason Sternhagen for their insightful comments and constructive criticism at different stages of my research. Moreover, I would like to thank Dr. Qiquan Qiao as a graduate coordinator for his support in pursuing my Master's degree by believing in me.

I would like to thank all professors and committee members for providing me valuable suggestions and encouragement. My special thank goes to Ujjwol and Dipesh for their cooperation in the joint project. Without their support in developing benchmark, validation of intelligent learning controller in the smart grid was not possible. Also, I would like to thank Lekhnath, Avijit, Tamal, Venkat, Abhilasha, Prateek, Rupak, Shuva, Fernando, Ayush, Bijen, Labi, Surendra, Prabin, Shiva, Prajina, and Maneesha for their valuable insight into my research work, and to all my friends who made this two years pass with joy.

CONTENTS

ABBREVIATIONS

ADP	Adaptive dynamic programming
CCVSI	Current controlled voltage source power inverter
CNN	Convolutional neural network
DFIG	Doubly-fed induction generator
DHP	Dual heuristic dynamic programming
DQN	Deep Q-network
EDV	Electric drive vehicle
FFT	Fast fourier transform
GDHP	Globalized dual heuristic dynamic programming
HDP	Heuristic dynamic programming
HPF	High pass filter
LPF	Low pass filter
MDP	Markov decision process
MIMO	Multiple-input multiple-output
MLP	Multi-layer perceptron
NN	Neural network
PER	Prioritized experience replay
PI	Proportional-integral
PID	Proportional-integral-derivative
PV	Photovoltaics

PWM	Pulse width modulation
RL	Reinforcement learning
RMS	Root-mean-square
RNN	Recurrent neural network
ROCOF	Rate of change of frequency
SAF	Shunt active filters
THD	Total harmonic distortion
UUB	Uniformly ultimately bounded
VSM	Virtual synchronous machines

LIST OF FIGURES

LIST OF TABLES

ABSTRACT

INTELLIGENT LEARNING CONTROL SYSTEM DESIGN BASED ON ADAPTIVE DYNAMIC PROGRAMMING

NARESH MALLA

2017

Adaptive dynamic programming (ADP) controller is a powerful neural network based control technique that has been investigated, designed, and tested in a wide range of applications for solving optimal control problems in complex systems. The performance of ADP controller is usually obtained by long training periods because the data usage efficiency is low as it discards the samples once used. Experience replay is a powerful technique showing potential to accelerate the training process of learning and control. However, its existing design can not be directly used for model-free ADP design, because it focuses on the forward temporal difference (TD) information (e.g., state-action pair) between the current time step and the future time step, and will need a model network for future information prediction. Uniform random sampling again used for experience replay, is not an efficient technique to learn. Prioritized experience replay (PER) presents important transitions more frequently and has proven to be efficient in the learning process.

In order to solve long training periods of ADP controller, the first goal of this thesis is to avoid the usage of model network or identifier of the system. Specifically, the experience tuple is designed with one step backward state-action information and the TD can be achieved by a previous time step and a current time step. The proposed approach is

tested for two case studies: cart-pole and triple-link pendulum balancing tasks. The proposed approach improved the required average trial to succeed by 26.5% for cart-pole and 43% for triple-link. The second goal of this thesis is to integrate the efficient learning capability of PER into ADP. The detailed theoretical analysis is presented in order to verify the stability of the proposed control technique. The proposed approach improved the required average trial to succeed compared to traditional ADP controller by 60.56% for cart-pole and 56.89% for triple-link balancing tasks. The final goal of this thesis is to validate ADP controller in smart grid to improve current control performance of virtual synchronous machine (VSM) at sudden load changes and a single line to ground fault and reduce harmonics in shunt active filters (SAF) during different loading conditions. The ADP controller produced the fastest response time, low overshoot and in general, the best performance in comparison to the traditional current controller. In SAF, ADP controller reduced total harmonic distortion (THD) of the source current by an average of 18.41% compared to a traditional current controller alone.

CHAPTER 1 INTRODUCTION

1.1 Background

In literature, reinforcement learning (RL) is defined as the learning about how to map situations to action taken by an agent so as to maximize the reward [1, 3]. A reinforcement learning task that satisfies the Markov property is called a Markov decision process (MDP). The traditional RL algorithms have great difficulty in evaluating the future optimal actions with huge searching space. Deep learning enables multi-layer processors/perceptrons to learn the data representation with multiple levels of encoding and decoding process. These methods improve the state-of-the-art pattern recognitions (e.g., speech, object, genomics and others) [7]. Deep Q network (DQN), one type of deep reinforcement learning algorithms, has been developed to mimic human and animals' decision making process through a combination of reinforcement learning and hierarchical sensory processing systems [6, 8–10]. It uses several layers of nodes to build up the mapping and representation of the data and it makes the neural network capable to learn concepts from raw sensory or image data (which is usually with very high-dimension state spaces). Backpropagation algorithm is one of the typical ways to show how such a model adjusts its weight parameters to represent the relationship between input and output [11]. Several different versions of backpropagation algorithms have also been developed to improve the convergent speed for large-scale input data stream [12]. Convolutional neural network (CNN) is another powerful technique to extract represented features and conduct end-to-end policy searching for 2-D images, yet may require certain computation at times [13, 14]. Interesting applications of deep reinforcement learning have been reported

in robot navigation and robot arm manipulation [15, 16]. Among all the reported results, the deep auto-encoder and certain training algorithms are generally used for this type of large searching space decision making problems. These frontier results also inspired the possible integration to other control and optimization fields.

One class of reinforcement learning methods is based on the actor-critic structure, where an actor generates an action and a critic component evaluates the performance. This principle motivates the development of a family of approximate or adaptive dynamic programming (ADP) designs. There are three typical structures in ADP design family, heuristic dynamic programming (HDP), dual heuristic dynamic programming (DHP) and Globalized dual heuristic dynamic programming (GDHP). They are mainly applied for online control and optimization problems. For example, in HDP design, it consists of an action and a critic networks. The critic network is designed to evaluate the current online learning performance based on the instant control action, while the action network produces an online learning based control signal to improve the control performance [2, 17–23]. DHP and GDHP are so-called advanced ADP designs, and are expected to learn and control faster and more accurate than the basic HDP design. HDP will be referred to as ADP hereinafter. Depending on whether it needs a model-network or not, ADP methods can also be categorized as model-free design [24–26] and model-based designs [27–30]. In recent years, event-based ADP [31] and data-based ADP [32] are developed to design the adaptive robust control for the nonlinear systems. Many of the theoretical and convergence analysis publications have reported to guarantee the optimal policy after the learning-interaction process [33–36]. However, learning/convergent speed and data efficiency are two of the challenges of applying ADP based methods in

real-world applications.

Currently, there has been a trend studying the advantages of experience replay technique to improve the data efficiency of ADP/RL methods from both theoretical and experimental perspectives [4, 37–39]. Classical ADP based control converges relatively slow, because it needs enough time to learn to interact with systems. The samples are used only once and then discarded right away. The controller could easily "forget" the previous experience after learning for a long time. Thus, the learning is not efficient and accurate after a long period of time. Some pilot studies with the integration of experience replay have been reported on ADP designs for dc motor, inverted pendulum, and zero-sum games [40, 41]. However, it is still an open question how to systematically facilitate the learning and association process of ADP based approaches in a general way and explore the various real applications.

1.2 Reinforcement Learning and Adaptive Dynamic Programming

In this section, we will first introduce the background and fundamental principles of Markov decision process, and then its optimal solutions based on reinforcement learning and adaptive dynamic programming.

1.2.1 Markov Decision Process

MDP model consists of a set of possible states (s_t), a set of possible actions (a_t), some reward function ($R(s_t, a_t)$), and a probability that an action $a_t = u(t)$ in state $s_t = x(t)$ will lead to next state $s_{t+1} = x(t+1)$. Note that in next chapters, x is used to denote state and u is used to denote control action. MDP must satisfy the Markov property that the effects of an action taken in a state is independent of the prior history and only

depends on that state. For given state, s, next state, s' and action a, the probability of each possible next state, known as the transitional probability can be defined in a particular finite MDP as [1]:

$$P^a_{ss'} = Pr\{s_{t+1} = s' | s_t = s, a_t = a\}. \tag{1.1}$$

The expected value of the next reward, r_{t+1} can be written as:

$$R^a_{ss'} = E\{r_{t+1} | s_t = s, a_t = a, s_{t+1} = s'\}. \tag{1.2}$$

The state-value function, V^π for an arbitrary policy π can be evaluated by cumulating future possible discounted reward:

$$V^\pi(s) = E_\pi\{r_{t+1} + \gamma r_{t+2} + \gamma^2 r_{t+3} + ... | s_t = s\} \tag{1.3}$$

$$= E_\pi\{r_{t+1} + \gamma V^\pi(s_{t+1}) | s_t = s\}. \tag{1.4}$$

where, r_{t+1}, r_{t+2}, r_{t+3}, ... are rewards received at time $t+1, t+2, t+3...$ and γ is the discount factor. The successive approximation $V(s)$ can be obtained by using the Bellman equation as an iterative update rule:

$$V_{t+1}(s) = E_\pi\{r_{t+1} + \gamma V_t(s_{t+1}) | s_t = s\}. \tag{1.5}$$

After following the existing policy π, we can consider selecting action a in state s. Then, the value of this way of behaving is:

$$Q^{\pi}(s,a) = E_{\pi}\{r_{t+1} + \gamma V^{\pi}(s_{t+1}) | s_t = s, a_t = a\}. \tag{1.6}$$

If $Q^{\pi}(s,\pi'(s)) \geq V^{\pi}(s)$, then policy π' must be as good as, or better than π. This is called policy improvement process.

1.2.2 Reinforcement Learning

Reinforcement learning is a type of learning in which learner or decision maker known as the agent interacts with its environment to take actions so as to achieve certain goal. The agent is naive at the beginning and is subjected to new situations from which it learns to make decisions so as to maximize cumulative reward from the environment over the time. This basic concept is also illustrated by Fig. 1.1 [1]. At each time step t, the agent observes the state of the environment, s_t, and based on that generates an action, a_t. Meanwhile, it will receive an instant reward r_t, indicating the control performance. The agent will adjust its parameters to maximize the reward in the future. This process is repeated along the time axis.

Almost all reinforcement learning algorithms are based on estimating value functions, that estimate how good it is to perform certain action in a certain state. The value function of a state s under an optimal policy, denoted by V^*, is the expected return and can be defined as:

$$\begin{aligned} V^*(s) &= \max_a E_{\pi^*}\{R_t | s_t = s, a_t = a\} \\ &= \max_a E_{\pi^*}\{r_{t+1} + \gamma \sum_{k=0}^{\infty} \gamma^k r_{t+k+2} | s_t = s, a_t = a\} \\ &= \max_a E\{r_{t+1} + \gamma V^*(s_{t+1}) | s_t = s, a_t = a\}. \end{aligned} \tag{1.7}$$

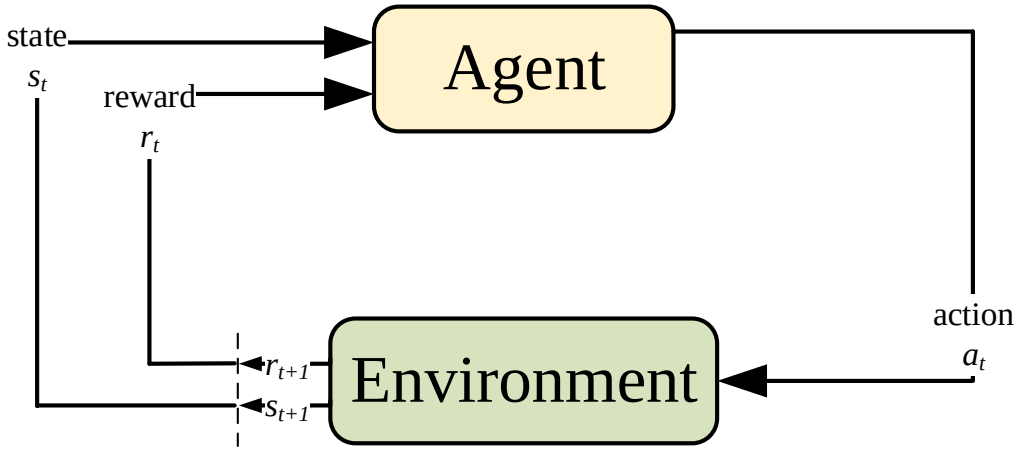


Figure 1.1. Interaction between agent and environment in reinforcement learning (this figure is reprinted from [1]).

where $E_{\pi^*}\{\}$ denotes the expected value if the agent follows the optimal policy π^*. The last equation is one of the form of Bellman optimality equation [42] for V^*. The Bellman optimality equation for Q^* is

$$Q^*(s,a) = E\{r_{t+1} + \gamma \max_{a'} Q^*(s_{t+1}, a') | s_t = s, a_t = a\} \tag{1.8}$$

Reinforcement learning algorithms are usually used to solve the discrete-time optimization problems with Markov properties. For years, Q-learning, SARSA (λ) and temporal difference (TD) algorithms are the most commonly used methods for maze navigation, path planning and other agent-environment interaction applications.

1.2.3 Adaptive Dynamic Programming

For continuous-time and continuous-state problem optimization with Markov properties, adaptive dynamic programming is one of efficient ways to solve it. Fig. 1.2 shows the actor-critic structure which is based on the reinforcement learning principle. In

this design, the critic and the actor are tuned online using the observed data containing state variable, reward and next state variable along the system trajectory. Neural networks are typically used to build the mapping of action/critic structure here. Weights of one Globalized neural network (NN) are kept constant while weights of other NN are tuned and the procedure is repeated until both NN have converged. After enough training, the actor produces the optimal control action and critic evaluates the performance of actor over time. Learning and interaction is the key that enables adaptive controller to converge to the optimal control. Neural network, fuzzy logic, and other computational intelligence techniques can be employed to implement the mapping of critic and action networks.

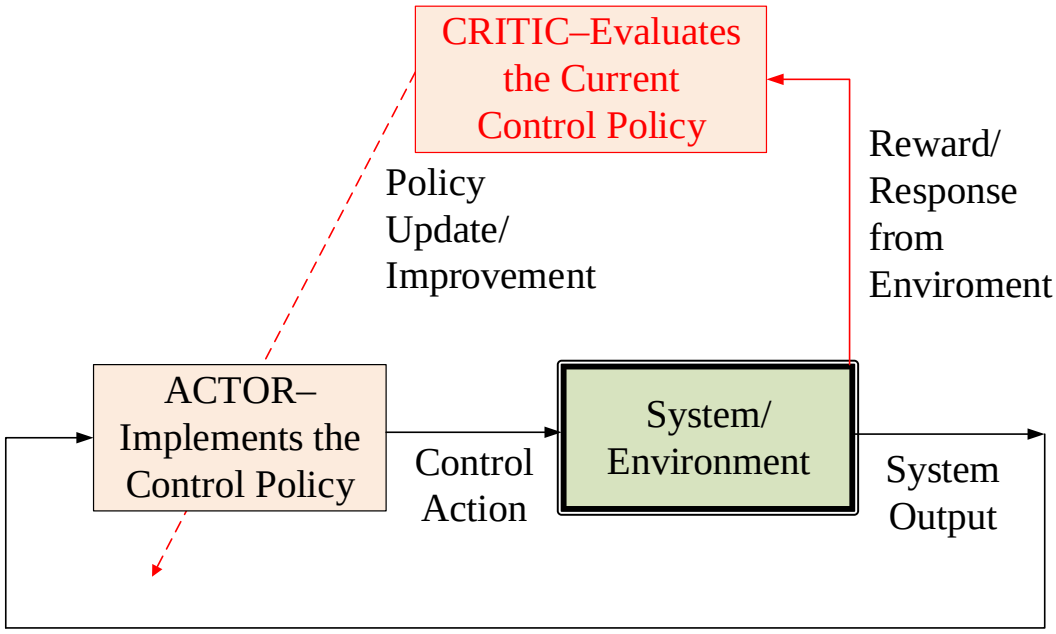


Figure 1.2. Reinforcement Learning with an actor/critic structure [2].

Heuristic dynamic programming is one of the basic ADP designs which only consists of action and critic networks. The architecture of critic network is a multi-layer perceptron (MLP) structure. The inputs to the critic network are the measured system

state vector, $s(t)$, and control action, $a(t)$. $J(t)$ is the output of the critic element and the J function approximates the discounted total reward to go.

Adaptive dynamic programming and reinforcement learning are both commonly used in the society for the optimization of decision making process. Though reinforcement learning can also deal with continuous time systems, it is usually required to discretize the state and action spaces before hand. This may possibly sacrifice accuracy or increase the computational load in some cases. The advantage of adaptive dynamic programming is to deal with continuous state system in continuous time fashion.

1.3 Computational Intelligence Techniques in Online Learning and Optimal Control

In this section, we will discuss the key components in deep reinforcement learning, e.g., deep autoencoder, convolutional neural network, and one of the important parameter to be used for algorithm development in next chapters: termed as experience replay.

1.3.1 Autoencoder and Deep Autoencoder

An autoencoder neural network is with MLP structure which has one or more hidden layers. The input is first encoded by hidden nodes which can be decoded to produce output.

The simplest example of autoencoder can be realized by 8-3-8 network as shown in Fig. 1.3 and Fig. 1.4, rooted from [3]. These results are generated from the authors' graduate course project: EE792: Computational Intelligence. In the network shown in Fig. 1.3, the eight network inputs are connected to three hidden units, which are in turn connected to the eight output units. Because of this structure, the three hidden units will be forced to represent the eight input values in some way that captures their relevant

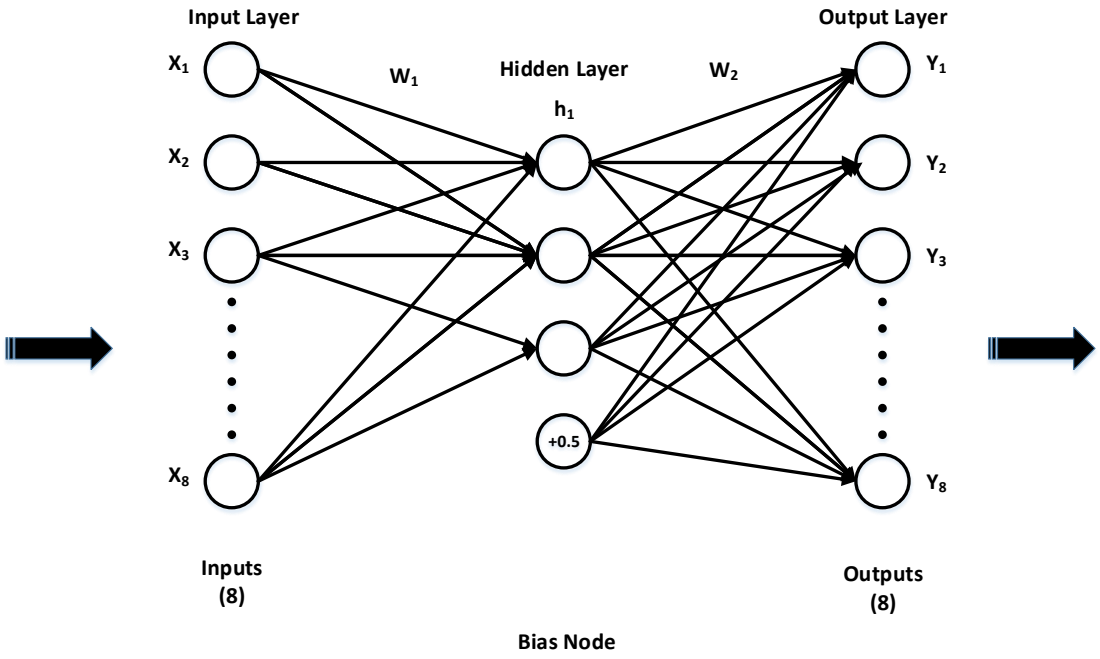


Figure 1.3. 8-3-8 neural network which can be used to learn input and output (this figure is reprinted from [3]).

features, so that this hidden layer representation can be used by the output units to compute the correct target values. The network shown in Fig. 1.3 can be trained to learn the simple output target function:

$$Output \;\; y = Input \;\; x. \tag{1.9}$$

where x is a vector containing binary values. The network must learn to reproduce the eight inputs at the corresponding eight output units. Although this is a simple function, the network in this case is constrained to use only three hidden units. Therefore, the essential information from all eight input units must be captured by the three learned hidden units. When backpropagation is applied to this task, the autoencoder successfully learns the

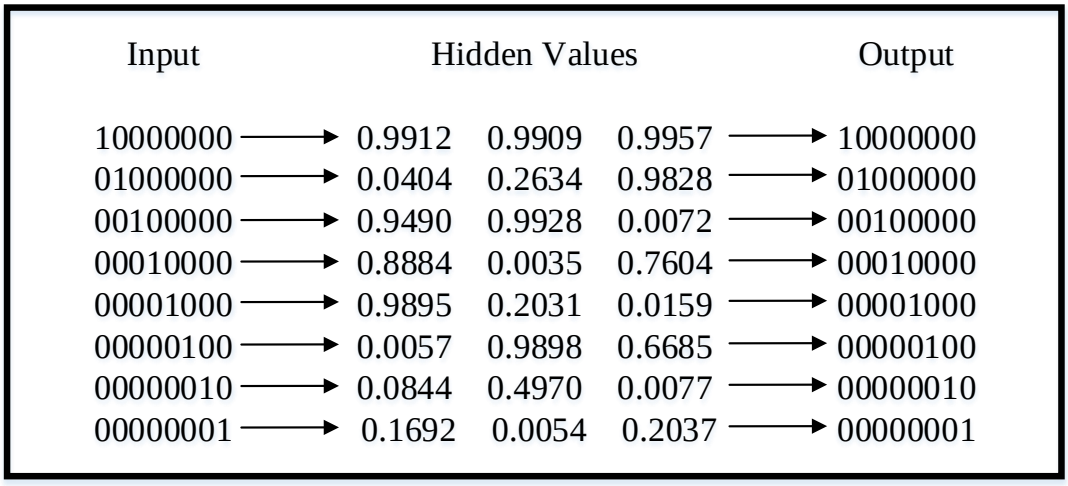


Figure 1.4. Learned hidden layer representation for 8-3-8 network

target function using the eight possible vectors as training examples. If the number of hidden layer increases, then the autoencoder is called deep autoencoder. Deep autoencoder finds many applications in image search [43], data compression [44], topic modeling and information retrieval [45]. The training algorithm and neural network structure may be different (e.g, Restricted Boltzmann Machine), however the principle remains the same in case of deep autoencoder [46].

1.3.2 Convolutional Neural Networks

Convolutional neural network (CNN) is a variation of MLPs which is mostly used in deep learning techniques. It consists of one or more convolutional layers followed by one or more fully connected MLP layers. The input to a convolutional layer is with input size as:

$$Input\ \ size = n \times n \times r. \tag{1.10}$$

where $n \times n$ is height and width of image in pixels and r=number of channels (e.g., r=3 for RGB image). The convolutional layer will have k filters of size as:

$$Filter \ \ size = m \times m \times q. \tag{1.11}$$

where $m \leq n$ and $q \leq r$. Within each layer of CNN, set of feature detectors exists, each of which responds to the presence of a particular pattern in an input tensor [47]. Each filter is replicated across the entire visual field and they share the same weights and bias to form a feature map which increases efficiency. Between layers, dimension reduction technique called max pooling layer [48] or strided convolutions [49] are utilized. Backpropagation algorithm can be used to compute the gradient with respect to the parameters of the model in order to use gradient based optimization [50].

1.3.3 Deep Q Network

Deep Q-network is an artificial agent with ability of recent advances in training deep neural network that can learn successful policies directly from high-dimensional sensory inputs using reinforcement learning [6]. Deep Q-network agent takes only the pixels and the game score as inputs and has learning capability to excel at a diverse array of challenging tasks. In case of deep Q-network, the compressed feature of deep autoencoder is treated as state and stored in experience replay. At every update iteration i, the current parameters θ are updated so as to minimize the mean-squared Bellman error with respect to old parameters θ, by optimizing the following loss function (DQN Loss):

$$L_i(\theta_i) = E[(r + \gamma \max_{a'} Q(s', a'; \theta_i^-) - Q(s, a; \theta_i))^2] \tag{1.12}$$

For each update i, a tuple of experience $(s,a,r,s') \sim U(D)$ (or a minibatch of such samples) is sampled uniformly from the replay memory D. For each sample (or minibatch), the current parameters θ are updated by a stochastic gradient descent algorithm. Specifically, θ is adjusted in the direction of the sample gradient g_i of the loss with respect to θ,

$$g_i = (r + \gamma \max_{a'} Q(s',a';\theta_i^-) - Q(s,a;\theta_i))\nabla_{\theta_i} Q(s,a;\theta) \tag{1.13}$$

Then, actions are selected at each time-step t by an ε-greedy behavior with respect to the current Q-network $Q(s,a;\theta)$ as described in [51].

1.3.4 Experience Replay

Experience replay (ER) technique delivers memory capacity for a learning agent to recall past experiences and apply them to update the current policy. Thus, high data efficiency is achieved by reusing the samples. The basic technique behind experience replay is to store all the experience tuple defined as,

$$e_t = (s,a,r,s') \tag{1.14}$$

where t refers to the time instance. The overall replay memory is defined as

$$D_t = \{e_1, e_2, ..., e_t\} \tag{1.15}$$

When training the neural network, random minibatches from the replay memory are sampled instead of the most recent transition to avoid local minimum. Experience replay is first proposed in [52], in which experience data are stored and chosen randomly to update the value function and policy in reinforcement learning for neural network approximation. As mentioned in [53], the advantages of experience replay are twofold, first it helps to increase the sample efficiency by allowing samples to be reused. On top of this, in the context of neural networks, experience replay allows for mini-batch updates which helps the computational efficiency, especially when the training is performed on a GPU. In addition, learning from mini-batch samples would cause the updates of the network parameters to have a low variance, leading to faster and potentially more stable learning. Second advantage is that the experiences used to train the networks are not only based on the most recent policy and ε can be used to scale the amount of exploration. If ε=1, the movement is completely random and if ε=0, the best policy is followed i.e., greedy behavior. Use of past experiences and past policy also avoids the algorithm getting stuck in a local minimum or diverging case [6].

1.4 Discussion of Opportunities for Deep Learning Enabled Adaptive Dynamic Programming

Deep reinforcement learning and experience replay methods open many new opportunities for adaptive dynamic programming in the continuous time and continuous state domain. “End-to-end” learning and control from raw images or sensory data to the optimized control action is also possibly solvable by the deep learning enabled ADP approach. Feature representation, deep auto-encoder, convolutional neural network, and

deep Q network are the possible necessary components for deep learning enabled ADP design. Currently, there are also a few pilot studies on applying experience replay to improve the learning and data efficiency of ADP approach for zero-sum optimization problem.

In [54], an actor-critic, model-free algorithm based on the deterministic policy gradient is presented that can operate over continuous action spaces. Using the same learning algorithm, network architecture and hyper-parameters, their algorithm robustly solved more than 20 simulated physics tasks, including classic problems such as cartpole swing-up, dexterous manipulation, legged locomotion and car driving. Similarly, [55] applied the deep Q-network on standard RL testing continuous as well as discrete domains, such as grid world, mountain car problem, and inverted pendulum problem. Training neural network with the recent data causes it to forget previous data and learning will be difficult. Thus, experience replay plays a vital role in convergence of neural network of deep reinforcement learning whose integration in ADP controller is at its pilot stage. The experience replay was integrated in [40] to update the weights of the action network and critic network in ADP approach [24] for optimal control of DC motor problem [56] and inverted pendulum.

1.4.1 Feasibility for Integration of Experience Replay for RL Structure

An integral reinforcement learning and experience replay algorithm is developed on an actor-–critic structure in [37] to learn online the solution to the Hamilton–Jacobi-Bellman equation for partially-unknown constrained-input systems. The near optimal solution convergence and stability was guaranteed in this paper with ER

based RL framework. The authors proposed one way to integrate experience relay based gradient-descent algorithm for tuning the weights of critic NN by using the current data and some past data (which are stored but can be removed and replaced) for learning the weights at each specific time. This can be observed in Eq. (28) and Eq. (42) of the paper [37]. They also stated that new data points can replace old ones in the history stack, only if a replacement increases the minimum eigenvalue and consequently increases the convergence rate of the critic NN weights. This implies that tuning weights of critic network with experience replay updating law improves the convergence speed. In the meawhile, the authors only used a single-layer NN for each actor-critic structure.

In [57], ER in actor-critic algorithm is proposed to address the issues of efficiency and autonomy that are required to make reinforcement learning feasible for real-world control tasks. Their experimental study with simulated octopus arm and half-cheetah demonstrates the practicability of the proposed algorithm to solve difficult learning control problems in a reasonable short time. A general ER framework is developed in [38] that can be combined with essentially any incremental RL technique, and instantiates this framework for the approximate Q-learning and SARSA algorithms. Authors also present promising real time learning results in inverted pendulum and robot systems, which are encouraging for real time applicability.

1.4.2 Maze Navigation Problem

Maze navigation problem is an interesting example for understanding deep reinforcement learning principle. The state of the environment in maze navigation problem example can be defined by the location of agent in the maze. It is possible to take

coordinate position as states but it may not be universal in other games. Thus, the authors in [12] take screen pixels containing sufficient information about maze navigation problem. Raw pixels are also used as input state of a reinforcement learning problem in using fitted Q-iteration to generate optimal policy. The first step in deep reinforcement learning is preprocessing the image and compressing it to lower dimension using deep autoencoder. Processing the benchmark dataset MNIST, a deep autoencoder as described in [46] could be trained and tested using the backpropagation algorithm as mentioned in [3]. In [12], the preprocessing technique is applied first: taking screenshot, resizing them to 30x30 pixels and converting to grayscale with 256 gray levels. 6200 of these images were taken, 3100 are used for training and 3100 for testing. These images are then feeded to 900-900-484-225-121-113-57-29-15-8-2-8-15-29-57-113-121-225-484-900-900 deep autoencoder structure. This deep autoencoder is retrained and fine-tuned until they get the reconstruction for testing images exactly same as input similar to method as shown in Fig. 1.5.

Deep reinforcement learning perfectly fits into this situation as we could represent our Q-function with a neural network, that takes the state (compressed 2 dimensional hidden layer values) and output Q-value for each possible action (1:right, 2:up, 3:left, 4:down). If the agent reaches the goal, reward is set as 1 else reward is 0. Resilient backpropagation proposed in [58] is used in [12] for training of NNs which is one of the fastest weight update mechanisms. It takes into account only the sign of the partial derivative over all patterns (not the magnitude), and acts independently on each weight factor.

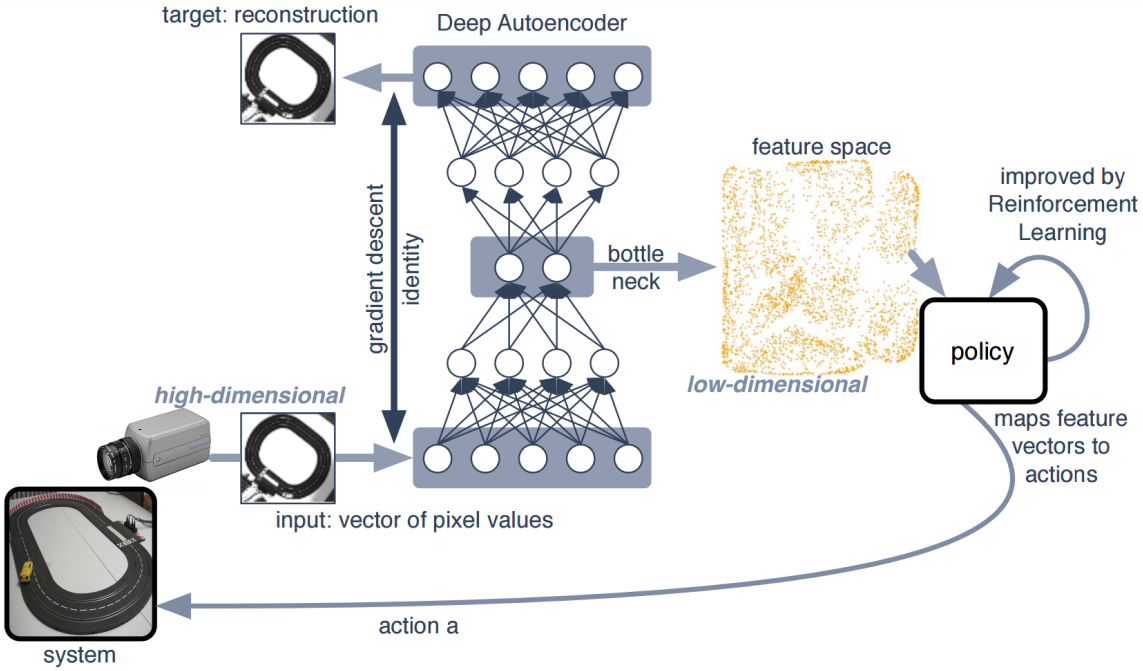


Figure 1.5. Deep Reinforcement Learning applied to visual control of a racing slot car [4].

1.4.3 Human Level Control

Significant progress has been made by integrating advances in deep learning for image pixels preprocessing and deep autoencoder with reinforcement learning, resulting in the deep Q network algorithm [6] which is capable of human level performance on many Atari video games with only raw pixels as input. Traditionally, DQN is implemented using state and action as the input to the network and the maximum discounted future reward by performing action *a* in state *s* (Q-value) is obtained as output. In the paper [6], a new deep Q-network is proposed to learn from represented features from raw images as shown in the right side of Fig. 1.6 for each possible action as mentioned in [5]. The advantage of this approach is that to identify the action with the highest Q-value, we only have to do one forward pass through the network and all Q-values for all actions are available immediately. To do so, convolutional neural network can be used for generating

actions as shown in Fig. 1.7. The network architecture is for 84x84 pixel image and the memory sizes are provided in Fig. 1.8. It shows a classical convolutional neural network with three convolutional layers, followed by two fully connected layers.

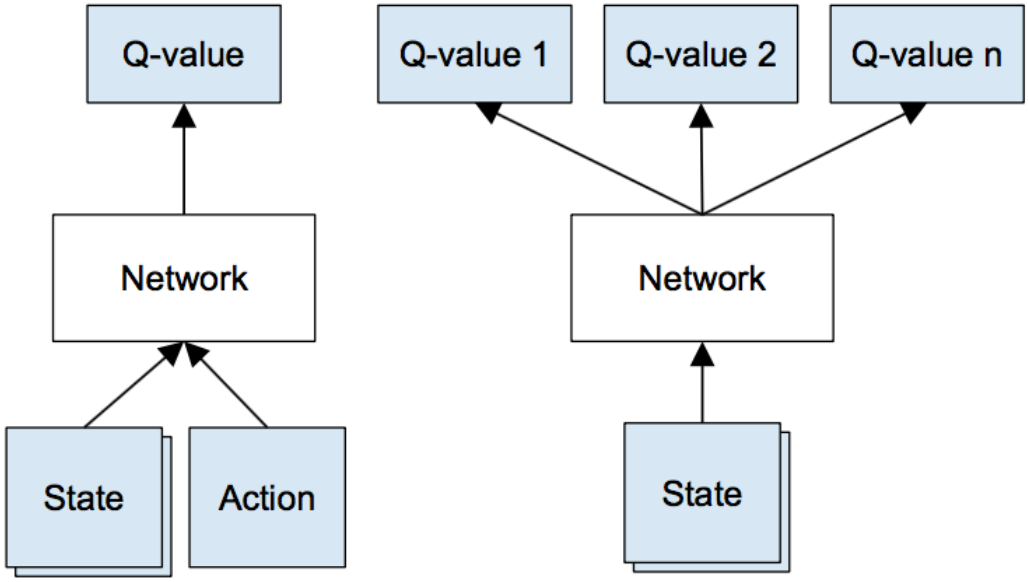


Figure 1.6. Schematic illustration of the DQN structure. Left: Naive formulation of DQN. Right: DQN structure used in Deepmind paper [5].

In [51], the first massively distributed architecture for deep reinforcement learning is presented which is applied to 49 out of Atari 2600 games in the Arcade Learning Environment, using identical hyper-parameters and its performance surpassed non-distributed DQN in above 80% of games. In [59], a conceptually simple and lightweight framework for deep reinforcement learning is proposed that uses asynchronous gradient descent for optimization of deep neural network controllers. It demonstrated that learning a predictive model of state dynamics can result in a pretrained hidden layer structure that reduces the time strongly required to solve reinforcement learning problems. The neural networks are trained in these literatures by rmsprop weight updating mechanism [60], which is a variant of rprop algorithm [58] used for mini-batch

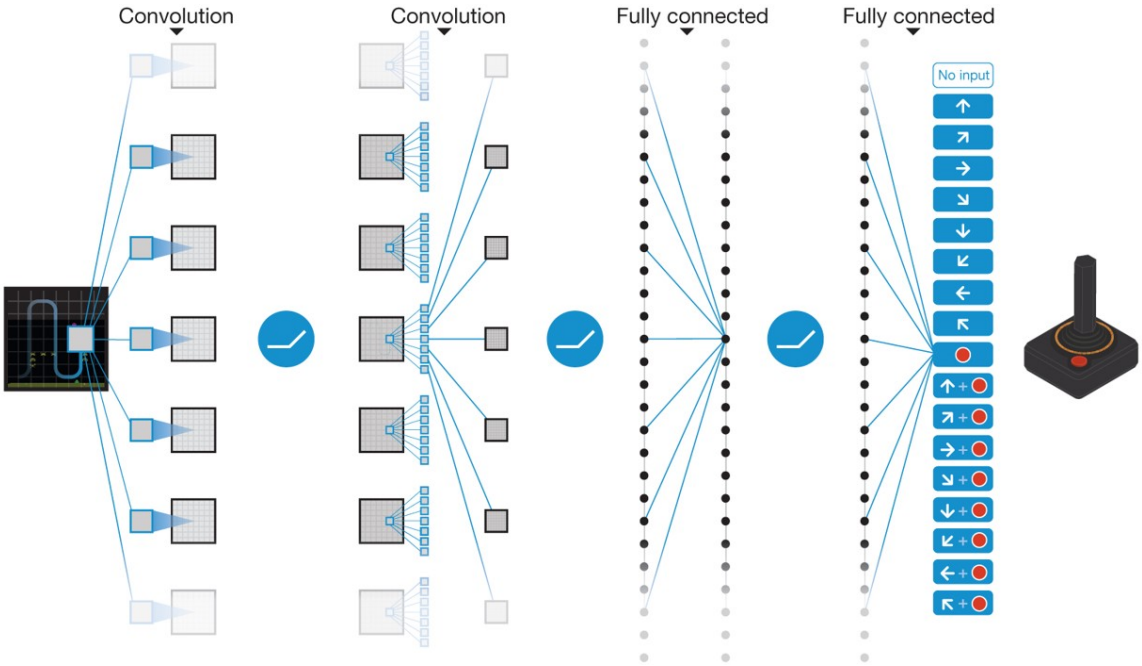


Figure 1.7. Schematic illustration of the convolutional neural network [6].

training.

1.4.4 Future Directions

There are still some open questions and challenges along this direction. For example, after the integration of experience replay technique, how to make sure that the computation and calculation will finish within a short sampling time from the algorithm perspective itself. Especially when the size of history stack increases (e.g., [3]), the computation for each time step will certainly increase dramatically. High-performance computational hardware will be strongly required to make sure the controller still finishes the calculation on time. Also, how to systematically select a batch of samples from history stack rather than doing random sampling. The questions and challenges also reveal many opportunities for the fundamental research in this area.

Layer	Input	Filter size	Stride	Num filters	Activation	Output
conv1	84x84x4	8x8	4	32	ReLU	20x20x32
conv2	20x20x32	4x4	2	64	ReLU	9x9x64
conv3	9x9x64	3x3	1	64	ReLU	7x7x64
fc4	7x7x64			512	ReLU	512
fc5	512			18	Linear	18

Figure 1.8. Convolutional neural network with three convolutional layer [5].

1.5 Contributions and Organization of This Thesis

1.5.1 Contributions

The main contributions of this thesis are stated below:

(a) Successfully integrated the history experience for training both the critic and action networks of the ADP controller. The integrated architecture focusses on improving the training speed and adapting to the system faster. The key parameters in the proposed ADP controller are successfully validated online for the different control benchmarks.

(b) Successfully integrated the history experience based on prioritized sampling for tuning weights of both critic and action network of ADP controller. A systematic approach is proposed to integrate history experience in both critic and action networks of ADP controller design.

(c) Experience replay is designed with the backward TD technique to ensure the model-free learning performance of the ADP controller. The model-free ADP design does not rely on an accurate mathematical model of the system. The traditional forward TD technique (e.g., state-action pair) is between the current time

step and the future time step and will need a model network for future information prediction.

(d) Detailed theoretical analysis is presented using Lyapunov theory in order to verify the stability of the proposed control technique.

(e) Application of ADP controller for transient and steady state performance improvement in virtual synchronous machine (VSM). Generally used proportional-integral (PI) controller in VSM has limited transient performance in current control. ADP controller has been proposed as a supplementary controller for fine tuning conventional PI current controller thereby improving the performance of VSM.

(f) Application of ADP controller for reduction of harmonics in current controlled voltage source inverters. A multiple-input multiple-output (MIMO) online learning control system is developed based on the ADP design. A series of time-delayed current error signals are designed as input for the ADP controller, which outputs compensating control actions (one is for d-axis and the other one is for q-axis) for the current controller in the power inverter. In addition, an appropriate reinforcement signal has been designed with the delayed tracking errors in the power inverter. Based on this, the ADP controller will generate compensating control actions, which guarantees its success.

1.5.2 Thesis Outline

This thesis has been organized as follows: Chapter 2 highlights the first project with the background of history experience with some of its key features, describes the design of

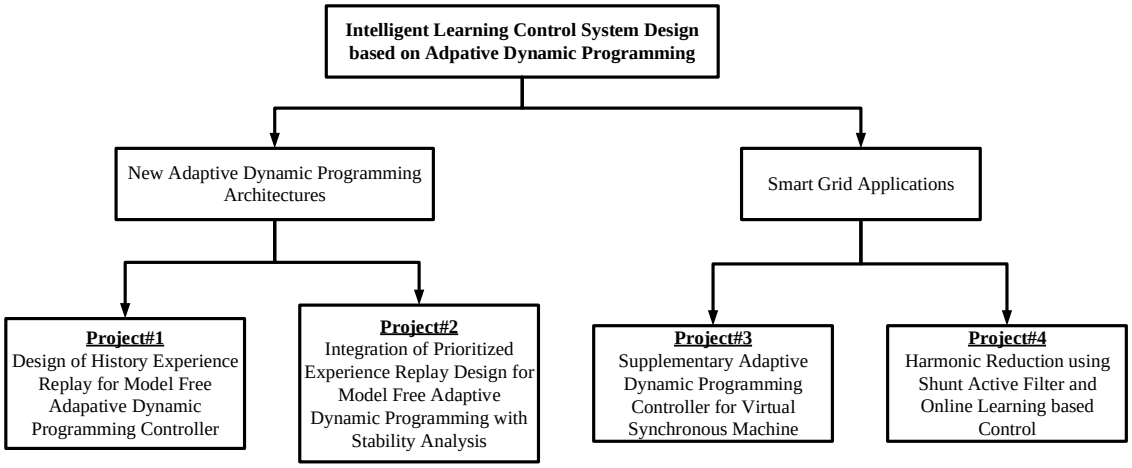


Figure 1.9. Overall organization of the thesis.

the proposed ADP controller and its implementation flowchart, gives the simulation results from two case studies: cart-pole balancing problem and triple-link inverted pendulum, and concludes the chapter along with some future directions along this work. Chapter 3 introduces the second project with prioritized experience replay and its importance for improvement of convergence of online learning control. The systematic integration technique, stability analysis and experimental results are presented as well. Chapter 4 presents the validation of ADP controller for improvement of performance of traditional PI based current controller in VSM. Chapter 5 presents the validation of ADP controller for harmonic reduction based on current controlled voltage source inverter. The benchmark description is presented along with detailed design procedure and experimental results. Finally, Chapter 6 presents the conclusion, limitations, and future developments related to the research thesis.

CHAPTER 2 Design of History Experience Replay for Model Free Adaptive Dynamic Programming Controller

This chapter deals with the new algorithm development and verification with the integration of experience replay. As mentioned in the previous chapter, reinforcement learning (RL) is the learning process of how to map situations to actions taken by an agent so as to maximize the reward [1]. An RL task that satisfies the Markov property is called a Markov decision process. Deep reinforcement learning is a recent advancement in RL tasks which achieved human level performance in Atari video games just by observing the screen pixels and receiving rewards from the game score [8]. The research on deep reinforcement learning has been growing ever since [6, 51, 54, 61]. One of the important elements that accelerated the convergence and speed up the training of deep reinforcement learning is history experience, also known as experience replay in these papers. History experience delivers memory capacity for a learning agent to recall past experiences and apply them to update the current policy. It stores the sample of experiences and repeatedly presents them to the deep reinforcement learning algorithm. This increases the data efficiency of the system because of the opportunity to reuse past experiences.

An adaptive dynamic programming (ADP) controller is one type of actor-critic structure which can solve optimal control problems in complex decision making systems. It consists of action and critic neural networks (NN) with a multi-layer perceptron (MLP) structure in general. The critic network is designed to evaluate the current online learning based controller while the action network produces an online learning based control signal to improve the control performance [17, 18, 23, 24, 62, 63]. Q-learning and ADP are with

good convergent properties and have been recently used as well to tackle the single-agent RL task without full information of the system dynamics [64, 65]. ADP controller has shown promising results on a number of power system control examples [66–68]. However, the performance of the ADP controller is usually obtained by long training periods. In other words, traditional ADP based control has a relatively slower convergence rate because it needs enough training to learn to interact with systems. The samples are used once and then discarded. The controller could forget the past experience after training for a long time. Thus, the training is not data efficient and accuracy is low. Following the trend of integrating history experience with reinforcement learning, several papers [38, 57] have shown interesting results by integrating a history experience table into the ADP/RL methods for improving its data efficiency from both theoretical and experimental perspectives. The authors in [37] proposed one way to integrate history experience with a gradient-descent algorithm for tuning the weights of critic NN by using the current data and some past data (which are stored but can be removed and replaced) for learning the weights at each specific time. The improved convergence speed was achieved, however, the experience replay is used to update only the critic weights and the performance improvement by updating both action and critic network weights is not mentioned. In [57], history experience in actor-critic algorithm is proposed to address the issues of efficiency and autonomy that are required to make reinforcement learning feasible for real-world control tasks to solve difficult learning control problems in a reasonably short time. Authors in [38] developed a general history experience framework that can be combined with essentially any incremental RL technique and present promising real time learning results in inverted pendulum and robot systems which are

encouraging for real time applicability. Data in the history experience database is repeatedly used in [40] to update the weights of the action network and critic network in the ADP approach for optimal control of a DC motor problem [56] and an inverted pendulum. The training speed is improved compared to the traditional ADP approach. The integration of history experience has been reported on actor-critic designs for nonzero-sum games [41]. However, these literatures which integrated history experience in ADP or actor-critic structure or deep reinforcement learning used a one time-step forward sample to train their network which requires a model network to predict the future state. Thus, to systematically facilitate the learning and association process of ADP based approaches by integrating history experience is missing. In addition, maintaining the model-free online learning capability of the integrated approach is also missing for various real-world applications.

There are several major approaches to analyze the stability of the ADP controller. In the learning process, two networks are tuned simultaneously and the stability of the closed-loop system is guaranteed using Lyapunov energy-based techniques [2, 69]. Detailed Lyapunov stability analysis of the ADP based controller is presented in [70] to support the ADP structure from a theoretical point of view. The authors demonstrated that the auxiliary error and the error in the weights estimates are uniformly ultimately bounded (UUB) using the Lyapunov stability construct. In [71], proportional-integral-derivative (PID) control rule is incorporated into neural networks and new results of UUB are provided using a Lyapunov stability construct. The monotonic convergence of optimality is discussed for the goal representation ADP control design in [72], and theoretical proof of convergence is given in terms of both the internal reinforcement signal and the

performance index. In [73, 74], stability of the ADP controller is presented, where the authors demonstrated the theoretical analysis that the estimation errors of NN weights are UUB by the Lyapunov stability construct. Experimentally, later in simulation section, it can be observed that the weights and parameters of the ADP controller are quickly converged and bounded after the system transients.

Modares et. al. [37] proposed the experience-replay based online algorithm for concurrent learning along with current data for adaptation of critic weights. The successful real-time learning results presented in [38] are encouraging for further research in experience replay for the real-time reinforcement learning control. A Lyapunov stability analysis of the ADP based controller is presented in [37] to support the actor-critic structure with integrated experience replay. The authors used the Lyapunov stability construct to guarantee the success of the overall system. An integral reinforcement learning and experience replay algorithm is discussed in [75] to show the convergence of the learning process. An analysis of the experience replay is presented in [76] with the theoretical analysis and the effects of replayed and backward TD. It is shown in [77] that the estimation bias is bounded and asymptotically vanishes, which allows the actor-critic and experience replay based algorithm to preserve the convergence properties of the original algorithm. All these existing publications provide guidance to derive the theoretical stability analysis for the proposed ADP design.

All of aforementioned literature motivates the research to integrate the powerful learning capability of history experience to ADP. However, the existing design of history experience cannot be directly integrated into a model-free ADP design. It is because the existing work uses the forward temporal difference (TD) information (e.g., state-action

pair) between the current time step and future time step to train the network. To predict the future state-action pair, the model network or identifier of the system/environment is essential. In such case, offline data is needed to build the model network and train it very well. The advantage of the backward TD information used in the history experience replay design in this chapter is to avoid the usage of the model network or identifier of the system/environment. Thus, this chapter proposes a systematic history experience replay design to avoid the model network usage. Specifically, the history experience is designed with the tuple or sample from the current time step, t, and the previous time step, $t-1$. Thus, the TD can be achieved by the previous time step and the current time step and experience tuple is designed with a one step backward state-action pair. This preserves the model-free online learning capability of the ADP method. The major contributions of this chapter are twofold:

1. The project in this chapter has integrated the history experience for training both the critic and action networks of the ADP controller. The integrated architecture focusses on improving the training speed and adapting to the system faster. The key parameters in the proposed ADP controller are successfully validated online for the different control benchmarks.

2. The history experience is designed in this chapter with the backward TD technique to ensure the model-free learning performance of the ADP controller. The model-free ADP design does not rely on an accurate mathematical model of the system. The traditional forward TD technique (e.g., state-action pair) is between the current time step and the future time step and will need a model network for future

information prediction.

In addition, the performance improvement is verified with the help of two control problems: a cart-pole balancing task and a triple-link pendulum balancing task. The integration of history experience improves the data efficiency and preserves the online learning capability of the ADP. For fair comparison, we set the same initial starting states and initial weight parameters for both approaches (traditional ADP and history experience integrated ADP) under the same simulation environment. The triple-link pendulum is considered to be one of the most difficult and challenging balancing tasks. Our proposed controller has balanced the system in a short time when the traditional ADP is still at its learning phase.

2.1 Background of History Experience

This section presents the background of history experience starting with the batch training method. The benefits of history experience are then discussed and the time complexity of using this method are focussed at the end.

2.1.1 Batch Learning

Batch reinforcement learning is defined as the task of learning the best possible policy from a fixed set of a transition samples, known as a batch, which can be easily adapted to the classical online case where the agent interacts with the environment while learning. It is a subfield of dynamic programming-based reinforcement learning [78]. Batch learning is one of the methods to learn from the history experience. The idea behind history experience for batch learning is to speed up convergence by using observed state transitions once and reusing them repeatedly as if they were new observations. The

transitions collected using history experience reflect the connection between the states. By spreading it along these connections, there is more efficient use of this information. One of the fastest methods to efficiently learn from these batches of samples is a resilient back-propagation algorithm [12, 58]. However, the algorithm complexity makes it difficult for online learning implementation [79] and thus the gradient descent algorithm is commonly used in the literature. One major difference between batch and online learning is that the batch algorithm computes the error associated with each input sample while keeping the system weights constant. However, online learning updates system weights for each sample in the input. At the end, both algorithms usually converge to the same global minima.

2.1.2 Benefits of History Experience

In addition to increase in data efficiency of the system, another beneficial effect of history experience is the aggregation of information from multiple trajectories as explained in [38]. For an example of a cart-pole, the trajectory to successfully balance a pole may be different. If more than one trajectory are used to train the actor-critic structure during the online process instead of only current data, the convergence will be faster and will be verified later. The other benefit of history experience is that it asymptotically has similar effects with eligibility traces. This is because it transmits similar information as propagated by the eligibility traces [38]. To collect the past information, eligibility traces can be used to provide a short-term memory of many previous input signals. The addition of eligibility traces can accelerate the TD learning process. λ is an eligibility trace decay parameter which normally lies in the range [0,1].

Here, 0 represents no eligibility traces. The benefit of an eligibility trace without having to tune the additional parameter, λ, is obtained with increased size of history experience.

History experience breaks the temporal correlations of the neural network learning updates. It also makes sure that the experiences used to train the networks are not only based on the most recent policy and *epsilon* can be used to scale the amount of exploration. This makes sure the algorithm won't get stuck in a local minimum or diverge. The neural networks are global function approximators and online learning might cause the network to forget when learning a new task, even if it has previously learned to do the related task well. Thus, it is important for the neural networks to properly generalize their knowledge to the whole state-action space by varying the training data enough in the experience database [53].

2.1.3 Discussion of Time Complexity

The most computationally demanding component of the history experience integrated algorithm is the weight updating part, which consumes certain time to reduce the error below the threshold. Parallel computing techniques in a recent simulation environment solves high computational and data-intensive problems using multicore processors, GPUs, and computer clusters. The weights updating part for the action and critic networks by history experience can be implemented using a parallel computing toolbox which divides the computation among available workers. This would reduce the computation time and improve the performance of the system. With the recent advancement in modern technology, the computation burden may not be a barrier for new processors. Thus, a history experience integrated algorithm can have a less computational

burden and have the potential to provide real time optimal control action.

2.2 Proposed ADP Controller and Implementation

This section presents our method of improving the training mechanism of the ADP using history experience along with design and implementation details. The stability of the ADP controller is discussed as well. The proposed algorithm stores the sample of experiences and repeatedly presents them to a gradient-descent based online ADP training algorithm. This increases the data efficiency of the system due to reuse of the otherwise discarded samples.

2.2.1 Overall Framework

The overall diagram of the proposed ADP controller design based on history experience is shown in Fig. 2.1. The proposed controller is a model-free technique that does not require a model network and can adapt to any system to which it is connected to. In one of the basic ADP designs, known as heuristic dynamic programming (HDP) [24], the action network generates the optimal control action iteratively and a critic network evaluates the performance of the action network by approximating J close to the optimal solution. J is also the output of the critic network. The controller observes the state, $x(t)$ from the system and generates the action, $u(t)$ by interacting with the action network. The basic idea in adaptive critic design is to adapt the weights of the critic network to approximate the optimal cost function, $J^*(x(t))$, satisfying the modified Bellman principle of optimality [42], given by:

$$J^*(x(t)) = \min_{u(t)} \{J^*(x(t+1)) + r(x(t)) - U_c\} \tag{2.1}$$

The optimal online learning based controller can be written as:

$$u^*(x(t)) = arg\min_{u(t)} J^*(x(t)) \tag{2.2}$$

where $x(t)$ is the input state vector, $r(x(t))$ is the immediate cost incurred by $u(t)$ at time t, and U_c is the ultimate desired objective which the cost function is desired to achieve. This equation cannot be analytically solved in general. Adaptive dynamic programming is one of the efficient ways to iteratively solve continuous-time and continuous-state problem optimization.

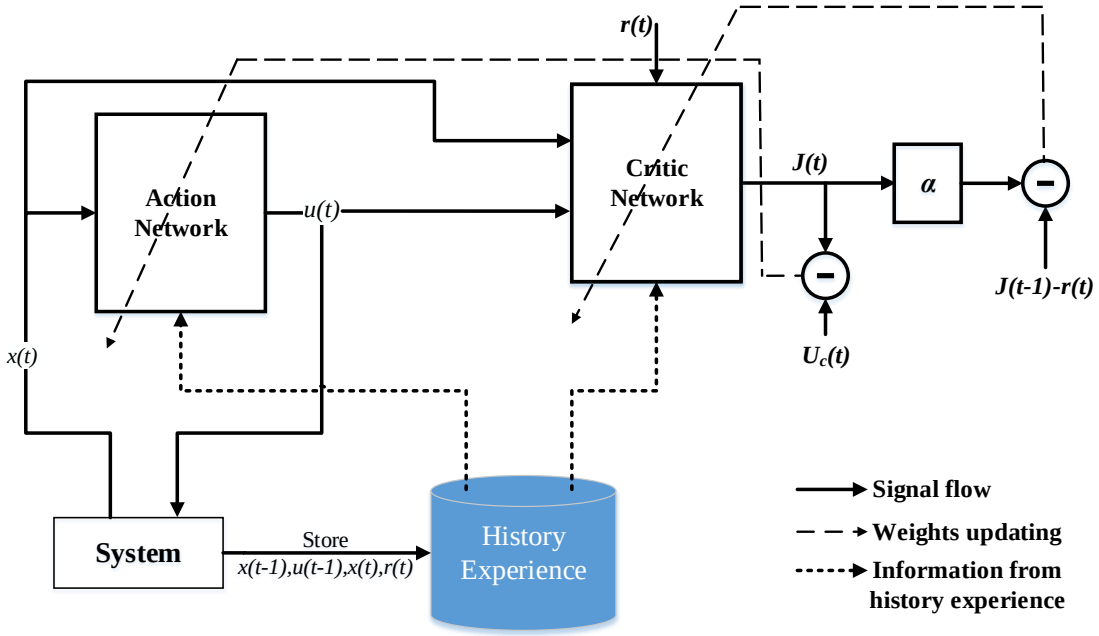


Figure 2.1. The architecture of the proposed ADP controller design with history experience memory.

2.2.2 Design of Model-Free History Experience

The measured history data in form of experience tuple:

$$d(t) = \{x(t-1), u(t-1), r(t), x(t)\} \tag{2.3}$$

is stored into the history experience,

$$D = \{d(1), d(2), d(3), ..., d(L)\}, \tag{2.4}$$

where L is the size of history experience. The entire history experience is limited to the size of length, L and the oldest tuple is discarded if the size of history experience increases over the limit. The system is set one time step back to make the controller learn online. Thus, for the first time step, the controller waits and starts training at the next time step. Feedback to the controller is given through a reinforcement signal, $r(t)$ and the next state is observed through taking action, $u(t-1)$. A random tuple is chosen from the history experience to train both the action and critic network as depicted by "Information from history experience" in Fig. 2.1. Here, the random tuple is chosen to break the correlation between the consecutive samples so as to avoid the local minimum [8]. Thus, a tuple of experience, $d(t)$ given by Eq. (2.3) is stored in the history experience, D given by Eq. (2.4), which will be used to train both the action and the critic network. There are two paths to tune the parameters of action/critic networks in ADP which will be discussed below.

2.2.3 Design of Critic Network

The architecture of the critic network is a MLP structure. The output of the critic network, J function, approximates the discounted total reward to go. At time, t, the measured system state vector, $x(t)$, and control action, $u(t)$ from the action network are inputs to the critic network, and $J(t)$ is the output of the critic element. $R(t)$ is the future accumulative reward-to-go value at time t and $r(t)$=external reinforcement value at t. In the ADP design, the J function is used to approximate R (i.e. $J \rightarrow R$). The weight updating mechanism in ADP for the critic network is based on gradient descent rule which minimizes the prediction error for the network given by: $e_c(t) = \alpha J(t) - [J(t-1) - r(t)]$. The reinforcement signal, $r(t)$ may be as simple as either a "0" or "-1" corresponding to "success" or "failure" respectively and is provided by the external environment and α is the discount factor. A tuple of experience, given by equation (2.3), is randomly chosen from the history experience. Then, the inputs to the critic network are the measured system state vector, $x(t-1)$, and control action, $u(t-1)$. $J(t-1)$ is the output of the critic element at time, $t-1$.

The following weights updating rule is presented to investigate the weights adaptation in the critic network:

$$w_c(t+1) = w_c(t) + \Delta w_c(t) \tag{2.5}$$

$$\Delta w_c(t) = -l_c(t)\left[\frac{\partial E_c(t)}{\partial w_c(t)}\right] - \sum_{n=1}^{N-1} l_c(t)\left[\frac{\partial E_{cn}(t)}{\partial w_{cn}(t)}\right]. \tag{2.6}$$

$$\frac{\partial E_c(t)}{\partial w_c(t)} = \frac{\partial E_c(t)}{\partial J(t)}\frac{\partial J(t)}{\partial w_c(t)} \tag{2.7}$$

The index n is used to refer to the n^{th} sample data $(n = 1, ..., L)$ stored in the history experience and the time, t, is used for the current time. Note in equation (2.6), the first term is a traditional gradient-descent updating law to minimize the objective function, $E_c(t) = 0.5e_c(t)^2$. The last term of equation (2.6) tries to minimize this objective function from the stored samples in the history experience.

2.2.4 Design of Action Network

The action network in the ADP has a similar MLP NN architecture as the critic network. However, the input neuron and output neuron numbers are different. The principle of adjusting the weights of the action network is to indirectly back-propagate the error between the approximate J function from the critic network and the desired ultimate objective, denoted by U_c. In turn, the action network can be implemented by either a linear or a nonlinear network depending on the complexity of the problem. The weight updating in the action network can be formulated as follows: $e_a(t) = J(t) - U_c(t)$. The ultimate objective here is to minimize the squared error by adjusting the action network weights. The input to the action network is the measured system state vector, $x(t)$, from a tuple of experience, $d(t)$, randomly chosen from the history stack as described in the previous subsection. The output of the action network is the control action, $u(t)$. The similar weight updating rule as in the critic network can be applied to action network as follows:

$$w_a(t+1) = w_a(t) + \Delta w_a(t) \tag{2.8}$$

$$\Delta w_a(t) = -l_a(t)\left[\frac{\partial E_a(t)}{\partial w_a(t)}\right] - \sum_{n=1}^{N-1} l_a(t)\left[\frac{\partial E_{an}(t)}{\partial w_{an}(t)}\right]. \tag{2.9}$$

$$\frac{\partial E_a(t)}{\partial w_a(t)} = \frac{\partial E_a(t)}{\partial J(t)}\frac{\partial J(t)}{\partial u(t)}\frac{\partial u(t)}{\partial w_a(t)} \tag{2.10}$$

Note in equation (2.9), the first term is a traditional gradient-descent updating law for the objective function, $E_a(t) = 0.5e_a(t)^2$. The last term of equation (2.9) tries to minimize this objective function from the stored samples in the history experience.

2.3 Online Learning Alogrithms

The algorithm for integrating a history experience into the ADP controller is shown by the flowchart as represented in Fig. 2.2. At the initialization stage, action network weights from input to hidden, w_{a1}, action network weights from hidden to output, w_{a2}, critic network weights from input to hidden, w_{c1}, and critic network weights from hidden to output, w_{c2} are initialized in a certain range typically between [-1,1] with uniform random distribution. The learning rates for the action network, l_a, and for the critic network, l_c, are also initialized (e.g., 0.3 for cart-pole and 0.1 for triple-link case studies). The discount factor, α, is initialized in the range (0,1). The history experience, D, is initialized as null ($\emptyset$). The initial state is set as $x(0)$, which is defined in each of the case study.

As shown in Fig. 2.2, a previous state and action pair is required, and the controller waits for the first time step when t=0 and records this state, $x(0)$ and action, $u(0)$. In the next time step, the new state, $x(t)$ and reinforcement signal, $r(t)$ are obtained. The control action, $u(t)$, is obtained by forward pass of the action network with input as $x(t)$. The experience tuple, $d(t)$, defined in equation (2.3), is built and saved in the history experience, D, as depicted by equation (2.4). If the size of D is greater than L, the oldest entry of D is removed and the space is released for a new tuple. Next, a tuple is sampled by a uniform random sampling technique from the history experience which is used for

training a critic and an action networks as described in Section 2.2.2 and Section 2.2.3. This training and adaptation will be repeated in the next time step until the termination criterion is met.

The weights of critic and action network can be updated using the rule which is derived as follows. The output $J(t)$ of the critic network can be written as:

$$J(t) = \sum_{i=1}^{N_{ch}} w_c^{(2)}(t)\phi_{c,i}(t) \tag{2.11}$$

$$\phi_{c,i}(t) = \frac{1 - exp^{-q_i(t)}}{1 + exp^{-q_i(t)}}, i = 1, ..., N_{ch} \tag{2.12}$$

$$q_i(t) = \sum_{i=1}^{n_a+1} w_{c_{i,j}}^{(1)}(t)x_j(t), i = 1, ..., N_{ch} \tag{2.13}$$

where,

q_i: ith hidden node input of the critic network;

$\phi_{c,i}$: corresponding output of the hidden node;

$\phi_c = [\phi_{c,1}, \phi_{c,2}...\phi_{c,N_{ch}}]^T$;

N_{ch}: total number of hidden nodes in the critic network;

$n_a + 1$: total number of inputs into the critic network including the analog action value $u(t)$ from the action network.

By applying the chain rule, the adaptation of the critic network is summarized as follows:

1) $\Delta w_c^{(2)}$ (hidden to output layer)

$$\Delta w_{c_i}^{(2)}(t) = l_c(t)\left[-\frac{\partial E_c(t)}{\partial w_{c_i}^{(2)}(t)}\right] - \sum_{n=1}^{N-1} l_c(t)\left[\frac{\partial E_{cn}(t)}{\partial w_{c_{in}}^{(2)}(t)}\right] \tag{2.14}$$

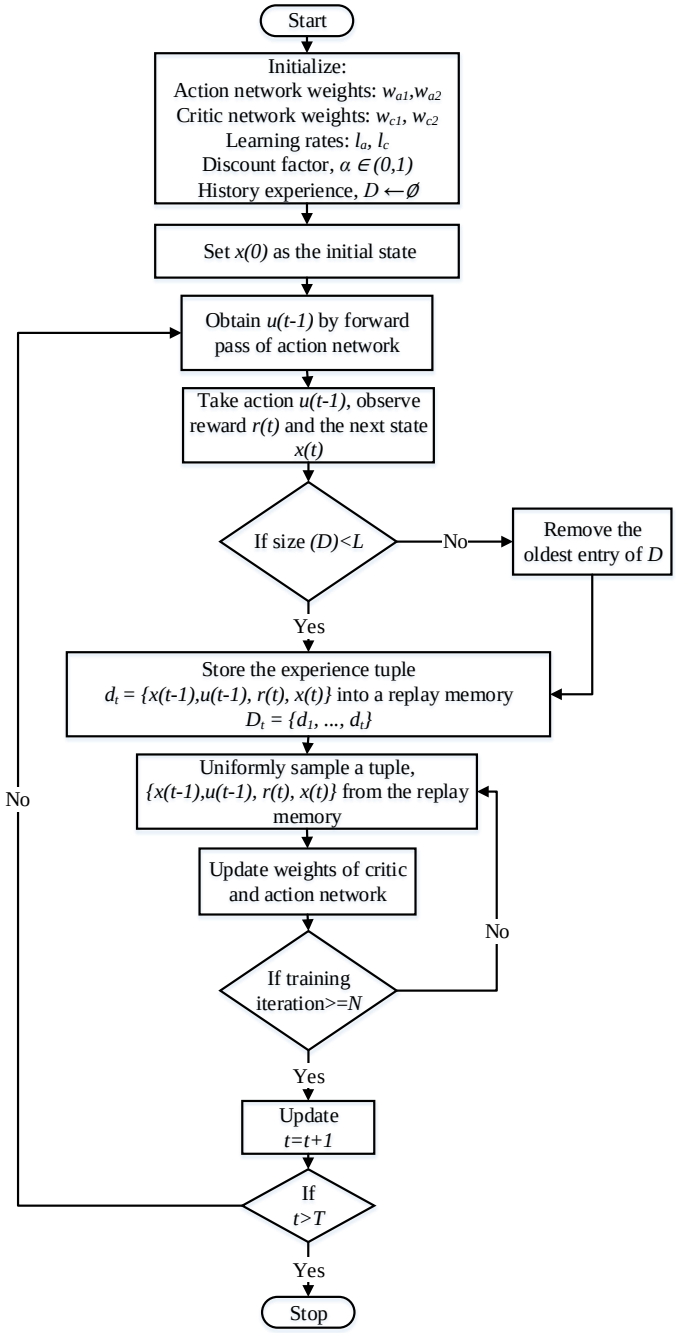


Figure 2.2. The algorithm flowchart of the proposed ADP design.

$$\frac{\partial E_c(t)}{\partial w_{c_i}^{(2)}(t)} = \frac{\partial E_c(t)}{\partial J(t)} \frac{\partial J(t)}{\partial w_c(t)} = \alpha e_c(t) \phi_{c,i}(t) \tag{2.15}$$

2) $\Delta w_c^{(1)}$ (input to hidden layer)

$$\Delta w_{c_{i,j}}^{(1)}(t) = l_c(t) \left[-\frac{\partial E_c(t)}{\partial w_{c_i}^{(1)}(t)} \right] - \sum_{n=1}^{N-1} l_c(t) \left[\frac{\partial E_{cn}(t)}{\partial w_{c_{in}}^{(1)}(t)} \right] \tag{2.16}$$

$$\frac{\partial E_c(t)}{\partial w_{c_{i,j}}^{(1)}(t)} = \frac{\partial E_c(t)}{\partial J(t)} \frac{\partial J(t)}{\partial \phi_{c,i}(t)} \frac{\partial \phi_{c,i}(t)}{\partial q_i(t)} \frac{\partial q_i(t)}{\partial w_{c_{i,j}}^{(1)}(t)} \tag{2.17}$$

$$= \alpha e_c(t) w_{c_i}^{(2)}(t) [\frac{1}{2}(1 - \phi_{c,i}^2(t))] \tag{2.18}$$

Similarly, the equations for the action network are:

$$u(t) = \frac{1 - exp^{-v(t)}}{1 + exp^{-v(t)}} \tag{2.19}$$

$$v(t) = \sum_{i=1}^{N_{ah}} w_{a_i}^{(2)}(t) \phi_{a,i}(t) \tag{2.20}$$

$$\phi_{a,i}(t) = \frac{1 - exp^{-h_i(t)}}{1 + exp^{-h_i(t)}}, i = 1, ..., N_{ah} \tag{2.21}$$

$$\phi_a = [\phi_{a,1}, \phi_{a,2} ... \phi_{a,N_{ah}}]^T \tag{2.22}$$

$$h_i(t) = \sum_{j=1}^{n_a} w_{a_{i,j}}^{(1)}(t) x_j(t), i = 1, ..., N_{ah} \tag{2.23}$$

where, v is the input to the action node, and $\phi_{a,i}$ and h_i are the output and the input of the hidden nodes of the action network, respectively. N_{ah} is the number of hidden nodes of action network. Now, for action network:

1) $\Delta w_a^{(2)}$ (hidden to output layer)

$$\Delta w_{a_i}^{(2)}(t) = l_a(t)\left[-\frac{\partial E_a(t)}{\partial w_{a_i}^{(2)}(t)}\right] - \sum_{n=1}^{N-1} l_a(t)\left[\frac{\partial E_{an}(t)}{\partial w_{a_{in}}^{(2)}(t)}\right] \tag{2.24}$$

$$\frac{\partial E_a(t)}{\partial w_{a_i}^{(2)}(t)} = \frac{\partial E_a(t)}{\partial J(t)}\frac{\partial J(t)}{\partial u(t)}\frac{\partial u(t)}{\partial v(t)}\frac{\partial v(t)}{\partial w_{a_i}^{(2)}(t)} \tag{2.25}$$

$$\begin{aligned} &= e_a(t)[\frac{1}{2}(1-u^2(t))]\phi_{a,i}(t) \\ &\sum_{i=1}^{N_{ch}}[w_{c_i}^{(2)}(t)\frac{1}{2}(1-\phi_{c,i}^2(t))w_{c_{i,n_a+1}}^{(1)}(t)] \end{aligned} \tag{2.26}$$

2) $\Delta w_a^{(1)}$ (input to hidden layer)

$$\Delta w_{a_{i,j}}^{(1)}(t) = l_a(t)\left[-\frac{\partial E_a(t)}{\partial w_{a_{ij}}^{(1)}(t)}\right] - \sum_{n=1}^{N-1} l_a(t)\left[\frac{\partial E_{an}(t)}{\partial w_{a_{ijn}}^{(1)}(t)}\right] \tag{2.27}$$

$$\frac{\partial E_a(t)}{\partial w_{a_{ij}}^{(1)}(t)} = \frac{\partial E_a(t)}{\partial J(t)}\frac{\partial J(t)}{\partial u(t)}\frac{\partial u(t)}{\partial v(t)}\frac{\partial v(t)}{\partial \phi_{a,i}(t)}\frac{\partial \phi_{a,i}(t)}{\partial h_i(t)}\frac{\partial h_i(t)}{\partial w_{a_{ij}}^{(1)}(t)} \tag{2.28}$$

$$\begin{aligned} &= e_a(t)[\frac{1}{2}(1-u^2(t))]w_{a_i}^{(2)}(t)[\frac{1}{2}(1-\phi_{a,i}^2(t))]x_j(t) \\ &\sum_{i=1}^{N_{ah}}[w_{c_i}^{(2)}(t)\frac{1}{2}(1-\phi_{c,i}^2(t))w_{c_{i,n_a+1}}^{(1)}(t)] \end{aligned} \tag{2.29}$$

The weights of the critic and action networks are updated with the algorithm as described in [24]. In both networks, normalization is used in order to confine the values of the weights into some appropriate range as follows:

$$w_c(t+1) = \frac{w_c(t)}{max||w_c(t)||}, if\ ||w_c|| > threshold \tag{2.30}$$

$$w_a(t+1) = \frac{w_a(t)}{max||w_a(t)||}, if\ ||w_a|| > threshold \tag{2.31}$$

2.4 Simulation Results

The simulation results presented here are from two case studies: a cart-pole balancing problem and a triple-link inverted pendulum. For both the cases, a comparison is made between the traditional ADP and the proposed ADP approach.

2.4.1 Cart-pole Balancing Problem

The proposed history experience based ADP architecture has been implemented on a cart-pole balancing problem [24, 80]. The ultimate objective is to generate appropriate control action in terms of force applied on the cart, so as to balance the single pole mounted on the cart. The detail model of the system can be described with the following differential equations:

$$\frac{\partial^2\theta}{\partial t^2} = \frac{g\sin\theta + \cos\theta[-F - ml\theta^2\sin\theta + \mu_c \mathrm{sgn}(\dot{x})] - \frac{\mu_p\dot{\theta}}{ml}}{l\left(\frac{4}{3} - \frac{m\cos^2\theta}{m_c+m}\right)} \tag{2.32}$$

$$\frac{\partial^2 x}{\partial t^2} = \frac{F + ml[\dot{\theta}^2\sin\theta - \ddot{\theta}\cos\theta] - \mu_c\mathrm{sgn}(\dot{x})}{m_c+m}. \tag{2.33}$$

where the acceleration $g = 9.8m/s^2$, the mass of the cart m_c=1.0 kg, the mass of the pole m=0.1 kg, half-pole length l=0.5 m, the coefficient of friction of the cart μ= 0.0005, and the coefficient of friction of the pole μ_p=0.000002. The force, F applied to the cart is in the range [-10, 10] N. A pole is considered fallen when the angle is outside the range of [-12°, 12°] or the cart is beyond the range of [-2.4, 2.4] m. To evaluate the statistical performance of our proposed approach, 100 runs were set to this task with different initial state conditions. Specifically, the angle and angular velocity of the pole in each of these initial states are uniformly generated within [-0.05, 0.05] radians for θ and [-0.5, 0.5] m

Table 2.1. Performance evaluation for cart-pole balancing task for 100 runs. The 2nd, 3rd and the 4th column are with the traditional ADP method, while 5th, 6th and 7th column are with our proposed ADP method.

Noise type	Traditional ADP			Proposed ADP		
	SR*	$\bar{x}^{\dagger}$	$\sigma^{\sharp}$	SR*	$\bar{x}^{\dagger}$	$\sigma^{\sharp}$
Noise free	99%	47.9	58.1	100%	32.7	35.6
Uniform 5% actuator	98%	57.8	72.7	100%	36.4	50.7
Uniform 10% actuator	99%	73.4	130.9	100%	59.2	111.1
5% sensor	98%	90.1	105.3	100%	75.0	88.5
10% sensor	98%	132.7	140.9	100%	95.9	115.6

SR*: 'Success rate' for 100 runs
$\bar{x}^{\dagger}$: Required average number of trials to succeed
$\sigma^{\sharp}$: Standard deviation of trials to succeed

for x, while the angular velocity and velocity of the cart are both 0.

Our proposed approach has been compared with the traditional ADP method presented in [24] with the same parameter setting, except in this case continuous control is used. These results are summarized in Table 2.1. From experience, the memory size is chosen as L=10. The different sizes of this history experience memory has been tried. For instance, if L is chosen as 16, the required average number of trials to succeed increased to 72.3 under noise free condition. This may be because of the overfitting issue in this benchmark. From our experiences in this case study, the optimum experience replay memory size is found by trial-and-error experience. In other systems from previous publications [6, 51, 54], the experience replay memory size might differ. For complex systems with a large number of states (e.g., high dimensional image pixels), the experience replay memory size should be large enough so as to remember the important experiences.

For a noise free condition, the proposed ADP provided 31.7% improvement in average trials to succeed over traditional ADP. In addition, our proposed method provided

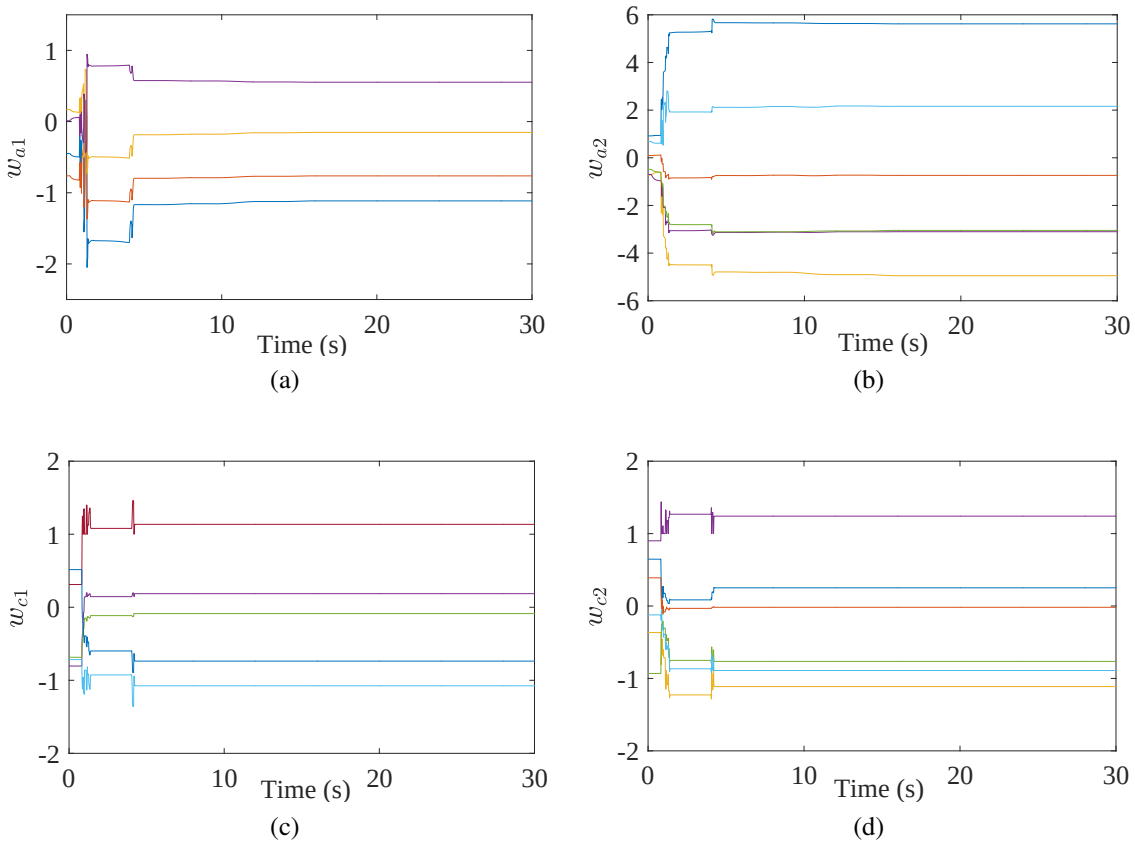


Figure 2.3. The training process of ADP controller: (a) weights trajectories from 4 inputs to 1 hidden node in action network; (b) weights trajectories from 6 hidden to 1 output node in action network; (c) weights trajectories from 5 inputs to 1 hidden node in critic network; and (d) weights trajectories from 6 hidden to 1 output node in critic network.

a less standard deviation of trails to succeed and better results under different noise conditions. This indicates that our approach is more robust and can work effectively under relatively large level of noises. The typical training process of the proposed ADP controller for a single run under noise-free condition is demonstrated in Fig. 2.3 and Fig. 2.4. Fig. 2.3(a) shows the weight evolution of the action network from 4 input nodes to 1 hidden node. Similarly, Fig. 2.3(b) shows the weight evolution of the action network from 6 hidden nodes to 1 output node. Similarly, Fig. 2.3(c) shows the weight evolution of the critic network from 5 input nodes to 1 hidden node. Similarly, Fig. 2.3(d) shows the

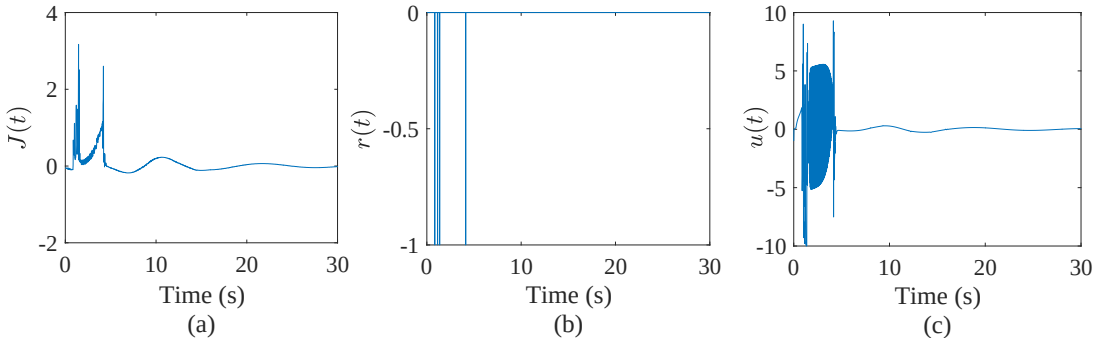


Figure 2.4. The training process of ADP controller: (a) trajectory of output of critic network, $J(t)$; (b) reinforcement signal, $r(t)$; and (c) control action, $u(t)$.

weight evolution of the critic network from 6 hidden nodes to 1 output node. The convergence of all w_{a1}, w_{a2}, w_{c1}, and w_{c2} indicates the convergence of the learning. In addition, Fig. 2.4(a) shows the convergence of the $J(t)$ function, and Fig. 2.4(b) shows the convergence of the reinforcement signal, $r(t)$. Fig. 2.4(c) shows the convergence of the control action, $u(t)$, which is also the output of the ADP controller. The convergence of $u(t)$ indicates that the investigated approach is stable and effective during the actuator noise. Experimentally, it can be observed from Fig. 2.3 and Fig. 2.4 that the weights and parameters of the proposed ADP controller are quickly converged and bounded after the system transients. On average, the proposed ADP provided 26.5% improvement in average trials to succeed over the traditional ADP.

The convergence of the states using the proposed and traditional ADP approaches are compared in Fig. 2.5. For a fair comparison, the same initial states and weights are used as described before. The results from traditional ADP and proposed ADP are at different time scales because the latter finished the simulation earlier. The traditional ADP controller took 287.4s to balance the cart-pole, whereas the proposed ADP controller learned to balance it in 125.7s. Although the angle, θ in case of traditional ADP seemed

to be converged around 50 to 100s, position of cart, x goes outside the limit at that time. This verifies that when the traditional ADP is still in the learning phase, our proposed ADP has learned to balance the cart-pole. The proposed ADP controller starts from

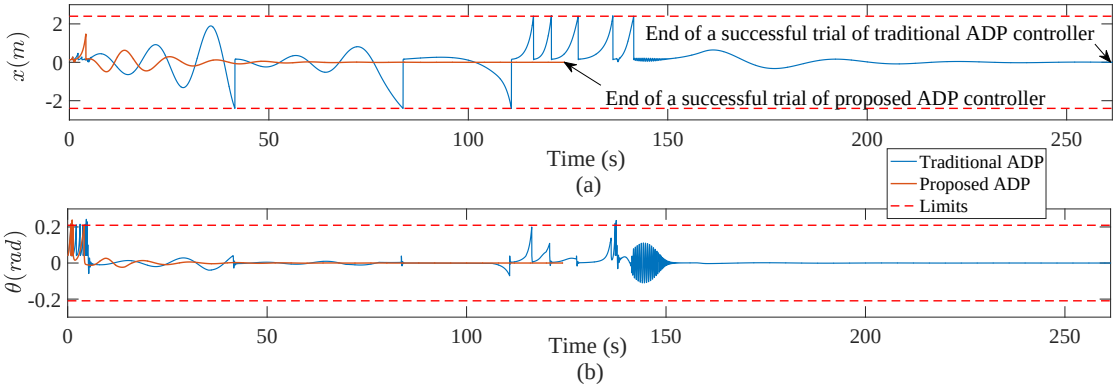


Figure 2.5. Performance comparison between proposed ADP controller and traditional ADP controller (a) x (m), and (b) θ (radians).

random initial weights as shown in Fig. 2.3. The controller starts to learn from the failure trails. The controller carries on the learning experience to the next trial and gradually learns the proper way to achieve the successful trial. From Fig. 2.5, the states: position, x and cart-pole angle, θ are slightly oscillating in order to balance the pole. These are the reasons why there is minor adjusting of action network weights, w_{a1} and w_{a2} at around 10s if they are zoomed-in.

2.4.2 Triple-link Inverted Problem

The same triple-link inverted pendulum balancing task is considered as in [24, 80, 81]. The ultimate objective is to generate appropriate control action in terms of force applied on the cart so as to balance the triple-link mounted on the cart. However, the level of difficulty is much higher compared to a simple cart-pole because there are three joints: mount joint, first joint and second joint. This makes the system highly unstable and

therefore is a challenging problem to solve. This is comparable to deep reinforcement learning trying to solve a difficult Atari video game from scratch in terms of difficulty. Instead of screen pixels, 8 states are used: $[x, \theta_1, \theta_2, \theta_3, \dot{x}, \dot{\theta}_1, \dot{\theta}_2, \dot{\theta}_3]$. The detail model of the system can be described with the following differential equation:

$$F(q)\frac{\partial^2 q}{\partial t^2} = -G\left(q, \frac{\partial q}{\partial t}\right)\frac{\partial q}{\partial t} - H(q) + L(q,u) \tag{2.34}$$

where the components and parameters used are the same as described in [24, 80].

There are eight state variables in this model: position of the cart on the track, x; vertical angle of the first link joint to the cart, θ_1; vertical angle of the second link joint to the first link, θ_2; vertical angle of the third link joint to the second link, θ_3; cart velocity, $\dot{x}$; angular velocity of θ_1, i.e. $\dot{\theta}_1$; angular velocity of θ_2, i.e. $\dot{\theta}_2$; and angular velocity of θ_3, i.e. $\dot{\theta}_3$. The constraints for the triple-linked inverted pendulum are (1) the cart track extends 1.0 m to both sides from the center point; (2) the voltage applied to the motor should be within [-30 V, 30 V]; (3) each link angle should be within the range of $[-20°, 20°]$ with respect to the vertical axis. Here, condition (2) is guaranteed by using a sigmoid function. While for the other two conditions, if either one fails or both fail, the system will be provided with an external reinforcement signal, r=-1 at the moment of failure, otherwise r=0 all the time. Based on this external reinforcement signal, our proposed ADP approach will be trained using the past experiences to facilitate the learning and optimization process over time.

100 runs was conducted for the triple link pendulum. Similar to the cart-pole problem, different initial starting states was used for each run. Specifically, the three

angles and angular velocity of the triple links are set to be uniformly within the range of [-0.02, 0.02] radians and [-0.5, 0.5] radians/s, respectively. As for x and $\dot{x}$, their initial states are set to zero. The critic network architecture is chosen as a 9-–14-–1 MLP (i.e., nine input neurons, fourteen hidden layer neurons, and one output neuron) structure, and the action neural network architecture is chosen as a 8—14-–1 MLP structure. The learning parameters such as learning rate, internal cycle, and internal training error threshold for the action network, and critic network of the proposed ADP are presented in Table 2.2 and the descriptions of the notation used are defined as following:

- $l_c(0)$ Initial learning rate of the critic network at the beginning of the simulation when $t = 0$;
- $l_a(0)$ Initial learning rate of the action network at the beginning of the simulation when $t = 0$;
- $l_c(f)$ Final learning rate of the critic network at the end of the simulation;
- $l_a(f)$ Final learning rate of the action network at the end of the simulation;
- N_c Internal cycle of the critic network;
- N_a Internal cycle of the action network;
- T_c Internal training error threshold for the critic network;
- T_a Internal training error threshold for the action network.

Table 2.2. Parameters used in the triple-link inverted pendulum benchmark

Parameter	$l_c(0)$	$l_a(0)$	$l_c(f)$	$l_a(f)$	N_c	N_a	T_c	T_a	α
Value	0.1	0.1	0.005	0.005	40	50	0.05	0.005	0.95

The summary of the simulation results with different noise conditions is presented in Table 2.3. From our experience and for better performance, the history experience size is set as a short term memory size of L=10. For a noise free condition, the proposed ADP provided 42.73% improvement in average trials to succeed and reduced its standard deviation over the traditional ADP. In addition, our proposed method provided better result in different noise conditions.

Table 2.3. Performance evaluation for triple-link inverted pendulum balancing task for 100 runs. The 2nd, 3rd and the 4th column are with the traditional ADP method, while 5th, 6th and 7th column are with our proposed ADP method.

Noise type	Traditional ADP			Proposed ADP		
	SR*	$\bar{x}^{\dagger}$	$\sigma^{\sharp}$	SR*	$\bar{x}^{\dagger}$	$\sigma^{\sharp}$
Noise free	100%	468.5	488.2	100%	268.3	305.8
Uniform 5% actuator	99%	500.6	511.5	100%	298.6	353.9
Uniform 10% actuator	96%	544.7	465.0	100%	323.6	367.0
Uniform 5% sensor	96%	609.0	608.6	99%	311.4	353.8
Uniform 10% sensor	95%	612.9	624.6	100%	352.9	380.3

SR*: 'Success rate' for 100 runs
$\bar{x}^{\dagger}$: Required average number of trials to succeed
$\sigma^{\sharp}$: Standard deviation of trials to succeed

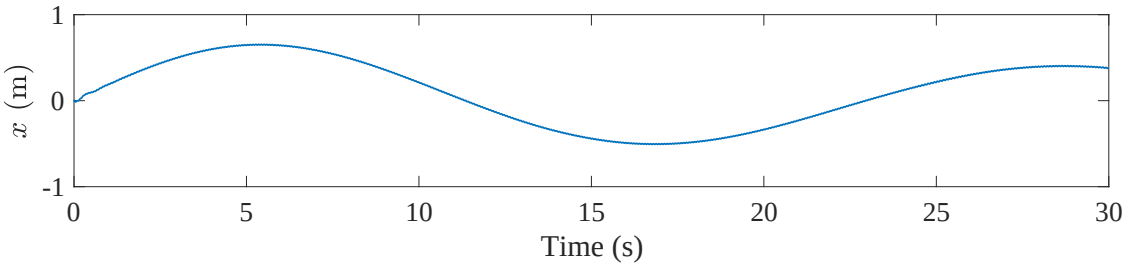


Figure 2.6. Typical trajectory of proposed ADP controller on the triple-link pendulum balancing task: position of the cart, x for a successful run.

Fig. 2.6 shows the convergence for the position of the triple-link inverted pendulum. For fair comparison, initial weights and states of both approaches are kept the same as described before. From the simulation for a typical run with a noise free condition as shown in Fig. 2.7, the proposed ADP controller has learned to balance the triple-link inverted pendulum much faster while the traditional ADP controller needed more time to learn to balance the triple-link pendulum.

2.5 Summary

This project successfully integrated the history experience into the traditional ADP design. The key idea of our approach was to simplify the prior extensive training and

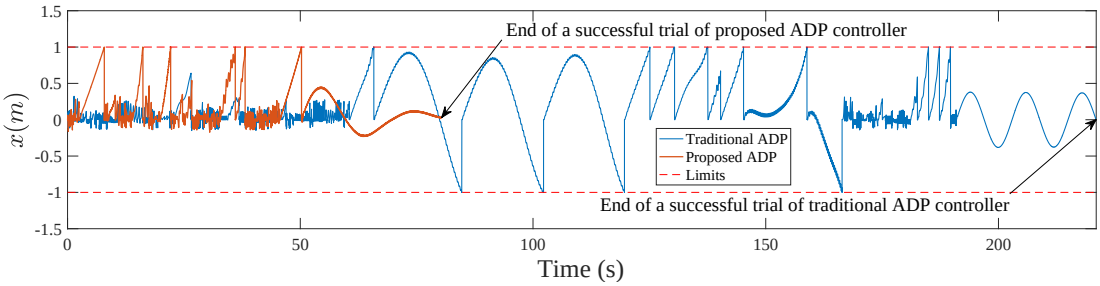


Figure 2.7. Performance comparison between proposed ADP controller and traditional ADP controller with the convergence of state, x (m).

reduce training time associated with the ADP controller thereby preserving the online, model-free learning capability of the ADP. A detailed design architecture and the methodology adapted was presented. Simulation analysis was performed on two case studies: a cart-pole model and a triple-link inverted pendulum model to demonstrate the superior learning capability of the integrated approach. The important remark from this project that creates motivation for next chapter is as follows. The history experience was sampled in our study with random sampling, however it can be sampled with some rule or priority which might further boost the convergence process.

CHAPTER 3 Integration of Prioritized Experience Replay Design for Model Free Adaptive Dynamic Programming with Stability Analysis

This chapter deals with the new algorithm development and verification with the integration of prioritized experience replay (PER). This chapter also provides detailed stability analysis for the proposed algorithm. As mentioned in previous chapter, adaptive dynamic programming (ADP) controller is a powerful technique that solves challenging control problems iteratively [24]. It consists of an action and a critic neural network (NN) with a multi-layer perceptron (MLP) structure. The critic network is designed to evaluate the current online learning based controller while the action network produces a control action [17, 18, 23, 24, 62]. However, convergence of ADP controller is relatively slow because it needs more time to learn from the system. Traditional ADP controller discards incoming data immediately. There are two drawbacks: (a) Traditional ADP control approach neglects possibly rare and important experiences that would be useful later on, and (b) possible correlated updates that break the independent and identically distributed (i.i.d.) assumption of stochastic gradient-based algorithms used for traditional ADP. Deep learning and deep reinforcement learning techniques provide some hints to solve this problem with the concept of storing past experiences [6, 8].

Experience replay (ER) is one of key techniques in the aforementioned papers, and it delivers memory capacity for a learning agent to recall past experiences to update the current policy [52]. It stores experience tuple similar to state-action pair in reinforcement learning and repeatedly presents them to various types of reinforcement learning algorithm (e.g., Deep-Q Network (DQN)). This technique breaks the temporal

correlations between the consecutive samples with randomly presenting tuple information [82]. However, existing uniform random sampling is not an efficient way to use history information. It is studied in [6, 8] that uniform random sampling method does not differentiate important tuples and always overwrites with the recent tuples due to the limitation of finite memory size. It also gives identical weight to all tuples in the experience replay. The authors in [6, 8] suggest that a more sophisticated strategy is needed similar to prioritized sweeping [83] that promotes the important tuples. Prioritized experience technique presents important tuples frequently, and therefore uses experience replay information more efficiently [84]. The experimentation of the Atari Learning Environment in [82] shows that the prioritized sampling with Double DQN significantly outperforms previous state-of-the-art Atari video game results. In [85], prioritized sampling strategy is combined with deep Q-network with experiment on four Atari 2600 games. The authors conclude that using prioritized sampling can lead to a faster and more stable learning process, and a better performance of scoring. In [82], the authors investigate how prioritization can make experience replay more efficient and effective than if all tuples are replayed uniformly. The key idea presented is that an DQN agent can learn more effectively from certain tuples than from others.

ER is integrated with actor-critic algorithm in [57] to address the issues of efficiency and autonomy that are required to make reinforcement learning feasible for real-world control tasks to solve difficult learning control problems in a reasonable short time.
In [38], a general experience replay framework is developed that can be combined with essentially any incremental RL technique and the authors present promising real time learning results in inverted pendulum and robot systems, which are encouraging for real

time applicability. The weights of the action network and critic network in ADP are updated using the experience replay [40] for optimal control of a DC motor problem [56] and an inverted pendulum. The training speed is improved compared to classic ADP approach. In [41], the experience replay has been integrated with actor-critic designs for nonzero-sum games. Because of prioritized experience replay (PER), more important sample has higher probability to be selected for training. Thus, ADP controller is expected to learn better policy efficiently and effectively. There are two variants of prioritization technique: proportional prioritization [86] and rank-based prioritization [87] out of which rank-based prioritization is considered in this paper. Modares et. al. [37] proposed the experience replay based online algorithm for concurrent learning along with current data for adaptation of critic weights in a continuous time domain. In [10], theoretical analysis for double Q-learning is given with verification that the lower bound on the absolute error of the double Q-learning estimate is zero. In [88], theoretical analysis is presented with proof that recent development in reinforcement learning called retrace can be interpreted as an application of the importance weight truncation and bias correction trick advanced in their paper. The stability of the proposed controller is verified in this paper using similar Lyapunov function as that in [70, 73].

Motivated by the new state-of-the-art achievement by prioritized experience replay of efficient learning, a similar technique has been proposed to integrate prioritized experience replay into ADP controller. The prioritized experience replay has been integrated to further improve the training process compared to experience replay based ADP methods [89]. In addition, the current tuple is the most important sample for on-line learning and it is selected for the first training step. The rest of the samples are collected

by prioritized sampling method apart from the current tuple. The samples collected by prioritized sampling method has the most important information and is used for updating weights of both action and critic network.

Modares et. al. [37] proposed the experience replay based online algorithm for concurrent learning along with current data for adaptation of critic weights in a continuous time domain. In [10], theoretical analysis for double Q-learning is given with verification that the lower bound on the absolute error of the double Q-learning estimate is zero. In [88], theoretical analysis is presented with proof that recent development in reinforcement learning called retrace can be interpreted as an application of the importance weight truncation and bias correction trick advanced in their paper. The stability of the proposed controller is verified in this chapter using similar Lyapunov function as that in [70, 73].

3.1 Design of Experience Replay

Experience replay technique delivers an agent with a capacity to memorize and recall past experiences. This can be applied to update the present policy. Thus, high data efficiency is achieved by reutilizing the samples. The basic technique behind experience replay is to store all the experience tuple defined as,

$$e_t = (x(t-1), u(t-1), r(t), x(t)) \tag{3.1}$$

where t refers to the current time instance, $x(t)$ refers to previous state, $u(t-1)$ is the action taken in order to move to the current state, $x(t)$ according to which reward signal, $r(t)$ is given. Thus, experience tuple is similar to state-action pair in the reinforcement

learning. Here, state action pair is backed up one-time step to avoid the model network. The overall experience replay with finite size of L is defined as

$$D = \{e_1, e_2, ..., e_L\} \tag{3.2}$$

Here, L is user-defined according to different applications. When training the neural network, random minibatches or a single tuple from the experience replay are/is sampled instead of the most recent tuple to avoid local minimum or diverging case [6]. Experience replay is first proposed in [52], in which experience data are stored and chosen randomly to update the value function and policy in reinforcement learning for neural network approximation. There are many advantages of experience replay as mentioned in [53], first it helps to increase the sample efficiency by allowing samples to be reused. In the context of neural networks when the training is performed on a GPU, experience replay allows for mini-batch updates which helps the computational efficiency. In addition, learning from mini-batch samples would cause the updates of the network parameters to have a low variance, leading to faster and stable learning. Another advantage of experience replay is that ε can be used to scale the amount of exploration and thus the experiences used to train the networks are not only based on the most recent policy. If ε=1, the movement follows uniform random distribution and if ε=0, the optimal policy is followed, known as greedy behavior.

3.2 Prioritized Sampling in Experience Replay and Integration Into ADP Design

The algorithm for integrating a prioritized experience replay into the ADP controller is shown in Fig. 3.1, which is extended from the recent work [89]. The similarity with the

previous work is that the current tuple is the most important sample for on-line learning and it is selected for the first training step. The main difference from the previous work is that the rest of the samples are collected by prioritized sampling method apart from the current tuple. The samples collected by prioritized sampling method has the most important information and is used for updating weights of both action and critic network as shown in Fig. 3.1.

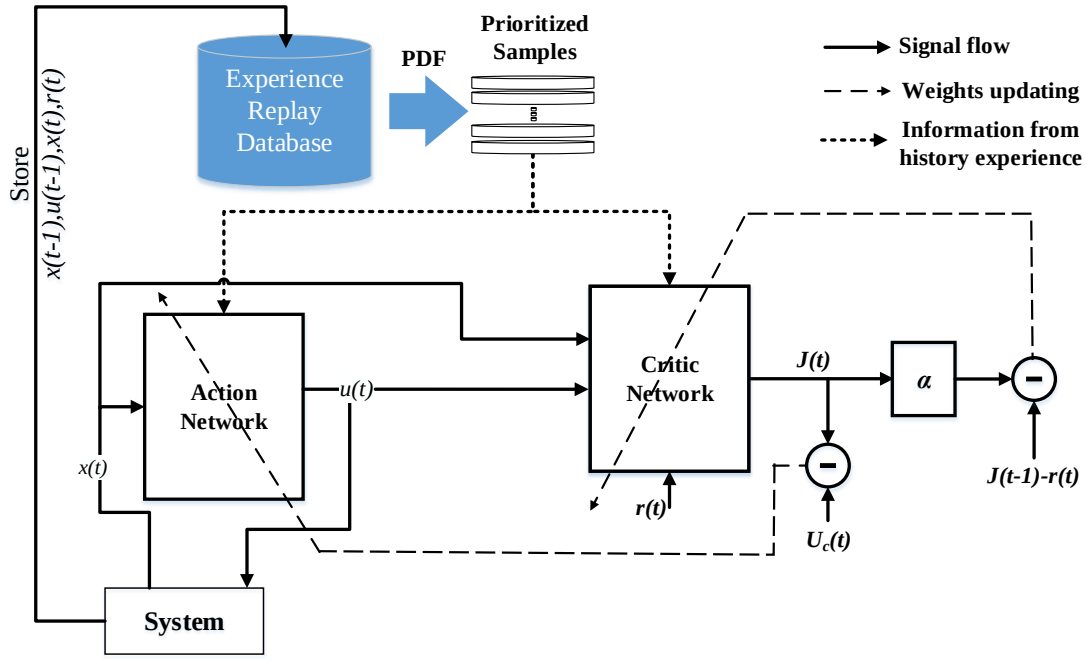


Figure 3.1. Proposed ADP diagram based on prioritized experience replay.

The architecture of the critic network is a MLP structure. The output of the critic network, J function, approximates the discounted total reward to go. At time, t, the measured system state vector, $x(t)$, and control action, $u(t)$ from the action network are inputs to the critic network, and $J(t)$ is the output of the critic element. $R(t)$ is the expected accumulative reward-to-go value at time t and $r(t)$=external reinforcement value at t. In the ADP design, the J function is used to approximate R (i.e. $J \rightarrow R$). The action

network in the ADP has a similar MLP NN architecture as the critic network. However, the input neuron and output neuron numbers are different. In turn, the action network can be implemented by either a linear or a nonlinear network depending on the complexity of the problem. This section describes about prioritized sampling method and how it is integrated into ADP design.

3.2.1 Prioritized Sampling of Experience Replay

Tuples are randomly picked from the experience replay with the equal probability as shown in Fig. 3.2(a) and thus learning may not be from the special ones. Fig. 3.2(b) shows probability density function for prioritized sampling with the most important information at the center.

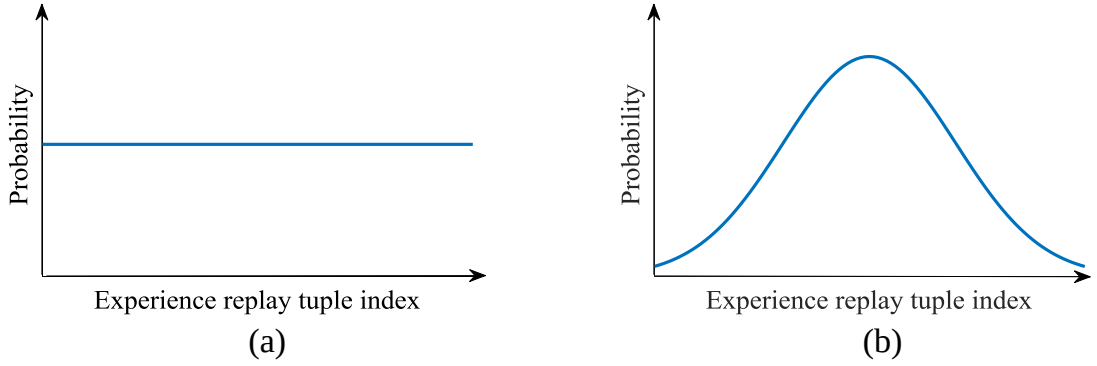


Figure 3.2. Probability density function for (a) Uniform random sampling (b) Prioritized sampling.

A stochastic sampling method is adapted from [82] which finds balance between pure greedy prioritization and uniform random sampling. Priority is translated to probability of being chosen for training. The probability of sampling k^{th} experience replay tuple is given as:

$$P(k) = \frac{p_k^{\alpha}}{\sum_{l=1}^{L-1} p_l^{\alpha}} \tag{3.3}$$

where $p_k^\alpha > 0$ is the priority of k^{th} tuple, and the exponent, α determines how much prioritization is used, with α=0 corresponding to the uniform case. Rank-based prioritization is defined with:

$$p_k = \frac{1}{rank(k)} \tag{3.4}$$

where $rank(k)$ is the rank of k^{th} experience replay tuple when the experience replay database is sorted according to $|Error_k|$. The concept is that a high TD-error is correlated to a high level of information. These tuples should be sampled with more probability than the others. The advantage is that this prioritization technique is not sensitive to outliers, and thus is more robust in addition to large speed-ups in learning curve. Therefore, the tuples are sorted in descending order of $|Error_k|$. The index, k returned after sorting the tuple is used to calculate the rank by using: $rank(k) = y$, where y ranges from 1 to size($Error_k$). Fig. 3.3 shows the prioritized sampling method in which tuple(s) of experience is selected according to probability density function. In an experience replay tuple, the current tuple of experience represented by Eq. (3.1) is added to the newest tuple entry and oldest tuple is destroyed according to the experience replay size limit. The probability density curve is obtained by Eq. (3.3) from which random mini-batch or tuples of experience of given size can be extracted using different sampling/selection method. The sample with the highest probability is the most important one and has the highest chance to be selected for training. This important tuple of experience are used for training the ADP controller.

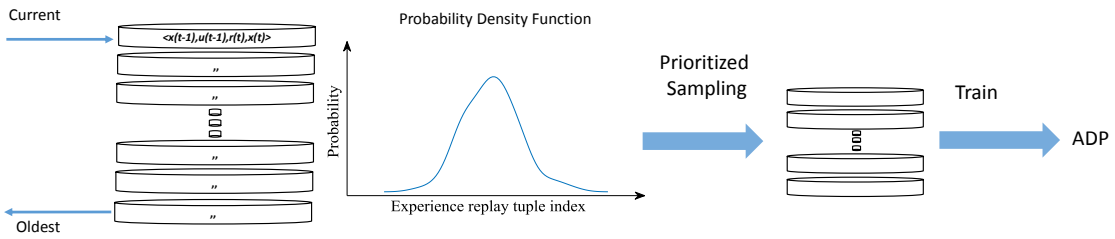


Figure 3.3. Training procedure for prioritized experience replay based ADP with prioritized sampling method.

3.2.2 Integration of Prioritized Experience Replay in Critic Network

The weight updating mechanism in ADP for the critic network is based on gradient descent rule which minimizes the prediction error for the network given by:

$$e_c(t) = \alpha J(t) - [J(t-1) - r(t)] \tag{3.5}$$

and the squared error is given by an objective function, $E_c(t) = 0.5 e_c(t)^2$. The reinforcement signal, $r(t)$ may be as simple as either a "0" or "-1" corresponding to "success" or "failure" respectively and is provided by the external environment and α is the discount factor. A tuple of experience, given by Eq. (3.1), is randomly chosen from the experience replay.

The following weights updating rule is presented to investigate the weights adaptation in the critic network:

$$w_c(t+1) = w_c(t) + \Delta w_c(t) \tag{3.6}$$

$$\Delta w_c(t) = -l_c(t) \underbrace{\left[\frac{\partial E_c(t)}{\partial w_c(t)}\right]}_{\substack{\text{traditional} \\ \text{error propagation}}} - \sum_{n=1}^{N-1} l_c(t) \underbrace{\left[\frac{\partial E_{cn}(t)}{\partial w_c(t)}\right]}_{\substack{\text{proposed additional} \\ \text{error propagation}}} . \tag{3.7}$$

The index n is used to refer to the n^{th} experience replay tuple $(n = 1, ..., N-1)$ sampled from the experience replay and the time, t, is used for the current time. The index k used in previous subsection represents index of experience replay before sampling and n represents index of tuple of experience after prioritized sampling. Here, $N-1$=number of samples from experience replay using prioritized sampling used for training critic and action networks where, $N \leq L$. Note in Eq. (3.7), the first term is a traditional gradient-descent updating law to minimize the objective function, $E_c(t)$. The last term of Eq. (3.7) tries to minimize this objective function from the stored samples in the experience replay. Here, $E_{cn}(t)$ represents the critic network squared error or objective function of n^{th} experience tuple obtained from prioritized sampling.

3.2.3 Integration of Prioritized Experience Replay in Action Network

The principle of adjusting the weights of the action network is to indirectly back-propagate the error between the approximate J function from the critic network and the desired ultimate objective, denoted by U_c. The weight updating in the action network can be formulated as follows:

$$e_a(t) = J(t) - U_c(t). \tag{3.8}$$

The ultimate objective here is to minimize the squared error, $E_a(t) = 0.5 e_a(t)^2$ by adjusting the action network weights. The input to the action network is the measured system state vector, $x(t)$, from a tuple of experience, $d(t)$, randomly chosen from the experience replay as described in the previous subsection. The output of the action network is the control action, $u(t)$. The similar weight updating rule as in the critic

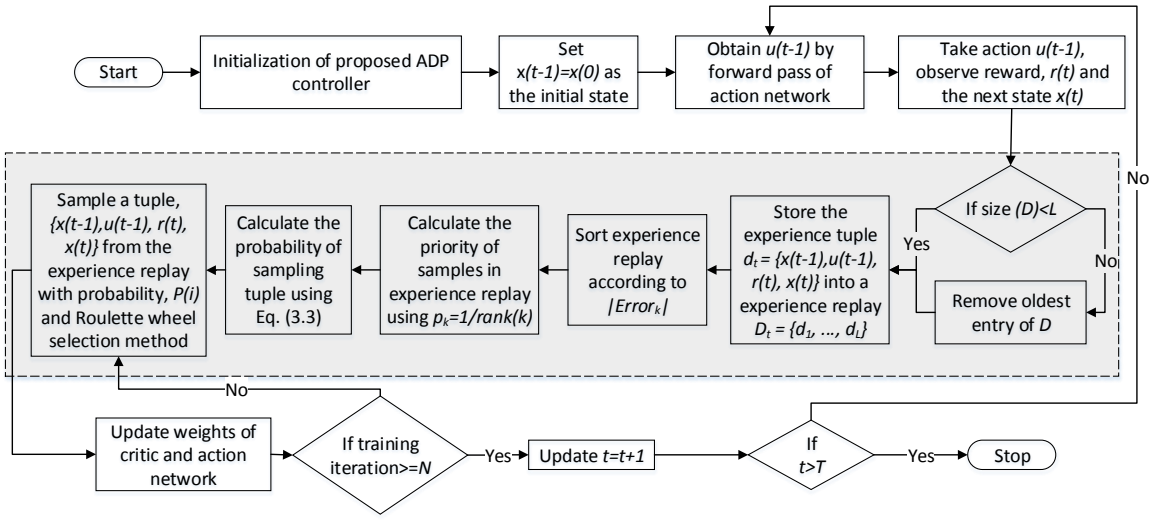


Figure 3.4. Proposed ADP algorithm based on prioritized experience replay. The highlighted portion shows detailed steps for prioritized experience replay integration.

network can be applied to action network as follows:

$$w_a(t+1) = w_a(t) + \Delta w_a(t) \tag{3.9}$$

$$\Delta w_a(t) = -l_a(t) \underbrace{\left[\frac{\partial E_a(t)}{\partial w_a(t)}\right]}_{\substack{\text{traditional} \\ \text{error propagation}}} - \sum_{n=1}^{N-1} l_a(t) \underbrace{\left[\frac{\partial E_{an}(t)}{\partial w_a(t)}\right]}_{\substack{\text{proposed additional} \\ \text{error propagation}}} . \tag{3.10}$$

Note in Eq. (3.10), the first term is a traditional gradient-descent updating law for the objective function, $E_a(t)$. The last term of Eq. (3.10) tries to minimize this objective function from the stored samples in the experience replay. Here, $E_{an}(t)$ represents the action network squared error or objective function of n^{th} experience tuple obtained from prioritized sampling.

3.2.4 On-line Learning Algorithms

The algorithm for integrating a prioritized experience replay into the ADP controller is shown by the flowchart in Fig. 3.4. At the initialization stage, action network and critic network weights are initialized in a certain range with uniform random distribution. The learning rates for the action network, l_a, and for the critic network, l_c, are also initialized typically in the small range. The discount factor, α, is initialized in the range (0,1). The experience replay, D, is initialized as null ($\emptyset$). The initial state is set as $x(0)$, which is defined in each of the case study. A previous state and action pair is required, and the controller waits for the first time step when t=0 and records this state, $x(0)$ and action, $u(0)$. In the next time step, the new state, $x(t)$ and reinforcement signal, $r(t)$ are obtained. The control action, $u(t)$, is obtained by forward pass of the action network with input as $x(t)$. The experience tuple, $d(t)$, defined in Eq. (3.1), is built and saved in the prioritized experience replay, D, as depicted by Eq. (3.2). The prioritized experience replay works on the principle similar to that of queue (first-in-first-out, FIFO principle). The current tuple is stored at the top of the prioritized experience replay as shown in Fig. 3.3.

If the size of D is greater than L, the oldest entry of D is removed and the space is released for a new tuple. As mentioned at the beginning of Section 3.2, current tuple contains the highest level of information and it is always selected for the first training period. For the next training period during each time step, the experience replay is sorted according to error $|Error_k|$ and priority of sample is calculated for the samples in the experience replay using Eq. (3.4). Next, a given number of tuples are sampled by a probability sampling technique following Eq. (3.3) with roulette wheel selection

method [90] from the experience replay which is used for training a critic and an action networks. The action and critic networks weights can be trained by algorithm as shown in chapter 2, section 2.3, Eq. (2.14)-(2.29). This training and adaptation will be repeated in the next time step until the termination criteria is met.

3.2.5 Stability Analysis

For the stability of the proposed ADP controller, a similar Lyapunov function as that in [70, 73] has been reconstructed. The Lyapunov function is as follows:

$$V = V_1 + V_2 + V_3 \tag{3.11}$$

where,

$$V_1 = \frac{1}{l_c} tr(\tilde{w}_c^T(t)\tilde{w}_c(t)) \tag{3.12}$$

$$V_2 = \frac{1}{2}||\zeta_c(t-1)||^2 \tag{3.13}$$

$$V_3 = \frac{1}{\gamma l_a} tr\Big(\tilde{w}_a^T(t)\tilde{w}_a(t)\Big) \tag{3.14}$$

In Eq. (3.13), $\zeta_c(t-1) = (\hat{w}_c(t-1) - w_c)^T \phi_c(t-1) = \tilde{w}_c^T(t-1)\phi_c(t-1)$ and $\gamma > 0$. The first difference of the Lyapunov function candidate can be written as:

$$\Delta V = \Delta V_1 + \Delta V_2 + \Delta V_3. \tag{3.15}$$

For ΔV_1,

$$\Delta V_1 = \frac{1}{l_c} tr\Big(\tilde{w}_c^T(t+1)\tilde{w}_c(t+1) - \tilde{w}_c^T(t)\tilde{w}_c(t)\Big) \tag{3.16}$$

where,

$$\begin{aligned}\hat{w}_c(t+1) =& \hat{w}_c(t) - l_c\alpha\phi_c(t)\Big(\alpha\hat{w}_c^T(t)\phi_c(t) + r(t) - \hat{w}_c^T(t-1)\phi_c(t-1)\Big)^T \\ & - l_c\alpha\sum_{n=1}^{N-1}\phi_{cn}(t)\Big(\alpha\hat{w}_c^T(t)\phi_{cn}(t) + r_n(t) - \hat{w}_c^T(t-1)\phi_{cn}(t-1)\Big)^T\end{aligned} \tag{3.17}$$

and the corresponding $\tilde{w}_c(t+1)$ can be expressed as:

$$\begin{aligned}\tilde{w}_c(t+1) =& \tilde{w}_c(t) - l_c\alpha\phi_c(t)\Big(\alpha\hat{w}_c^T(t)\phi_c(t) + r(t) - \hat{w}_c^T(t-1)\phi_c(t-1)\Big)^T \\ & - l_c\alpha\sum_{n=1}^{N-1}\phi_{cn}(t)\Big(\alpha\hat{w}_c^T(t)\phi_{cn}(t) + r_n(t) - \hat{w}_c^T(t-1)\phi_{cn}(t-1)\Big)^T \qquad (3.18)\\ =& \Big(I - l_c\alpha^2\phi_c(t)\phi_c^T(t) - \sum_{n=1}^{N-1} l_c\alpha^2\phi_{cn}(t)\phi_{cn}^T(t)\Big)\tilde{w}_c(t) \\ & - l_c\alpha\phi_c(t)\Big(\alpha\hat{w}_c^T(t)\phi_c(t) + r(t) - \hat{w}_c^T(t-1)\phi_c(t-1)\Big)^T \qquad (3.19)\\ & - l_c\alpha\sum_{n=1}^{N-1}\phi_{cn}(t)\Big(\alpha\hat{w}_c^T(t)\phi_{cn}(t) + r_n(t) - \hat{w}_c^T(t-1)\phi_{cn}(t-1)\Big)^T.\end{aligned}$$

Then,

$$\begin{aligned}\Delta V_1 = \frac{1}{l_c} tr\Big(& \tilde{w}_c^T(t)A^TA\tilde{w}_c(t) - \tilde{w}_c^T(t)\tilde{w}_c(t) + B\alpha^2 l_c^2\phi_c^T(t)\phi_c(t)B^T \\ & -\tilde{w}_c^T(t)A^T l_c\alpha\phi_c(t)B^T - Bl_c\alpha\phi_c^T(t)A\tilde{w}_c(t)\Big)\end{aligned} \tag{3.20}$$

where $A = I - l_c\alpha^2\phi_c(t)\phi_c^T(t) - \sum_{n=1}^{N-1} l_c\alpha^2\phi_{cn}(t)\phi_{cn}^T(t)$ and $B = \alpha w_c^T\phi_c(t) + r(t) - \tilde{w}_c^T(t-1)\phi_c(t-1) - l_c\alpha\sum_{n=1}^{N-1}\phi_{cn}(t)\Big(\alpha\hat{w}_c^T(t)\phi_{cn}(t) + r_n(t) - \hat{w}_c^T(t-1)\phi_{cn}(t-1)\Big)^T$. From [70],

$$\begin{aligned}\tilde{w}_c^T(t)A^TA\tilde{w}_c(t) - \tilde{w}_c^T(t)\tilde{w}_c(t) =& -l_c\alpha^2||\zeta_c(t)||^2 - l_c\alpha^2\tilde{w}_c^T(t)\phi_c\phi_c^T \\ & \times\Big(I - l_c\alpha^2\phi_c\phi_c^T - \sum_{n=1}^{N-1} l_c\alpha^2\phi_{cn}(t)\phi_{cn}^T(t)\Big)\tilde{w}_c(t)\end{aligned} \tag{3.21}$$

where $\zeta_c(t) = \tilde{w}_c^T(t)\phi_c(t)$.

Thus,

$$\begin{aligned}\Delta V_1(t) = &-\alpha^2||\zeta_c(t)||^2-\alpha^2\Big(1-l_c\alpha^2||\phi_c(t)||^2-\sum_{n=1}^{N-1} l_c\alpha^2||\phi_{cn}(t)||^2\Big)||\zeta_c(k)||^2\\ &+l_c\alpha^2||\phi_c(t)||^2\times||\alpha w_c^T\phi_c(t)+r(t)-\hat{w}_c^T(t-1)\phi_c(t-1)\\ &-l_c\alpha\sum_{n=1}^{N-1}\phi_{cn}(t)\Big(\alpha\hat{w}_c^T(t)\phi_{cn}(t)+r_n(t)-\hat{w}_c^T(t-1)\phi_{cn}(t-1)\Big)^T||^2\\ &-2tr\Big[\alpha\Big(I-l_c\alpha^2\times||\phi_c(t)||^2-\sum_{n=1}^{N-1} l_c\alpha^2||\phi_{cn}(t)||^2\Big)\zeta_c(t)\\ &\times\Big(\alpha w_c^T\phi_c(t)+r(t)-\tilde{w}_c^T(t-1)\phi_c(t-1)\\ &-\sum_{n=1}^{N-1}(\alpha\hat{w}_c^T(t)\phi_{cn}(t)+r_n(t)-\hat{w}_c^T(t-1)\phi_{cn}(t-1))\Big)^T\Big]\end{aligned} \tag{3.22}$$

The upper bound of Eq. (3.22) can be obtained by applying Cauchy-Schwarz inequality for the fourth term. Thus,

$$\begin{aligned}\Delta V_1(t) \le &-\alpha^2||\zeta_c(t)||^2-\alpha^2(1-l_c\alpha^2||\phi_c(t)||^2\\ &-\sum_{n=1}^{N-1} l_c\alpha^2||\phi_{cn}(t)||^2)||\zeta_c(k)||^2+||\alpha w_c^T\phi_c(t)+r(t)-\hat{w}_c^T(t-1)\phi_c(t-1)\\ &-l_c\alpha\sum_{n=1}^{N-1}\phi_{cn}(t)(\alpha\hat{w}_c^T(t)\phi_{cn}(t)+r_n(t)\\ &-\hat{w}_c^T(t-1)\phi_{cn}(t-1))^T||^2\alpha^2||\zeta_c(t)||^2-\alpha^2\Big(1-l_c\alpha^2||\phi_c(t)||^2\\ &-\sum_{n=1}^{N-1} l_c\alpha^2||\phi_{cn}(t)||^2\Big)\times||\zeta_c(t)+w_c^T\phi(t)+\alpha^{-1}r(t)\\ &-\alpha^{-1}\hat{w_c^T}(k-1)\times\phi_c(k-1)+\sum_{n=1}^{N-1} w_c^T\phi(t)\\ &+\alpha^{-1}r_l(t)-\alpha^{-1}\hat{w_c^T}(k-1)\times\phi_{cn}(k-1)||^2.\end{aligned} \tag{3.23}$$

Then,

$$\begin{aligned}\Delta V_1(t) \leq & -\alpha^2||\zeta_c(t)||^2+\frac{1}{2}||\zeta_c(t-1)||^2+2||\alpha w_c^T\phi_c(k)+r(t)\\ & -\hat{w}_c^T(t-1)\phi_c^T(t-1)||^2+2\sum_{n=1}^{N-1}||\alpha w_c^T\phi_c(k)+r_l(t)\\ & -\hat{w}_c^T(t-1)\phi_{cn}^T(t-1)||^2-\alpha^2\Big(1-l_c\alpha^2||\phi_c(t)-\sum_{n=1}^{N-1}l_c\alpha^2||\phi_{cn}(t)||^2\Big) \qquad (3.24)\\ & \times||\zeta_c(t)+w_c^T\phi(t)+\alpha^{-1}r(t)-\alpha^{-1}\hat{w_c^T}(k-1)\times\phi_c(k-1)\\ & +\sum_{n=1}^{N-1}\Big(w_c^T\phi(t)+\alpha^{-1}r_l(t)-\alpha^{-1}\hat{w_c^T}(k-1)\times\phi_{cn}(k-1)||^2\Big).\end{aligned}$$

For ΔV_2,

$$\Delta V_2(t)=\frac{1}{2}\Big(||\zeta_c(k)||^2-||\zeta_c(k-1)||^2\Big). \qquad (3.25)$$

ΔV_3 can be written as:

$$\Delta V_3=\frac{1}{\gamma l_a}tr\Big(\tilde{w}_a^T(t+1)\tilde{w}_a(t+1)-\tilde{w}_a^T(t)\tilde{w}_a(t)\Big). \qquad (3.26)$$

where,

$$\begin{aligned}\hat{w}_a(t+1)=\hat{w}_a(t)-l_a\alpha\phi_a(t)\hat{w}_c^T(t)C_a(t)\times\Big(\hat{w}_c^T(t)\phi_c(t)\Big)^T\\ -l_a\alpha\sum_{n=1}^{N-1}\phi_{an}(t)\hat{w}_c^T(t)C_{an}(t)\times\Big(\hat{w}_c^T(t)\phi_{cn}(t)\Big)^T.\end{aligned} \qquad (3.27)$$

where, $C(t)$ is a matrix of $N_{hc}\times n$ dimension, and its elements can be expressed as

$$C_{ki}(t)=\frac{1}{2}\Big(1-\phi_{c_k}^2(t)\Big)w_{c_{k,m+i}},k=1,...,N_{hc},i=1,...n. \qquad (3.28)$$

Substituting Eq. (3.27) into Eq. (3.26), and using the property $tr(AB) = tr(BA)$:

$$\begin{aligned}\Delta V_3 =& \frac{1}{\gamma}\Big\{ -2\hat{w}_c^T(t)C(t)\zeta_a(t)\Big[\hat{w}_c^T\phi_c(t)\Big]^T - 2\sum_{n=1}^{N-1}\hat{w}_c^T(t)C_n(t)\zeta_{an}(t) \\ & \times [\hat{w}_c^T\phi_{cn}(t)]^T + l_a||\hat{w}_c^T(t)C(t)||^2||\phi_a(t)||^2||\hat{w}_c^T(t)\phi_c(t)||^2 \\ & + l_a\sum_{n=1}^{N-1}||\hat{w}_c^T(t)C_l(t)||^2||\phi_{an}(t)||^2||\hat{w}_c^T(t)\phi_{cn}(t)||^2\Big\}\end{aligned} \tag{3.29}$$

With further simplification,

$$\begin{aligned}\Delta V_3 =& \frac{1}{\gamma}\Big\{ -[||\hat{w}_c^T(t)C(t)||^2 - l_a||\hat{w}_c^TC(t)||^2||\phi_a(t)||^2\Big] \times ||\hat{w}_c^T(t)\phi_c(t)||^2 \\ & - ||\hat{w}_c^T(t)C(t)||^2||\zeta_a(t)||^2 + ||\hat{w}_c^T(t)\phi_c(t) - \hat{w}_c^T(t)C(t)\zeta_a(t)||^2 \\ & - \sum_{n=1}^{N-1}\Big[||\hat{w}_c^T(t)C_l(t)||^2 - l_a||\hat{w}_c^T(t)C_l(t)||^2||\phi_{an}(t)||^2\Big] \times ||\hat{w}_c^T(t)\phi_{cn}(t)||^2 \\ & - \sum_{n=1}^{N-1}||\hat{w}_c^T(t)C_l(t)||^2||\zeta_{an}(t)||^2 + \sum_{n=1}^{N-1}||\hat{w}_c^T(t)\phi_{cn}(t) - \hat{w}_c^T(t)C_l(t)\zeta_{an}(t)||^2\}\end{aligned} \tag{3.30}$$

Using Cauchy-Schwarz inequality,

$$\begin{aligned}\Delta V_3 \leq& \frac{1}{\gamma}\Big\{ -\Big[||\hat{w}_c^T(t)C(t)||^2 - l_a||\hat{w}_c^TC(t)||^2||\phi_a(t)||^2\Big] \times ||\hat{w}_c^T(t)\phi_c(t)||^2 \\ & + ||\hat{w}_c^T(t)C(t)||^2||\zeta_a(t)||^2 + 4||\hat{w}_c^T(t)\phi_c(t)||^2 + 4||\zeta_c(t)||^2 \\ & - \sum_{n=1}^{N-1}\Big[||\hat{w}_c^T(t)C_l(t)||^2 - l_a||\hat{w}_c^T(t)C_l(t)||^2||\phi_{an}(t)||^2\Big] \times ||\hat{w}_c^T(t)\phi_{cn}(t)||^2 \\ & - \sum_{n=1}^{N-1}||\hat{w}_c^T(t)C_l(t)||^2||\zeta_{an}(t)||^2 + 4\sum_{n=1}^{N-1}\Big[||\hat{w}_c^T(t)\phi_{cn}(t)||^2 + ||\zeta_{cn}(t)||^2\Big]\Big\}\end{aligned} \tag{3.31}$$

Substituting Eq. (3.24), (3.25), and (3.31) into Eq. (3.15), the first difference of the

Lyapunov function candidate:

$$\begin{aligned}\Delta V(t) \leq & -\left(\alpha^2-\frac{1}{2}-\frac{4}{\gamma}\right)||\zeta_c(t)||^2-\alpha^2\left(1-l_c\alpha^2||\phi_c(t)-\sum_{n=1}^{N-1}l_c\alpha^2||\phi_{cn}(t)||^2\right)\\
&.||\zeta_c(t)+w_c^T\phi(t)+\alpha^{-1}r(t)-\alpha^{-1}\hat{w_c^T}(k-1)\times\phi_c(k-1)\\
&+\sum_{n=1}^{N-1}\left(w_c^T\phi(t)+\alpha^{-1}r_l(t)-\alpha^{-1}\hat{w_c^T}(k-1)\times\phi_{cn}(k-1)||^2\right)-\frac{1}{\gamma}[||\hat{w}_c^T(t)C(t)||^2\\
&-l_a||\hat{w}_c^TC(t)||^2||\phi_a(t)||^2]\times||\hat{w}_c^T(t)\phi_c(t)||^2-\frac{1}{\gamma}\Big[||\sum_{n=1}^{N-1}[\hat{w}_c^T(t)C_l(t)||^2\\
&-\sum_{n=1}^{N-1}l_a||\hat{w}_c^T(t)C_l(t)||^2||\phi_{an}(t)||^2]\times||\hat{w}_c^T(t)\phi_{cn}(t)||^2\Big]\\
&+2||\alpha w_c^T\phi_c(k)+r(t)-\hat{w}_c^T(t-1)\phi_c^T(t-1)||^2\\
&+2\sum_{n=1}^{N-1}||\alpha w_c^T\phi_c(k)+r_l(t)-\hat{w}_c^T(t-1)\times\phi_{cn}^T(t-1)||^2\\
&+\frac{1}{\gamma}\Big\{||\hat{w}_c^T(t)C(t)||^2||\zeta_a(t)||^2+\frac{4}{\gamma}||\hat{w}_c^T(t)\phi_c(t)||^2\\
&-\sum_{n=1}^{N-1}||\hat{w}_c^T(t)C_l(t)||^2||\zeta_{an}(t)||^2+\frac{4}{\gamma}\sum_{n=1}^{N-1}\Big[||\hat{w}_c^T(t)\phi_{cn}(t)||^2+||\zeta_{cn}(t)||^2\Big]\Big\}\end{aligned} \tag{3.32}$$

Choose

$$\frac{1}{\sqrt{2}}<\alpha<1, l_c<\frac{1}{N\alpha^2||\Phi_c(t)||^2}, l_a<\frac{1}{N||\phi_a(t)||^2} \tag{3.33}$$

and select γ satisfying

$$\gamma>\frac{4}{(\alpha^2-\frac{1}{2})}. \tag{3.34}$$

Then:

$$\begin{aligned}\Delta V(t) \leq & -\Big(\alpha^2-\frac{1}{2}-\frac{4}{\gamma}\Big)||\zeta_c(t)||^2-\alpha^2\Big(1-l_c\alpha^2||\phi_c(t)-\sum_{n=1}^{N-1}l_c\alpha^2||\phi_{cn}(t)||^2\Big)\\ & .||\zeta_c(t)+w_c^T\phi(t)+\alpha^{-1}r(t)-\alpha^{-1}\hat{w_c^T}(k-1)\times\phi_c(k-1)+\sum_{n=1}^{N-1}\Big(w_c^T\phi(t)\\ & +\alpha^{-1}r_l(t)-\alpha^{-1}\hat{w_c^T}(k-1)\times\phi_{cn}(k-1)||^2\Big)-\frac{1}{\gamma}\Big[||\hat{w}_c^T(t)C(t)||^2\\ & -l_a||\hat{w}_c^TC(t)||^2||\phi_a(t)||^2\Big]\times||\hat{w}_c^T(t)\phi_c(t)||^2-\frac{1}{\gamma}[\sum_{n=1}^{N-1}\Big[||\hat{w}_c^T(t)C_l(t)||^2\\ & -\sum_{n=1}^{N-1}l_a||\hat{w}_c^T(t)C_l(t)||^2||\phi_{an}(t)||^2\Big]\times||\hat{w}_c^T(t)\phi_{cn}(t)||^2]+D^2\end{aligned} \tag{3.35}$$

where D^2 is defined as:

$$\begin{aligned}D^2 =& 2||\alpha w_c^T\phi_c(k)+r(t)-\hat{w}_c^T(t-1)\phi_c^T(t-1)||^2+2\sum_{n=1}^{N-1}||\alpha w_c^T\phi_c(k)+r_l(t)\\ & -\hat{w}_c^T(t-1)\phi_{cn}^T(t-1)||^2+\frac{1}{\gamma}\Big\{||\hat{w}_c^T(t)C(t)||^2||\zeta_a(t)||^2+\frac{4}{\gamma}||\hat{w}_c^T(t)\phi_c(t)||^2\\ & -\sum_{n=1}^{N-1}||\hat{w}_c^T(t)C_l(t)||^2||\zeta_{an}(t)||^2+\frac{4}{\gamma}\sum_{n=1}^{N-1}\Big[||\hat{w}_c^T(t)\phi_{cn}(t)||^2\Big]\Big\}.\end{aligned} \tag{3.36}$$

Applying the Cauchy-Schwarz inequality,

$$\begin{aligned}D^2 \leq & 8\Big(||\alpha^2||w_c^T\phi_c(t)||^2+r^2(t)+\frac{1}{4}||\hat{w}_c^T(t-1)\phi_c(t-1)||^2+\frac{1}{4}||\hat{w}_c^T\phi_c(t-1)||^2\Big)\\ & +8\Big(\sum_{n=1}^{N-1}||\alpha w_c^T\phi_c(t)||^2+r_l^2(t)+\frac{1}{4}||\hat{w}_c^T(t-1)\phi_{cn}(t-1)||^2\\ & +\frac{1}{4}||\hat{w}_c^T\phi_{cn}(t-1)||^2\Big)+\frac{2}{\gamma}\Big\{||\hat{w}_c^T(t)C(t)||^2||\times\Big(||w_a^T\phi_a(t)||^2+||w_a^T\phi_a(t)||^2\Big)\\ & +\frac{4}{\gamma}||\hat{w}_c^T(t)\phi_c(t)||^2+\frac{2}{\gamma}\sum_{n=1}^{N-1}||\hat{w}_c^T(t)C_l(t)||^2\\ & \times(||w_a^T\phi_{an}(t)||^2+||w_a^T\phi_{an}(t)||^2)+\frac{4}{\gamma}\sum_{n=1}^{N-1}\Big[||\hat{w}_c^T(t)\phi_{cn}(t)||^2\Big]\Big\}\end{aligned} \tag{3.37}$$

$$
\begin{aligned}
&\leq \Big(8\alpha^2+4+\frac{4}{\gamma}\Big)w_{cm}^2\phi_{cm}^2+\frac{4}{\gamma}w_{cm}^2C_m^2w_{am}^2\phi_{am}^2+8r_m^2 \\
&\quad+\sum_{n=1}^{N-1}\Big[\Big(8\alpha^2+4+\frac{4}{\gamma}\Big)w_{cmn}^2\phi_{cmn}^2+\frac{4}{\gamma}w_{cmn}^2C_{mn}^2w_{amn}^2\phi_{amn}^2+8r_{mn}^2\Big] \\
&\quad=D_m^2+\sum_{n=1}^{N-1}D_{mn}^2
\end{aligned}
\tag{3.38}
$$

where $w_{cm}, w_{am}, \phi_{cm}, \phi_{am}, C_m$ and r_m are the upper bounds of $w_c, w_a, \phi_c(t), \phi_a(t), C(t)$ and $r(t)$, respectively. If Eq (3.33) holds, then for any:

$$
||\zeta_c(t)|| > \sqrt{\frac{D_m^2+\sum_{n=1}^{N-1}D_{mn}^2}{\alpha^2-\frac{1}{2}-\frac{4}{\gamma}}}, \tag{3.39}
$$

the first difference $\Delta L(t) \leq 0$ holds. According to the Lyapunov extension theorem, this demonstrates that the errors between the optimal network weights w_c^*, w_a^* and their respective estimations $\hat{w}_c(t)$, $\hat{w}_a(t)$ are uniformly ultimately bounded (UUB), respectively. This verifies the stability of the proposed method.

3.3 Simulation and Evaluation

The simulation results presented here are from two case studies: a cart-pole balancing problem and a triple-link inverted pendulum balancing problem. For both cases, a comparison among the traditional ADP in [24], the authors' most recent ADP in [89] and the proposed ADP approach is presented.

3.3.1 Cart-pole Balancing Problem

The ultimate objective of cart-pole balancing task is to generate appropriate control action force applied on the cart, so as to balance the single pole mounted on the cart. The

model of the system can be described with the following differential equations:

$$\frac{\partial^2\theta}{\partial t^2} = \frac{g\sin\theta + \cos\theta[-F - ml\theta^2\sin\theta + \mu_c \mathrm{sgn}(\dot{x})] - \frac{\mu_p\dot{\theta}}{mn}}{l\left(\frac{4}{3} - \frac{m\cos^2\theta}{m_c+m}\right)} \tag{3.40}$$

$$\frac{\partial^2 x}{\partial t^2} = \frac{F + ml[\dot{\theta}^2\sin\theta - \ddot{\theta}\cos\theta] - \mu_c \mathrm{sgn}(\dot{x})}{m_c + m}. \tag{3.41}$$

where the acceleration $g = 9.8m/s^2$, the mass of the cart m_c=1.0 kg, the mass of the pole m=0.1 kg, half-pole length l=0.5 m, the coefficient of friction of the cart μ= 0.0005, and the coefficient of friction of the pole μ_p=0.000002. The force, F applied to the cart is in the range [-10, 10] N. A pole is considered fallen when the angle is outside the range of [-12°, 12°] or the cart is beyond the range of [-2.4, 2.4] m. To evaluate the statistical performance, 100 runs is set with different initial state conditions. Specifically, the angle (θ) and position (x) of the pole in each of these initial states are uniformly generated within [-0.05, 0.05] radians and [-0.5, 0.5] m, while the angular velocity ($\dot{\theta}$) and velocity of the cart ($\dot{x}$) are both 0.

The proposed approach has been compared with the traditional ADP method presented in [24] with the same parameter setting, however continuous control is used here. The experience replay based ADP (ER based ADP) method presented in [89] is also included as well for comparison. These results are summarized in Table 3.1. From experience, the experience replay size is chosen as L=10. Different sizes of this experience replay has been tried, and this is the best size from the trial-and-error experience. For noise free conditions, PER based ADP provided 42.8% improvement over ER based ADP. To identify whether this is significant improvement or not, ANOVA test is

Table 3.1. Performance evaluation for cart-pole balancing task for 100 runs. The 2nd, 3rd and the 4th column are with the traditional ADP method, while 5th, 6th and 7th column are with experience replay ADP method. The 8th, 9th and 10th column are with the proposed ADP method.

Noise type	Traditional ADP [24]			ER based ADP [89]			PER based ADP		
	SR^*	$Mean^\dagger$	$\sigma^\sharp$	SR^*	$Mean^\dagger$	$\sigma^\sharp$	SR^*	$Mean^\dagger$	$\sigma^\sharp$
Noise free	99	47.9	58.1	100	32.7	35.6	100	18.7	18.5
Uniform 5% a.	98	57.8	72.7	100	36.4	50.7	100	19.8	17.8
Uniform 10% a.	99	73.4	130.9	100	59.2	111.1	100	23.5	26.3
Uniform 5% s.	98	61.1	93.1	100	59.9	84.1	100	21.9	27.3
Uniform 10% s.	98	61.4	142.3	100	68.8	112.2	100	34.4	55.0

SR^*: 'Success rate' for 100 runs (in %)
$Mean^\dagger$: Required average number of trials to succeed
$\sigma^\sharp$: Standard deviation of trials to succeed
a.:actuator, s.:sensor

used with following hypothesis:

- Null hypothesis: there is no significant difference in mean.
- Alternative hypothesis: there is significant difference in mean.

It has been tested with 99% confidence interval and it has been found that p-value for noise-free condition is 5.80e-04 which is very much less than 0.01. Thus the inference is to reject null hypothesis and the conclusion is that there is significant difference of mean value for noise-free condition. Similarly, for different noise conditions as in Table 3.1, the p-values are: 0.0022, 0.0020, 2.73e-05, 0.006. Therefore, it is concluded that PER based ADP achieved significant improvement over ER based ADP at 99% confidence interval. It can also seen from Fig. 3.5 that there is significant improvement in required average number of trials to succeed for PER ADP in comparison to ER ADP for 10% uniform sensor noise condition.

For analyzing the details of prioritized experience replay, a random instance has

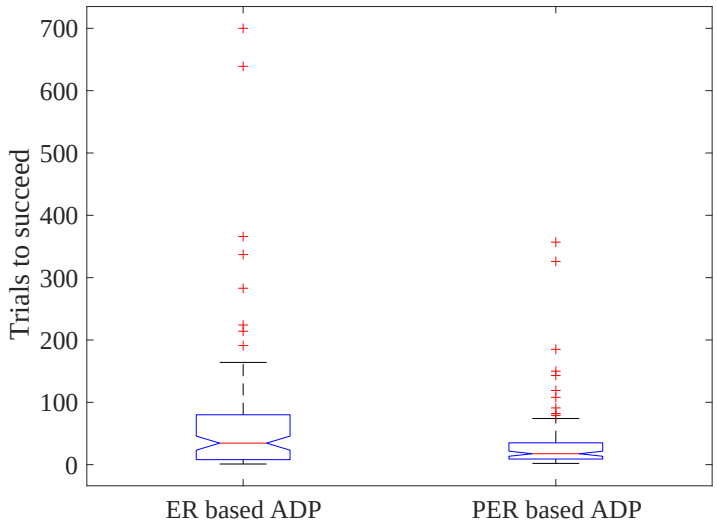


Figure 3.5. Box plot comparison of two methods: ER based ADP and PER based ADP with respect to trials to succeed for uniform 10% sensor noise condition.

been selected with no failure experience (i.e. $r(t)$=0 for all samples) in experience replay and the Figs. 3.6(a) and 3.6(b) were extracted. Fig. 3.6(a) depicts the error=$|Error_k|$ as calculated by Eq. (3.5). Based on this error, Eq. (3.3) is used to calculate the probability of samples in experience replay which is represented in Fig. 3.6(b). Fig. 3.6(b) also shows rank represented by numbers with 1 representing the tuple with highest error and having the highest probability to be selected. It can be concluded from these figures that experience tuple with highest error represents tuple with highest probability of being selected for training action and critic networks. This accelerates the training process compared to uniform random sampling method in ER based ADP [89]. This explains the improvement in trial to succeed for different noise conditions as shown in Table 3.1.

3.3.2 Triple-link Inverted Pendulum Balancing Problem

The same triple-link inverted pendulum balancing task as in [24, 80] has been considered. The ultimate objective is to generate appropriate control action force applied

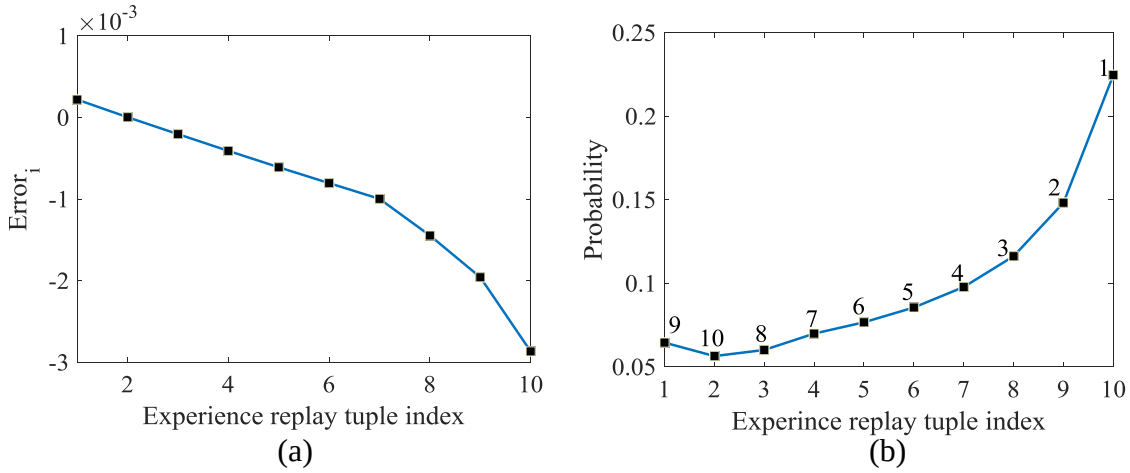


Figure 3.6. Typical curves: (a) error curve (b) probability curve with rank represented by numbers from 1 to 10 for 10 tuples in experience replay with no failure experiences.

on the cart so as to balance the triple-link mounted on the cart. However, the level of difficulty is much higher compared to a simple cart-pole because there are three joints: mount joint, first joint and second joint. This makes the system highly unstable and therefore is a challenging problem to solve. The model of the system can be described with the following differential equation:

$$F(q)\frac{\partial^2 q}{\partial t^2} = -G\left(q, \frac{\partial q}{\partial t}\right)\frac{\partial q}{\partial t} - H(q) + L(q,u) \tag{3.42}$$

where the components and parameters used are the same as described in [24, 80].

There are eight state variables in this model: position of the cart on the track, x; vertical angle of the first link joint to the cart, θ_1; vertical angle of the second link joint to the first link, θ_2; vertical angle of the third link joint to the second link, θ_3; cart velocity, $\dot{x}$; angular velocity of θ_1, i.e. $\dot{\theta}_1$; angular velocity of θ_2, i.e. $\dot{\theta}_2$; and angular velocity of θ_3, i.e. $\dot{\theta}_3$. The constraints for the triple-linked inverted pendulum are (1) the cart track extends 1.0 m to both sides from the center point; (2) the voltage applied to the motor

should be within [-30 V, 30 V]; (3) each link angle should be within the range of $[-20°, 20°]$ with respect to the vertical axis. Here, condition (2) is guaranteed by using a sigmoid function. While for the other two conditions, if either one fails or both fail, the system will be provided with an external reinforcement signal $r = -1$ at the moment of failure, otherwise $r = 0$ all the time.

100 runs has been conducted for the triple link pendulum. Similar to the cart-pole problem, different initial starting states has been used for each run. Specifically, the three angles and angular velocity of the triple links are set to be uniformly within the range of [-0.02, 0.02] radians and [-0.5, 0.5] radians/s, respectively. As for x and $\dot{x}$, their initial states are set to zero. The descriptions of the notation used are defined as following. The initial learning rates of critic and action network are initialized at 0.1 which is decreased by 0.05 every five time steps until it reaches 0.005 and stays at 0.005 thereafter. The internal cycles for critic and action networks are set as: N_c=40 and N_a=50 respectively. The internal error threshold for critic and action networks are set as: T_c=0.05 and T_a=0.005 respectively.

The summary of the simulation results with different noise conditions is presented in Table 3.2. It shows comparison for three control approaches: traditional ADP [24], ER based ADP [89], and the proposed method i.e. prioritized experience replay based ADP (PER based ADP). The experience replay size has been set as a short term experience replay size of L=10 and number of training cycles for action and critic networks as N=10. For noise free condition, ER based ADP provided 42.73% improvement in required average number of trials to succeed. PER based ADP provided 51.63% improvement and reduced the standard deviation for required average number of trials to succeed. In

Table 3.2. Performance evaluation for triple-link inverted pendulum balancing task for for 100 runs. The 2nd, 3rd and the 4th column are with the traditional ADP method, while 5th, 6th and 7th column are with experience replay ADP method. The 8th, 9th and 10th column are with the proposed ADP method.

Noise type	Traditional ADP [24]			ER based ADP [89]			PER based ADP		
	SR^*	$Mean^\dagger$	$\sigma^\sharp$	SR^*	$Mean^\dagger$	$\sigma^\sharp$	SR^*	$Mean^\dagger$	$\sigma^\sharp$
Noise free	100	468.5	488.20	100	268.3	305.8	100	226.6	256.7
Uniform 5% a.	99	500.6	511.54	100	298.6	353.9	100	228.7	305.8
Uniform 10% a.	96	544.7	465.00	100	323.6	367.0	100	233.5	301.6
Uniform 5% s.	96	609.0	608.63	99	311.4	353.8	100	226.7	289.4
Uniform 10% s.	95	612.9	624.64	100	352.9	380.3	100	253.8	332.2

SR*: 'Success rate' for 100 runs (in %)
$\bar{x}^\dagger$: Required average number of trials to succeed
$\sigma^\sharp$: Standard deviation of trials to succeed
a.:actuator, s.:sensor

average for all noise conditions, from Table 3.2, the experience replay based ADP could achieve only 43% improvement of required average number of trails to succeed. However, the proposed PER based ADP approach improved the required average number of trials to succeed by 56.89% compared to traditional ADP for this triple-link balancing task.

3.4 Summary

Prioritized experience replay is a method that can do efficient learning from experience replay. This project integrated prioritized experience replay to model free ADP controller, thereby preserving the model-free capability of ADP. First, one step backward state-action information is used for the design of experience replay tuple to avoid the model network usage. Second, a systematic approach is proposed to integrate history experience in both critic and action networks of ADP controller design. Rank based prioritization is used to train ADP controller in two control benchmarks: cart-pole and triple-link balancing task for accelerating the training process.

CHAPTER 4 Smart Grid Application 1: Supplementary Adaptive Dynamic Programming Controller for Virtual Synchronous Machine

The project in this chapter is a joint work with Dr. Reinaldo Tonkoski and Dipesh Shrestha who kindly provided benchmark of this work. Increased photovoltaics (PV) penetration in PV-hydro integrated systems decreases relative inertia of the system due to lack of rotating masses like in conventional synchronous generator. This leads to frequency instability problem and can have a significant impact on its dynamic performance and transient stability [91]. Load/generation changes and PV fluctuations in such systems can cause large frequency deviations at a high rate of change of frequency (ROCOF) [92]. A solution to improve the transient stability of such integrated PV-hydro micro-grids is to add inertia into the system using virtual synchronous machines (VSM). A VSM can emulate virtual inertia by using small amounts of stored energy in the battery or super-capacitor controlled by a power electronics converter [93]. Renewable based generation, such as PV, use power electronics converter system as an interface between the source and the grid. The batteries are usually used in renewable energy systems with inverters because of the intermittent nature of renewable energy sources like solar. This combination of PV and battery can also be used as VSM. Conventionally, this type of converter is controlled using the standard decoupled d-q vector, proportional-integral (PI) control approach whose control strategy is inherently limited due to the difficulty in tuning the proportional-integral controller parameters [94]. Such challenge motivates the development of adaptive control approach for the VSM application.

In [95], a strong ability of neural network vector control is shown in the case of

permanent magnet synchronous motors to tolerate system disturbances, rapidly changing reference commands, and satisfy control requirements for complex electric drive vehicle (EDV) needs as in [96]. In [94] and [97], the vector control of a grid-connected rectifier/inverter using an artificial neural network has been claimed. In [98], an efficient grid-connected controller is produced for a three-phase grid-connected converter to solve a tracking problem under disturbances. However, d-axis current under variable grid-filter inductance conditions shows high spikes at transient when tracking reference current. In [99], recurrent neural network (RNN) training with the Levenberg-Marquardt algorithm is used for optimal control of a grid-connected converter, but this does not involve on-line training and requires well-trained RNN controller. In [100], current harmonics has been mitigated by using fuzzy controller based 3-phase 4-wire shunt active filter with $I_d - I_q$ control strategies, but there is no clue of performance of this controller during sudden load change condition. In recent years, adaptive dynamic programming (ADP) has been widely used for balancing a inverted pendulum [24], improving the system damping as well as dynamic transient stability for not only small step changes, but also large disturbances such as three phase short circuit [101–103]. ADP has also been widely used for damping oscillations in a large power system [104], as a supplementary controller for a doubly-fed induction generator (DFIG) to enhance power system stability [62, 105–107], and so on. However, none of them have applied ADP for current control of pulse width modulation (PWM) inverters used in VSM.

All of the aforementioned literatures indicate the need for developing a supplementary adaptive control approach for VSM system. The basic idea is to complement an existing controller using a supplementary controller based on ADP. In this

way, the prior knowledge of the original controller can be utilized. On the other hand, it can make the original controller become "smarter" by incorporating an actor-critic structure. The critic network is designed to evaluate the current supplementary controller, while the actor is to produce a supplementary control signal to improve the control performance [24, 62, 108]. The major contributions of the project in this chapter are as follows. Firstly, ADP controller is proposed for use as a supplementary controller in the VSM benchmark system developed by Dipesh and Ujjwol. The advantage of the online learning capability of ADP is used, which is model-free and data-driven, for improving performance of the current controller for grid-connected rectifier/inverter in renewable and electric power system acting as a VSM. Secondly, adaptation of ADP is demonstrated under different types of disturbance and fault in VSM application.

The rest of the chapter is organized as follows. Section II describes the benchmark used for this simulation. Section III illustrates the algorithm used for ADP applied here as a supplementary controller. Section IV compares the performance of the ADP controller with the conventional PI controller in two case studies: first for step change in I_d and second for unbalanced single line to ground fault in power system. Finally, Section V summarizes the chapter.

4.1 Benchmark VSM System

Fig. 4.1 depicts the schematic diagram of VSM control structure with current controlled voltage source inverter, which includes an external control loop. This loop calculates the net active power (P_{ref}) to be consumed or delivered by the battery system used to maintain the frequency and ROCOF under standard values. The inner current

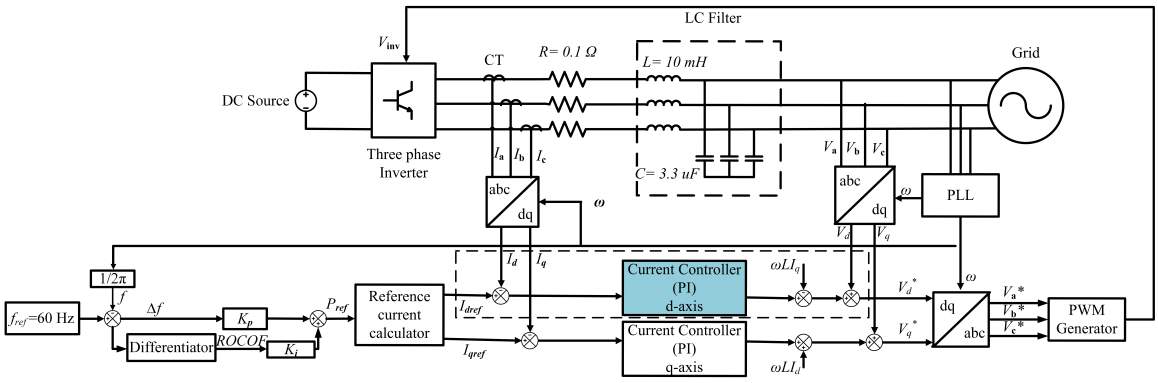


Figure 4.1. Schematic diagram of VSM current controlled voltage source inverter with closed loop current controller and phase locked loop tracked from the grid.

control loop consists of PI controller for regulating the output current of the inverter. The goal of the control scheme is to regulate the output current by providing the appropriate gate signal to the PWM inverter. The sinusoidal output currents (I_a, I_b and I_c) are transformed in to rotatory d-q frame using Park's transformation. This conversion converts I_a, I_b and I_c into dc signal I_d and I_q. Conversion into dc signals is necessary for the PI controller. Since only active power control is necessary for frequency control in VSM, only I_d control is required and I_q is always made to be zero. Then, the error signal after comparing with their reference values is fed to a PI controller, which generates reference values V_d^* and V_q^* after considering cross-coupling terms ωLI_q and ωLI_d, adding feed-forward terms V_d and V_q. These reference values undergo dq to abc transformation also known as inverse Park's transformation to generate V_a^*, V_b^*, V_c^*. Finally, they are compared with the carrier signal through a comparator to generate gate signals for the PWM inverter.

Whenever there is a change in irradiance of the sun, there will be sudden change in PV output causing change in frequency. The reference value of I_d is generated by reference current calculator in the external current control loop whose input is the

difference between reference frequency of 60 Hz and actual frequency of the system. Here, I_q is made zero by default to control active power only. Phase locked loop is used to measure the frequency of the system and the phase angle of the voltage signals. The net active power to be absorbed or supplied based on the error in frequency and rate of change of frequency is calculated as in [109]. The d-axis reference current is generated based on this net active power. A PI-controller is used to track this reference current and absorb or supply the net active power to maintain the frequency of the system. There are certain limitations in the transient performance of PI controller which will be discussed in later sections.

The simplified d-q equivalent model for current controller design is shown in Fig. 4.2. The d-q equivalent model consists of two circuits, one each for the d-axis and q-axis. Based on this figure, the transfer function of the plant is represented by following equation:

$$G_{plant}(s) = \frac{I_L(s)}{V_{inv}(s)} = \frac{\frac{1}{L}}{s + \frac{R}{L}} \tag{4.1}$$

where, V_{inv} is the unfiltered output voltage of the inverter, L and C are the inductance and capacitance values of the LC filter and R is the resistance of the inductor which was assumed to be 0.1 Ω. The frequency response of the plant was then obtained. The three phase inverter was designed with a switching frequency of 10 kHz. The cross–over frequency of the current controller was selected to be 1 kHz (or 6283.18 rad/s) which is a decade below the switching frequency. The 1 kHz cross-over frequency provides fast enough speed response for the designed current controller. Similarly, the phase margin of the loop transfer function (combined transfer function of the plant and controller) was

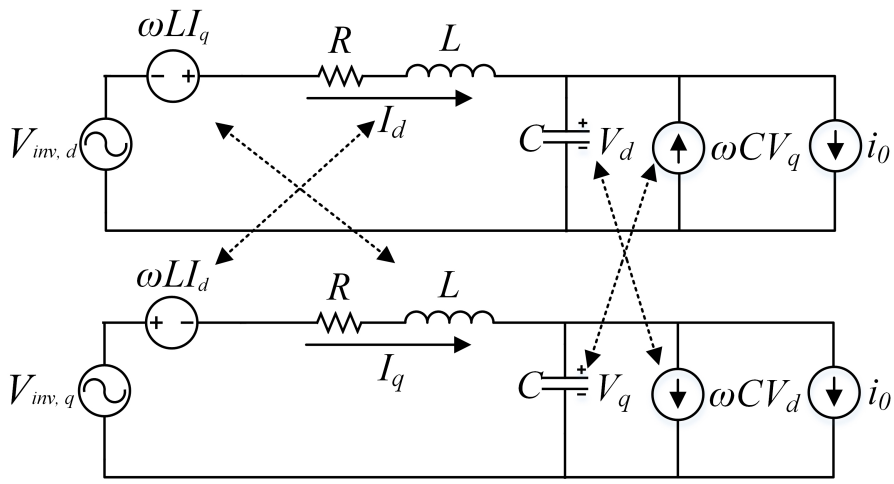


Figure 4.2. d-q equivalent model of three-phase inverter with cross-coupling and feed-forward terms included.

selected to be 45° which provides a large overshoot as presented later in Section IV. Overshoot reduction by increasing phase margin is not desired because the system response will become slow [110]. It was desirable to achieve a high gain at the low frequencies to ensure that the overall system had minimal steady state error and a higher attenuation at high frequencies to suppress the switching noises in the inverter. Taking these points into consideration, a modified Proportional Integral type 2 controller was selected as the appropriate controller for the current control. The generic transfer function used for the PI type 2 controller is:

$$C_{PI2}(s) = K_{PI}\frac{1+s\tau}{s\tau}\frac{1}{1+sT_p} \tag{4.2}$$

where, K_{PI} is the proportional gain, τ is the time constant of the controller and T_p is the time constant of the additional pole at high frequency used to attenuate high frequency noises. This modified PI type 2 controller is tuned as described in [111].

In case of sudden load change, inverter has to supply current to the grid accordingly

for which command is given by gate signal to the inverter. Because of sudden load change, frequency fluctuation occurs at the output of inverter which is sensed by PLL and outer frequency control loop generates reference I_d and I_q current and PI controller is tuned in the benchmark so that actual I_d and I_q current tracks their respective reference values. However, in doing so, overshoot occurs after step change and also there is some steady state error as discussed in Section IV later for I_d current with PI controller only.

4.2 Controller Design Based on Adaptive Dynamic Programming

In order to reduce the overshoot and steady state error mentioned in the previous section, ADP is connected as a supplementary controller as shown in Fig. 4.3.

The basic idea in adaptive critic design is to adapt the weights of the critic network to make the approximating or optimal cost function, $J^*(X(t))$, satisfy the Bellman principle of optimality [42], given by:

$$J^*(X(t)) = \min_{u(t)}\{J^*(X(t+1)) + r(X(t)) - U_c\} \tag{4.3}$$

The optimal supplementary controller can be computed from:

$$u^*(X(t)) = arg\min_{u(t)} J^*(X(t)) \tag{4.4}$$

Eq. (4.3) is the modified Bellman equation where $r(X(t))$ is the immediate cost incurred by $u(t)$ at time t, and U_c is a heuristic term used to balance [112]. This equation cannot be analytically solved in general, thus the problem of optimal current control can be solved iteratively and approximately by using ADP as a supplementary controller.

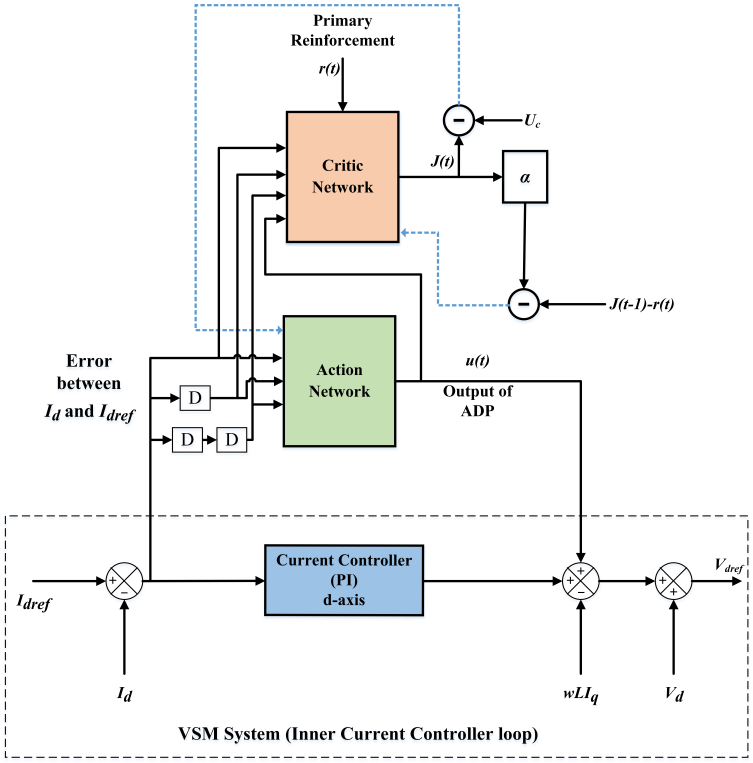


Figure 4.3. ADP controller used as a supplementary controller (Refer to Fig. 4.1 for implementation in VSM system).

The reinforcement signal $r(t)$ is provided from the external environment and may be as simple as either a "0" or "-1" corresponding to "success" or "failure" respectively [24]. In this chapter, the reinforcement signal for the critic network is given as follows:

$$r(t) = -c(a_1 X_1^2 + a_2 X_2^2 + a_3 X_3^2) \tag{4.5}$$

where, c, a_1, a_2 and a_3 are the coefficients of this quadratic equation.

X_1=error signal between I_d and I_{dref};

X_2=one time step delayed error signal;

X_3=two time step delayed error signal;

$X(t)$=[X_1, X_2, X_3].

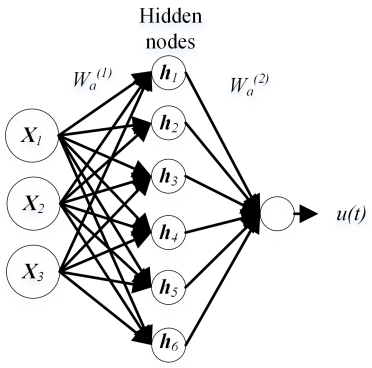


Figure 4.4. Action neural network with 3 inputs, 6 hidden neurons, and 1 output neuron.

The above two delayed signals of error signal (1 time step and 2 time step delayed) are fed into the ADP controller in order for the ADP controller to work properly.

Fig. 4.4 demonstrates the 3-6-1 architecture of the action network. It is a three-layer neural network with 6 hidden neurons. The inputs to the action network are the measured system state vector X_1, its one time-step delayed values X_2, and two time-step delayed values X_3. The output of the action network is the supplementary control signal $u(t)$. The principle in adapting the action network is to indirectly back-propagate the error between the desired ultimate objective, denoted by U_c, and the approximate J function from the critic network. Since "0" is defined as the reinforcement signal for "success," U_c is set to "0" in the design paradigm. In the action network, the state measurements are used as inputs to create a control as the output of the network. In turn, the action network can be implemented by either a linear or a nonlinear network, depending on the complexity of the problem. The weight updating in the action network can be formulated as follows.

$$e_a(t) = J(t) - U_c(t) \tag{4.6}$$

The performance error measure to be minimized in action network is given by:

$$E_a(t) = \frac{1}{2} e_a^2(t) \tag{4.7}$$

Thus, the action network can be represented as:

$$u = nn_a(X, w_a) \tag{4.8}$$

The weight update algorithm by a gradient descent rule is given as follows:

$$w_a(t+1) = w_a(t) + \Delta w_a(t) \tag{4.9}$$

$$\Delta w_a(t) = l_a(t) \left[-\frac{\partial E_a(t)}{\partial w_a(t)} \right] \tag{4.10}$$

$$\frac{\partial E_a(t)}{\partial w_a(t)} = \frac{\partial E_a(t)}{\partial J(t)} \frac{\partial J(t)}{\partial u(t)} \frac{\partial u(t)}{\partial w_a(t)} \tag{4.11}$$

where $l_a(t) > 0$ is the learning rate of the action network at time t, which usually decreases with time to a small value, and w_a is the weight vector in the action network.

Fig. 4.5 demonstrates the 4-6-1 architecture of the critic network. Similar to the action network, it is a three-layer neural network with 6 hidden neurons. The inputs to the critic network are the measured system state vector X_1, their one time-delayed values X_2, two time-delayed values X_3, and the action network output $u(t)$. Here, $J(t)$=output of the critic element and J function approximates the discounted total reward to go.

If $R(t)$ is future accumulative reward-to-go value at time t, α is discount factor which is taken as 0.95, then

$$R(t) = r(t+1) + \alpha r(t+2) + ... \tag{4.12}$$

where, this $r(t+1)$=external reinforcement value at $t+1$. In ADP design, J function is used to approximate R (i.e. $J \rightarrow R$) [62, 104, 106]. The Bellmann error or prediction error

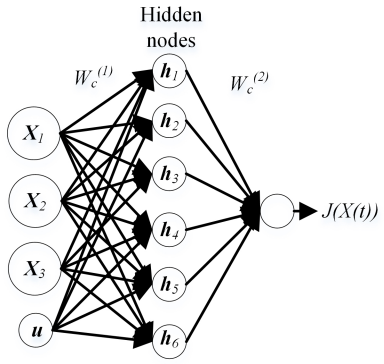


Figure 4.5. Critic neural network with 4 inputs, 6 hidden neurons, and 1 output neuron.

is given by:

$$e_c(t) = \alpha J(t) - [J(t-1) - r(t)] \tag{4.13}$$

The objective function to be minimized in critic network is given by:

$$E_c(t) = \frac{1}{2} e_c^2(t) \tag{4.14}$$

Thus, critic network can be represented as:

$$J = nn_c(X, u, w_c) \tag{4.15}$$

The weight update algorithm by a gradient descent rule for the critic network is given as follows:

$$w_c(t+1) = w_c(t) + \Delta w_c(t) \tag{4.16}$$

$$\Delta w_c(t) = l_c(t)\left[-\frac{\partial E_c(t)}{\partial w_c(t)}\right] \tag{4.17}$$

$$\frac{\partial E_c(t)}{\partial w_c(t)} = \frac{\partial E_c(t)}{\partial J(t)}\frac{\partial J(t)}{\partial w_c(t)} \tag{4.18}$$

where $l_c(t) > 0$ is the learning rate of the critic network at time t, which usually decreases with time to a small value, and w_c is the weight vector in the critic network.

The weights of the critic and action networks are updated with the algorithm as described in [24]. The objective of the ADP controller is to provide an optimal control signal that can reduce the oscillations in I_d current for efficient tracking of reference current. In both networks, normalization is used in order to confine the values of the weights into some appropriate range as follows:

$$w_c(t+1) = \frac{w_c(t)}{max||w_c(t)||}, if\ ||w_c|| > threshold \tag{4.19}$$

$$w_a(t+1) = \frac{w_a(t)}{max||w_a(t)||}, if\ ||w_a|| > threshold \tag{4.20}$$

4.3 Simulation and Evaluation

To assess the performance of the supplementary control approach using ADP, the integrated controller and the benchmark was tested in MATLAB/Simulink with ADP embedded in S-function block of Simulink. The three-phase inverter was replaced by an ideal voltage source to observe the best performance of the ADP controller. Two case studies were conducted: the first case study was step change in I_d and the second case study was an unbalanced single phase to ground fault. Only the current control loop was considered here and the influence of the outer loop control was unchanged for a fair

evaluation of the controllers.

4.3.1 Training The Adaptive Dynamic Programming Controller

The effectiveness of the supplementary controller is ensured by prior extensive training. To train the ADP for this benchmark VSM system, the following parameters were adjusted:

For critic network,

Number of inputs=4;

Number of output=1;

Number of hidden nodes=6;

Weights were initialized in the range of [-0.1.0.1].

For action network,

Number of inputs=3;

Number of output=1;

Number of hidden nodes=6;

Weights were initialized in the range of [-0.1,0.1].

Inputs were normalized in the range of [-1,1] after diving by 10 since the input current is assumed to vary in the range of 0 to 10 A for 2.5 kW inverter. Outputs were amplified by the factor of 40 since output of PI controller varies from 0 to 400. Here 10% adjustment by supplementary ADP controller is assumed. The coefficients of Eq. (4.5) were set as follows: c=1, a_1=0.4, a_2=0.3 and a_3=0.3. If system parameters like range of I_d changes, normalization of input and amplification of output may change or ADP might need training again. The following additional parameters were set in ADP.

N_c = internal cycle of the critic network;

N_a = internal cycle of the action network;

T_a = internal training error threshold for the action network;

T_c = internal training error threshold for the critic network;

N_h = Number of hidden nodes=6;

$l_a(0)$ = initial learning rate of the action network;

$l_c(0)$ = initial learning rate of the critic network;

l_a = learning rate of the action network at time t, which is decreased by 0.05 every 5 time steps until it reaches 0.005 and it stays at $l_a(f)$=0.005 thereafter;

l_c = learning rate of the critic network at time t which is decreased by 0.05 every 5 time steps until it reaches 0.005 and it stays at $l_c(f)$=0.005 thereafter.

The weights in the action and the critic networks were trained using their internal cycles, N_a and N_c, respectively. The weights of the two networks were updated for at most N_a and N_c times, respectively within each time step, or stopped once the internal training error threshold T_a and T_c have been met. For the training process, after above parameters were set in MATLAB code inside s-function block, the ADP controller was connected as shown in Fig. 4.3 to the VSM benchmark [109] shown in Fig. 4.1. Repeated simulations were performed until reduced transient overshoot was obtained. These trained weights were used for improving the transient and steady state stability for both of cases that follow. After the end of the training process, internal cycles (N_c, N_a) and learning rates (l_a, l_c) of the critic and action networks were reduced in order to avoid over-fitting in neural

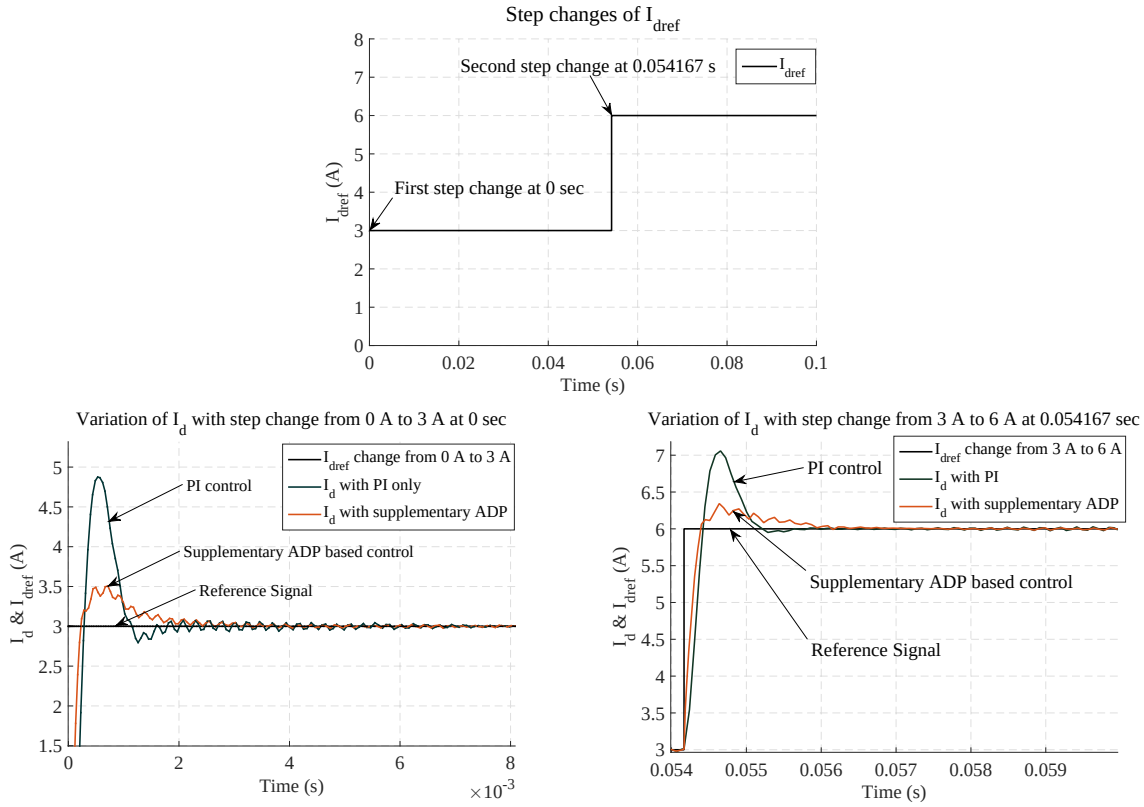


Figure 4.6. Comparision between ADP and conventional PI Controller. (a) d-axis reference current signal (I_{dref}). (b) Overshoot at first step change reduced with supplementary ADP controller and system response made faster. (c) Overshoot at second step change reduced with supplementary ADP controller and system response made faster.

networks.

4.3.2 Case Study I: Step Change in The d-axis Reference Current

The system was tested for step change in d-axis reference current from 3 A to 6 A which is represented in Fig. 4.6(a). I_d with ADP control in Fig. 4.6(b) and Fig. 4.6(c) demonstrates the d-axis current behavior of the PI controlled Grid connected inverter system acting as virtual synchronous machine. At the beginning, d-axis current changes from 0 A to 3 A at t=0 sec. ADP controller quickly regulated the d-axis currents to reference current. For simulation setup with total simulation time of 0.1 sec, step change was made from 3 to 6 A at 0.054167 sec (which implies the increase in generation from

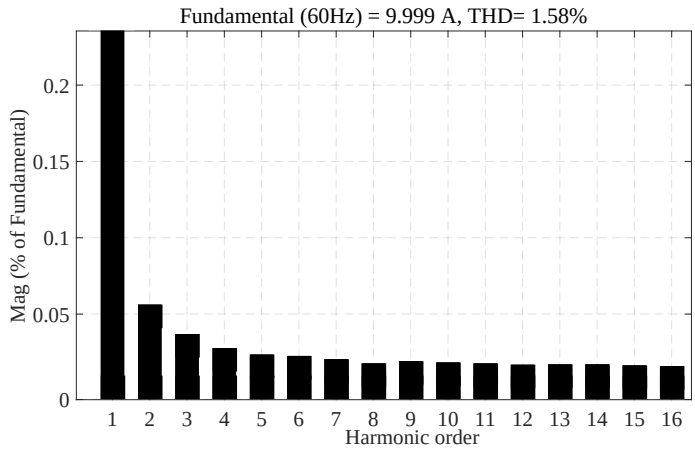


Figure 4.7. Harmonic spectrum of the grid current in the case of PI controller.

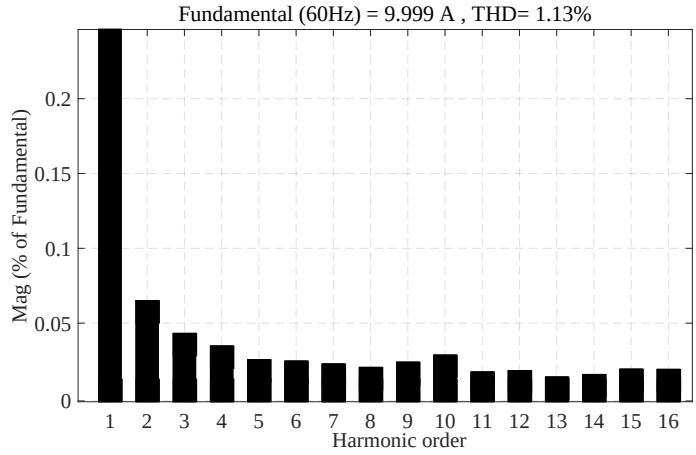


Figure 4.8. Harmonic spectrum of the grid current in the case of ADP as a supplementary controller.

PV). This particular step change time was chosen because at this time V_a, output of the inverter, is at its peak value so that ADP controller performance could be observed in the worst scenario. The experiment showed that ADP controller can be applied successfully for this system.

The simulation time was set to 0.1 sec for properly observing the three phase inverter output waveform. The inverter output voltage was not much affected with disturbance since only I_d control was performed but not I_q control. Fig. 4.6 presents a

comparison study for conventional PI and ADP as a supplementary controller. Fig. 4.6(b) and 4.6(c) shows the three advantages of ADP: first is the reduction in overshoot, second is the faster system response and third is the reduced steady state error. Table II shows the overshoot in percentage corresponding to two different control methods. The transient overshoot for first step change from 0 A to 3 A with conventional PI controller [109] is 62.49%. With ADP controller, this overshoot is reduced to 16.88% which can protect the inverter from surge currents. It can be seen that from Fig. 4.6 and Table II, ADP has the fastest response time, minimum overshoot and the best performance.

The total harmonic distortion (THD) of the grid current in the case of d-axis current control with PI controller is shown in Fig. 4.7 and ADP as a supplementary controller for current regulation is shown in Fig. 4.8. For measuring the THD, 6 cycles were taken by setting simulation time as 0.2 sec and full load reference current i.e. 10 A. In this case, the first 16 harmonic orders are shown. It can be clearly observed that ADP reduced the THD from 1.58% to 1.13%. Thus, for different values of reference current, the comparison study demonstrated that the ADP performs better than conventional PI control approach.

Table 4.1. Performance measurement of d-axis current control corresponding to Fig. 4.6

Method / Measurement	Conventional PI Controller [109]	ADP Controller
First overshoot	62.49%	16.88%
Second overshoot	17.63%	5.72%

4.3.3 Case Study II: Single-phase Ground Fault

A single phase to ground fault with fault impedance of 0.001 Ω was introduced in phase-*A* at time 0.054167 sec and removed at 0.1 sec. Fig. 4.9 shows the schematic

diagram of the setup for simulating this fault at the point of common coupling where a $Y-\Delta$ transformer is introduced on left side of the fault. The phase voltages under this conditions is shown in Fig. 4.10. Under such unbalanced fault conditions, current supplied by inverter becomes unbalanced with conventional PI controller as shown in Fig. 4.12 [113].

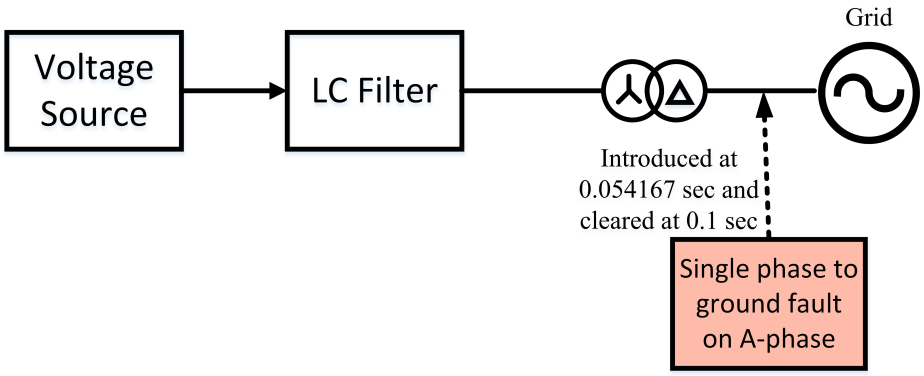


Figure 4.9. Schematic diagram of VSM benchmark system connected to grid through Δ-Y transformer and single phase unsymmetrical fault with fault impedance of Z_g.

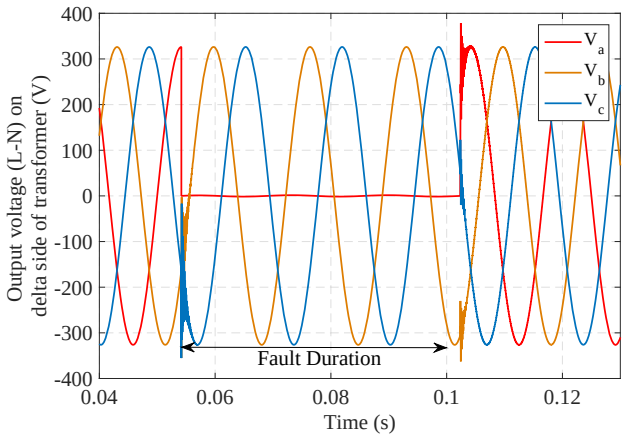


Figure 4.10. Three phase inverter output voltages (line to neutral) under single phase unsymmetrical fault at 0.054167 sec and cleared at 0.1 sec.

For this case study, the pre-trained weights from case study I were used. Single phase to ground fault was simulated in MATLAB/Simulink with the help of three phase fault block of Simulink where V_a was shorted to ground and the parameters were set as

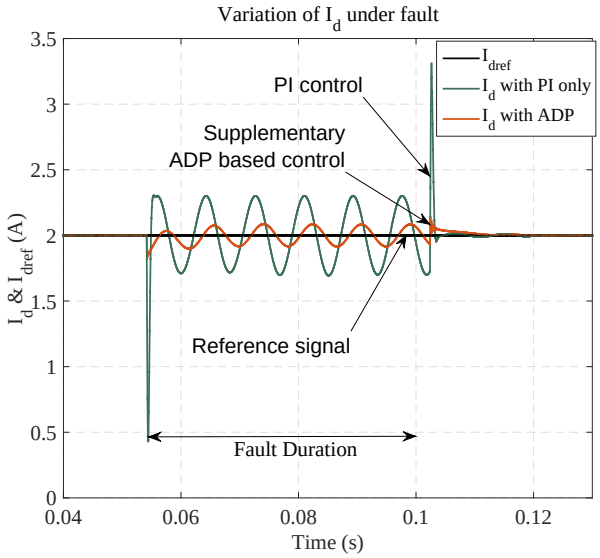


Figure 4.11. Ability of supplementary ADP to track I_{dref} under single phase unsymmetrical fault at 0.054167 sec and fault cleared at 0.1 sec.

mentioned above. For robustness in the phase angle detection during the fault, the PLL in MATLAB was set to have automatic gain control, that enabled the PLL to keep the phase difference between actual signal and the voltage controlled oscillator to zero even during the fault condition. The reference I_d current was set to 2 A. The total simulation time was set to 0.2 sec.

Initially, the system was operated at a steady state condition with d-axis reference current of 2 A. At t=0.054167 sec (worst condition when inverter output voltage, V_a is at its peak), single-phase to ground fault was introduced in the system with fault impedance, Z_g of 0.001Ω. The results in Fig. 4.11 shows that the ADP controller reduces the transient overshoot and steady state error as well for the efficient tracking of I_d current even during unbalanced fault conditions and after fault clearance at 0.1 sec also. Table III shows the overshoot in percentage corresponding to two different control methods for this case study. The transient overshoot during single phase to ground fault at 0.054167 sec with

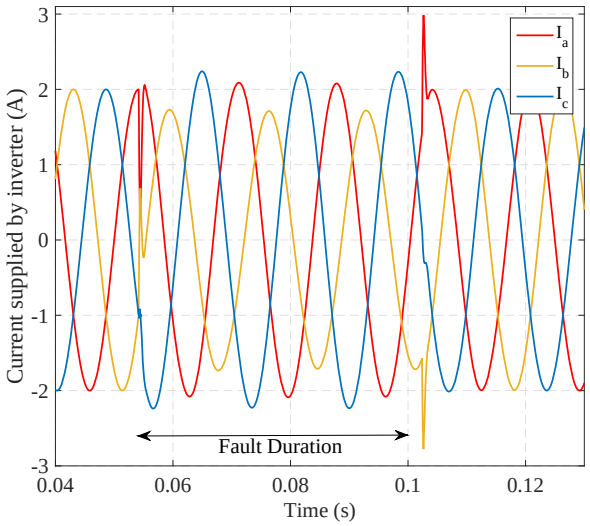


Figure 4.12. Three phase current with PI controller under single phase unsymmetrical fault at 0.054167 sec and fault cleared at 0.1 sec.

Table 4.2. Performance measurement of d-axis current control corresponding to Fig. 4.11

Method / Measurement	Conventional PI Controller [109]	ADP Controller
Maximum overshoot with fault at 0.054167s	78.45%	8.8%
Maximum overshoot with fault cleared at 0.1s	65.44%	6.97%

conventional PI controller [109] is 78.45%. With ADP controller, this overshoot is reduced to 8.8% which can protect the inverter from surge currents. Similar is the condition after fault clearance at 0.1 sec. Comparing Fig. 4.12, Fig. 4.13 and Table III, improvement of the inverter output current by ADP controller is threefold: first the spikes during and after the unbalanced fault is reduced that can protect the inverter from surge currents, second the distortions in the current are minimized and third is that the supplementary ADP controller enables the inverter to supply balanced current to the grid even under unbalanced fault conditions.

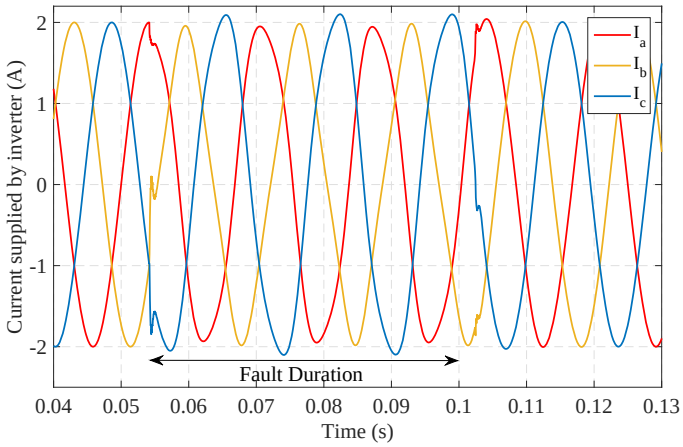


Figure 4.13. Three phase current with ADP controller under single phase unsymmetrical fault at 0.054167 sec and fault cleared at 0.1 sec.

4.4 Summary

In this chapter, a supplementary ADP controller for d-axis current control is proposed so that I_d current effectively tracks reference current. It plays an important role for the effective application of VSM, not only under sudden step change condition, but also under the fault in three-phase system such as unbalanced single line to ground fault. With the advantage of NN weights updating rules, the ADP controller adopted its key parameters when the system is under different faults. More robust results were obtained by ADP controller in comparison with the conventional PI controller. This VSM system with adaptive control has long-term benefit to micro-grids and data center applications.

CHAPTER 5 Smart Grid Application 2: Harmonic Reduction Using Shunt Active Filter and Online Learning based Control

The project in this chapter is a joint work with Dr. Reinaldo Tonkoski, Ujjwol Tamrakar and Dipesh Shrestha who kindly provided benchmark of this work. Electrical energy is expected to contribute to about 60% of all world energy consumption by 2040 [114]. Power electronics based equipment in data centers and in commercial and industrial applications as shown in Fig. 5.1 are with non-linear nature. Hence, they draw harmonic currents from the power grid and can cause power quality issues. Some of the major issues related to current harmonics include vibration in motors, generator burnouts, and computer network failures to name a few [115, 116]. Among the different approaches, shunt active filters (SAF) have become a particularly popular solution for harmonic reduction, owing to the recent advancements in power electronic switches and their guaranteed performance over a wide range of dynamic operating conditions unlike passive filters [117, 118].

The total harmonic distortion (THD) of a current signal is a measurement of the harmonic distortion present and is defined as the summation of all the harmonic components of the current waveform compared against the fundamental component of the current wave. The IEEE Std. 519 recommendations for harmonic control in power systems states that the THD of the source current should be less than 5% [119]. The standard also has stringent requirements on individual harmonic components. The basic concept behind a SAF is to compensate the higher order harmonic currents and the reactive power demand of the load through a power electronics based current controlled

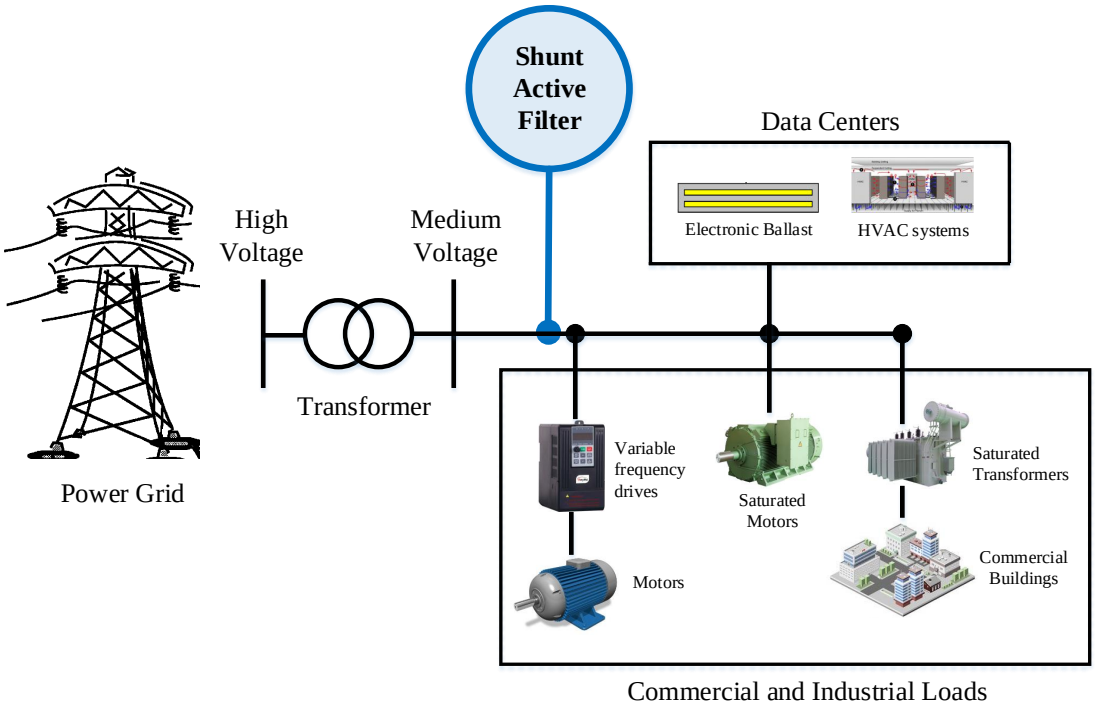


Figure 5.1. Microgrid with non-linear loads and shunt active filter connected to reduce harmonics.

voltage source power inverter (CCVSI), hereinafter referred to as power inverter. Thus, the source only needs to provide the fundamental component of the load current [120]. The control structure of a SAF has two control loops: (i) an outer loop that estimates the current that the inverter needs to inject/absorb to compensate the load harmonics and the reactive power, and (ii) an inner loop responsible for tracking the generated reference current. The outer current loop is typically implemented using either the instantaneous reactive power theory [121], or the synchronous reference frame theory [122]. The inner current control loop is traditionally implemented using a proportional-integral (PI) control based approach. However, the dynamics of the PI controller are inadequate because of the limited bandwidth of the controller. As a result, the SAF may not give the required performance when the loads have a high degree of non-linearity [123].

This has led to the development of a number of machine learning based adaptations to improve the performance of the traditional control in power inverters. It also motivated us to design an online learning based controller to improve the performance of the power inverter in terms of the THD. Artificial neural network (NN) techniques and classical back-propagation algorithms have been used to improve the control performance of the SAF in [115, 120, 124]. However, the implemented algorithm lacks online adaptation and the performance under varying load conditions has not been quantified. In [125], NNs were used in the outer loop to improve the estimation of compensating currents. Although the THD was reduced within the IEEE Std. 519 limits, the inner control structure still used a hysteresis based current control approach which has its own disadvantages related to high switching frequency and difficulty in the filter design. In [99], recurrent neural network (RNN) training with the Levenberg-Marquardt algorithm is used for optimal control of a grid-connected inverter, but this does not involve online training and requires a well-trained RNN controller. Other optimal control techniques like the model predictive control with selective harmonic elimination is proposed in [126] for multilevel power converters which resulted in a fast dynamic response. However, these techniques are highly dependent on the accuracy of the system model and are prone to limited transient and steady state behavior due to parameter uncertainties [127].

All of the aforementioned literature indicates the need for developing an online learning approach to improve the dynamic performance of the inner current control loop, hence improving the harmonic compensating capability of power inverters. An adaptive dynamic programming (ADP) controller has been proposed to address these issues in recent power and control communities. It consists of two NNs based architecture

consisting of an action network and a critic network. The critic network is designed to evaluate the online learning based controller, while the action network produces an online learning based control signal to improve the control performance [17, 18, 23–25, 62, 128, 129]. The ADP controller has shown promising results on a number of power system control examples [66, 81, 103–107, 130–134]. For example, the concept of the supplementary ADP control has been used to enhance power system stability in [106, 107, 135, 136]. In [62, 96, 105], a similar control approach is incorporated into the generators' frequency control loop to improve the frequency response of the system and smooth out the variations due to large scale wind, photovoltaics (PVs) and/or electric vehicles (EVs) integration.

Yet, only a few papers have investigated the power electronics pulse width modulation (PWM) current control through the ADP based controller concerning the difficulty involved as mentioned in [97]. Literature [94, 95, 97, 98] proposed and validated the vector control of a grid-connected rectifier/inverter using an artificial neural network and back-propagation through time weights updating rule. However, the system was not implemented for harmonic reduction applications and the current controller lacks the online learning capability. In this chapter, an integration of the online learning ADP control approach is proposed for the power electronic inverter to improve the harmonic compensating capability and the transient performance. The model-free ADP control does not rely on an accurate mathematical model of the system or the pole-zero placement design as in traditional controllers. Thus, the mathematical model descriptions of the benchmark system are not required. The ADP controller is a purely data-driven control approach which observes the system states and outputs the control actions over time. In

addition, the ADP controller has learning and adaptation capabilities. The key parameters in the ADP controllers will be adjusted online for the disturbances and changing of loads.

The major contributions of the research in this chapter are summarized as follows: 1) A multiple-input multiple-output (MIMO) online learning control system is developed based on the ADP design. A series of time-delayed current error signals are designed as input for the ADP controller, which outputs compensating control actions (one is for d-axis and the other one is for q-axis) for the current controller in the power inverter. In addition, an appropriate reinforcement signal has been designed with the delayed tracking errors in the power inverter. Based on this, the ADP controller will generate compensating control actions, which guarantees its success. 2) This proposed approach has been applied in two challenging case studies. First, the proposed system was tested for a three-phase full wave bridge rectifier with a resistive load (non-linear load). The current drawn by such a load shows a high degree of non-linearity. The ADP controller improved the bandwidth of the inner current control loop leading to efficient tracking of the currents and also an average reduction by 18.41% in THD. Next, the proposed system was subjected to a sudden step change in the load. Furthermore, the system was tested under different load conditions. The system obtained a better transient and steady state performance (e.g., faster response, lower overshoot, and efficient tracking) compared to a traditional PI controller based system.

5.1 Benchmark and System Configuration

A simplified diagram of the SAF configuration is shown in Fig. 5.2. The non-linear load is connected to the three-phase AC source. The non-linear load considered in this

chapter is a three-phase diode bridge rectifier, followed by a resistance, R_L. In order to supply the harmonics and the reactive power consumed by this non-linear load, the SAF is connected in parallel to the system. The SAF thus provides the harmonic current demand of the load and as a consequence only sinusoidal currents with an almost unity factor would be delivered from the source.

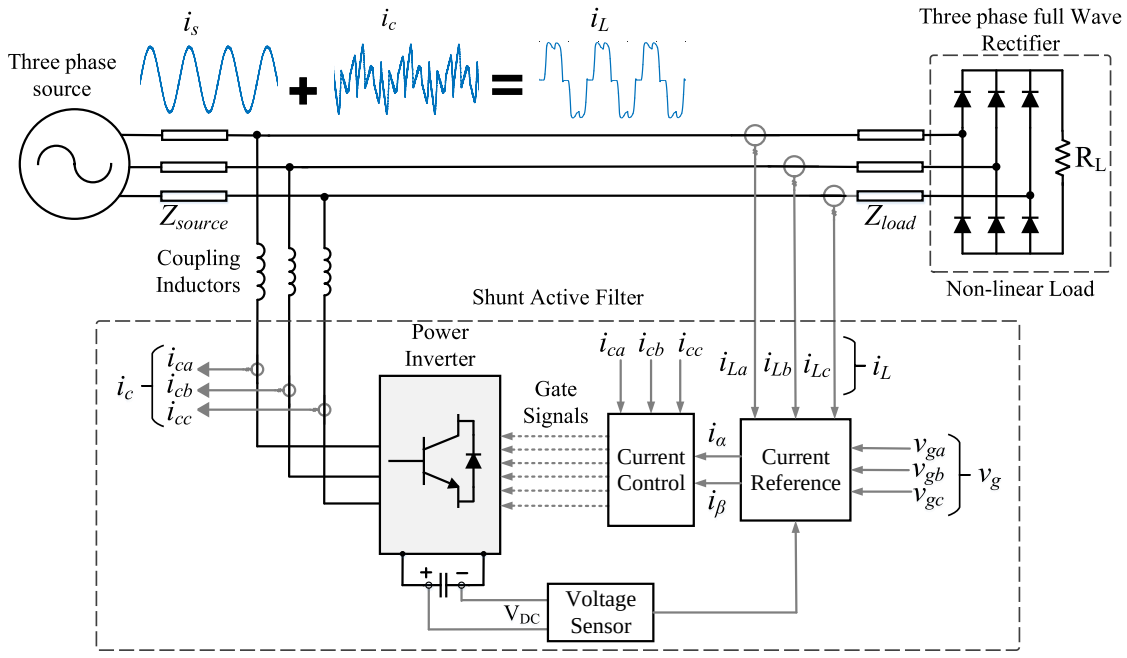


Figure 5.2. Schematic diagram of a shunt active filter connected to source and non-linear load for compensation of harmonics.

The SAF generates the compensation current, i_C. The sum of the source current, i_S and i_C is the non-linear load current, i_L. The coupling inductor is used to suppress the high-frequency currents originating from the switching of power inverter. There are two main control loops associated with the SAF control algorithm: an inner current control loop and an outer harmonic current estimation loop. The inner current control loop tracks the reference current in the $d-q$ frame using the PI controller. The outer control loop measures the load current and the source voltage and generates the reference active and

reactive power to be compensated by the SAF using instantaneous $p-q$ theory as shown in Fig. 5.3. First, the instantaneous three-phase source voltages (v_{ga}, v_{gb} and v_{gc}) and the three-phase load currents (i_{La}, i_{Lb} and i_{Lc}) are measured and transformed to α-β-0 coordinates (i_{L0}, $i_{L\alpha}$, $i_{L\beta}$, v_{g0}, $v_{g\alpha}$ and $v_{g\beta}$ as shown in Fig. 5.3) using the Clarke's transformation as follows:

$$\begin{bmatrix} v_{g0} \\ v_{g\alpha} \\ v_{g\beta} \end{bmatrix} = \sqrt{\frac{2}{3}} \begin{bmatrix} \frac{1}{\sqrt{2}} & \frac{1}{\sqrt{2}} & \frac{1}{\sqrt{2}} \\ 1 & -\frac{1}{2} & -\frac{1}{2} \\ 0 & \frac{\sqrt{3}}{2} & -\frac{\sqrt{3}}{2} \end{bmatrix} \begin{bmatrix} v_{ga} \\ v_{gb} \\ v_{gc} \end{bmatrix} \tag{5.1}$$

$$\begin{bmatrix} i_{L0} \\ i_{L\alpha} \\ i_{L\beta} \end{bmatrix} = \sqrt{\frac{2}{3}} \begin{bmatrix} \frac{1}{\sqrt{2}} & \frac{1}{\sqrt{2}} & \frac{1}{\sqrt{2}} \\ 1 & -\frac{1}{2} & -\frac{1}{2} \\ 0 & \frac{\sqrt{3}}{2} & -\frac{\sqrt{3}}{2} \end{bmatrix} \begin{bmatrix} i_{La} \\ i_{Lb} \\ i_{Lc} \end{bmatrix}. \tag{5.2}$$

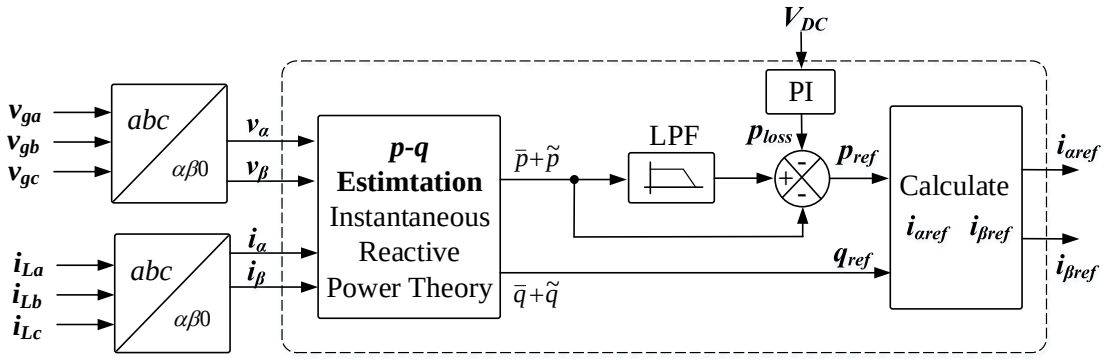


Figure 5.3. Current reference generation ($i_{\alpha ref}$ and $i_{\beta ref}$).

Next, these currents and voltages in α-β-0 coordinate are used to calculate the

instantaneous active (p) and reactive power (q) consumed by the load as follows:

$$p = v_{g\alpha} i_{L\alpha} + v_{g\beta} i_{L\beta} = \bar{p} + \tilde{p} \tag{5.3}$$

$$q = v_{g\alpha} i_{L\beta} - v_{g\beta} i_{L\alpha} = \bar{q} + \tilde{q} \tag{5.4}$$

where, $\bar{p}$ and $\tilde{p}$ are the low frequency and the high frequency components of instantaneous active power, and $\bar{q}$ and $\tilde{q}$ are the low frequency and the high frequency components of instantaneous reactive power. The instantaneous active power consumed by the load ($\bar{p} + \tilde{p}$) is passed through a low pass filter (LPF, butterworth type) to filter out the high frequency harmonics component ($\tilde{p}$). Next, the output of the LPF is deducted from the original instantaneous active power to give just the high frequency component ($\tilde{p}$). This combination is essentially a high pass filter (HPF) that extracts only the high frequency components of the load active power. This is the reference compensating active power (p_{ref}) for the SAF. In order to compensate all the reactive power demanded by the load and to maintain the unity power factor in the source current, all the calculated instantaneous reactive power becomes the reference compensating reactive power (q_{ref}) for the SAF. There is one extra control loop which compensates the active power loss in the capacitor to maintain constant DC voltage across it. The DC voltage across the capacitor is compared with the standard voltage (400 V): the error is then fed to a PI controller which generates the compensating active power (p_{loss}) to maintain the constant voltage across the DC capacitor. For the PI controller design of this extra control loop, the inverter is considered to be a constant current source for the DC side of the voltage

controller. Thus, the transfer function of the plant for the DC side controller is as follows:

$$\frac{V_{DC}}{p_{loss}} = \frac{\frac{1}{CV_{CO}}}{s + \frac{2}{RC}} \tag{5.5}$$

where, p_{loss} is the compensating active power loss in the capacitor, V_{DC} is the DC bus voltage across the capacitor, C is the rated capacitance, V_{C0} is the reference capacitor voltage, and R is the resistance parallel to the capacitor.

Next, the reference active power loss for the DC voltage controller is added to the reference active power calculated earlier from the HPF. Thus, the reference instantaneous active and reactive power are calculated using:

$$\begin{bmatrix} i_{\alpha ref} \\ i_{\beta ref} \end{bmatrix} = \frac{1}{v_\alpha^2 + v_\beta^2} \begin{bmatrix} v_{g\alpha} & -v_{g\beta} \\ v_{g\beta} & v_{g\alpha} \end{bmatrix} \begin{bmatrix} \tilde{p} - p_{loss} \\ q \end{bmatrix} \tag{5.6}$$

where, $i_{\alpha ref}$ and $i_{\beta ref}$ are the reference compensating currents in the α-β-0 coordinate system. Since the PI controller cannot track sinusoidal currents, the reference current in the α-β-0 coordinate is converted into $d-q$ coordinates using the following relation:

$$\begin{bmatrix} I_{dref} \\ I_{qref} \end{bmatrix} = \begin{bmatrix} cos(\omega t) & sin(\omega t) \\ -sin(\omega t) & cos(\omega t) \end{bmatrix} \begin{bmatrix} i_{\alpha ref} \\ i_{\beta ref} \end{bmatrix} \tag{5.7}$$

where, ωt is the synchronous reference frame angle, and I_{dref} and I_{qref} are reference currents in the $d-q$ coordinate system. Finally, the reference current in $d-q$ coordinates is passed to a standard tuned PI controller designed with a switching frequency of 20 kHz, a cut-off frequency of 4 kHz and a phase margin of 45° (tuned by the methods described

in [68, 111]) which generates the modulating signal for the PWM generator of the inverter. The cut-off frequency is kept relatively higher in this case to increase the PI controller's bandwidth.

The benchmark simulation model was developed and tested in MATLAB/Simulink. A three-phase three-wire balanced system is considered in this chapter. A three-phase bridge diode rectifier with a DC load of 700 W (base case) is used as the non-linear load. The load was simulated using six diodes configured in three-phase bridge configuration as illustrated in Fig. 5.2. The output of the diode-bridge was then connected to resistor R_L. The value of R_L was obtained using the simple equation:

$$P = \frac{V_r^2}{R_L} \tag{5.8}$$

where V_r is the output voltage of the rectifier which in this case is 272 V. The power inverter is rated at 1 kW. A capacitor rated at 400 V DC is used as the energy storage device. The switching frequency of the PWM inverter is 20 kHz. The grid side impedance is set as: $Z_{source} = (0.01 + j0.00038)\Omega$ and load side impedance is set as: $Z_{load} = (0.01 + j3.8)\Omega$. The parameters used for the simulation of the benchmark are given in Table 5.1.

In this chapter, THD is used as the performance measurement to evaluate the success of the control approaches. The THD is computed as:

$$THD = \frac{\sqrt{I_2^2 + I_3^2 + I_4^2 + ... + I_N^2}}{I_1} \tag{5.9}$$

Table 5.1. Benchmark system parameters

	Parameter	Value
1.	Inductance of coupling inductor	10 mH
2.	Internal resistance of coupling inductor	0.1 Ω
3.	Resistance of grid	0.01 Ω
4.	Inductance of grid	1 μH
5.	Load side resistance	0.01 Ω
6.	Load side inductance	10 mH
7.	Capacitance of DC bus capacitor	450 μF

where $I_2, I_3, I_4, ..., I_N$ are the root-mean-square (RMS) value of the harmonics 2,3,4,...,N respectively and I_1 is the RMS value of the fundamental source current. Here, the N^{th} harmonic order corresponds to the Nyquist frequency. The Nyquist frequency is half the sampling frequency of the selected signal. In this chapter, the sampling frequency is 1 MHz and the switching frequency is 20 KHz. Thus, the Nyquist criterion is satisfied.

5.2 Proposed Controller Design for Benchmark System

The basic idea in an adaptive-critic design is to adapt the weights of the critic network to approximate the optimal cost function, $J^*(X(t))$, satisfying the modified Bellman principle of optimality [42], given by:

$$J^*(X(t)) = \min_{u(t)} \{J^*(X(t+1)) + r(X(t)) - U_c\} \tag{5.10}$$

The optimal online learning based controller can be written as:

$$u^*(X(t)) = arg \min_{u(t)} J^*(X(t)) \tag{5.11}$$

where $X(t)$ is the input state vector, $r(X(t))$ is the immediate cost incurred by $u(t)$ at time t, and U_c is ultimate desired objective which is to be achieved by the cost function. This equation cannot be analytically solved in general, thus the problem of optimal current control needs to be solved iteratively and approximately by using ADP [137–140]. In ADP, the action network generates the optimal control action iteratively and the critic network evaluates the performance of action network by approximating J close to the optimal solution. J is also the output of the critic network. The connection detail for the inner current control loop with proposed online based ADP controller is shown in Fig. 5.4 with the signal flow from left to right. The errors in d-axis and q-axis feedback currents (I_d and I_q) are fed to both the PI and ADP controllers whose output combination produces the appropriate modulating signal for the PWM inverter. The gate signals for the inverter are generated using this modulating signal after considering the cross-coupling terms ωLI_q and ωLI_d, and adding feed-forward terms V_d and V_q. This signal is then transformed from $d-q$ to $a-b-c$ coordinates using the inverse Park's transformation. A LPF is added before PWM generation to avoid potential high frequency noise in the modulating signal by the ADP. Here, the online learning based control action produced by ADP is defined as: $u(t)$=[u_d(t), $u_q(t)$], where $u_d(t)$ and $u_q(t)$ are supplementary control actions for the outputs of d-axis and q-axis PI controllers respectively. There are two paths to tune the parameters of the two types of networks in ADP which will be discussed below.

The critic network is a three-layer neural network with 12 hidden neurons. The architecture of the critic network is 8-12-1 as shown in Fig. 5.5. The inputs to the critic network are the measured system state vector, $X(t)$, and action network output, $u(t)$. Here, $J(t)$ is the output of the critic element and the J function approximates the discounted total

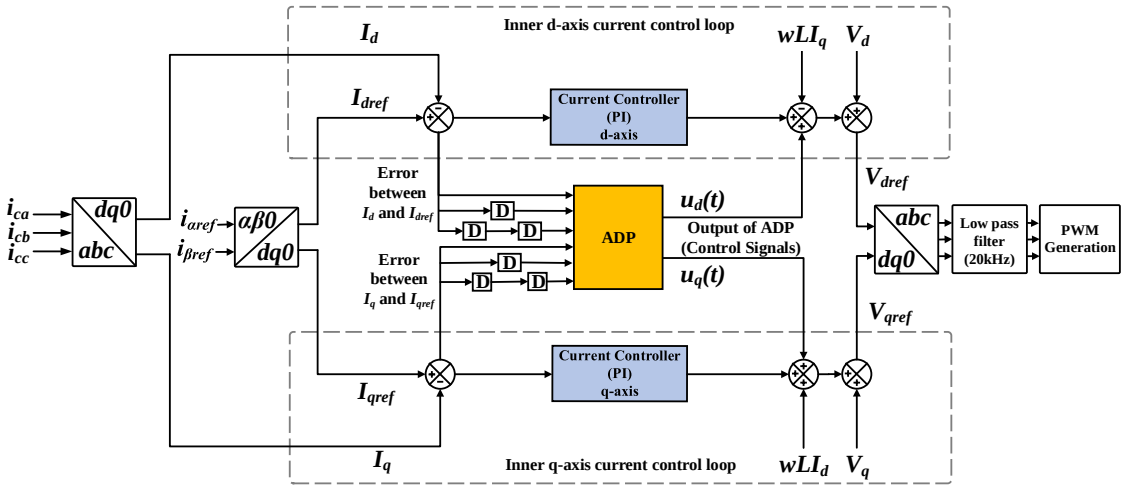


Figure 5.4. Overall diagram of online learning based ADP controller for the inner current control loop in the power inverter (Fig. 5.2).

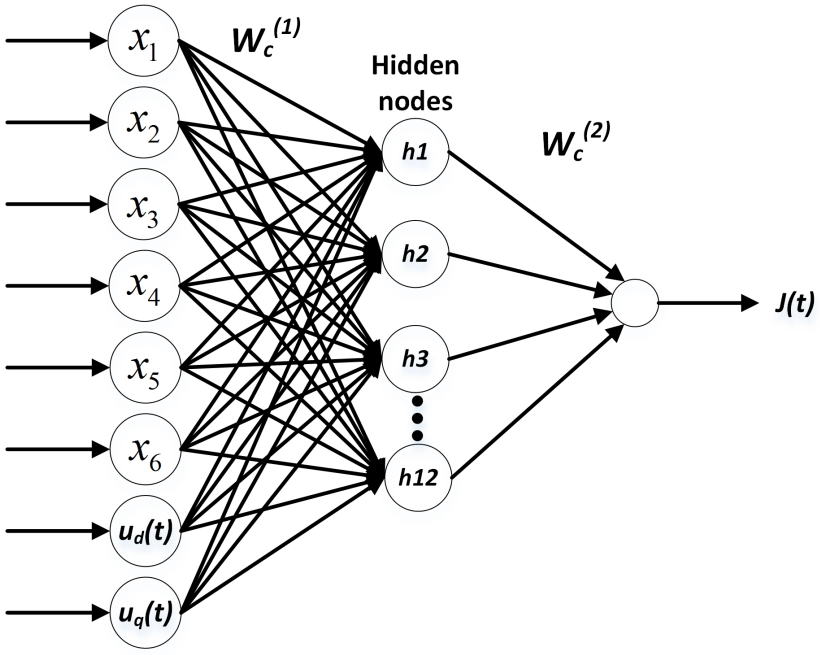


Figure 5.5. Critic neural network with 8 inputs, 12 hidden neurons, and 1 output neuron.

reward to go.

The prediction error for the critic network is given by:

$$e_c(t) = \alpha J(t) - [J(t-1) - r(t)] \tag{5.12}$$

The reinforcement signal, $r(t)$ for the critic network is defined as follows:

$$r(t) = -c(a_1 x_1^2 + a_2 x_2^2 + a_3 x_3^2 + a_4 x_4^2 + a_5 x_5^2 + a_6 x_6^2) \tag{5.13}$$

where c, a_1, a_2, a_3, a_4, a_5 and a_6 are the coefficients of this quadratic equation.

x_1=error signal between I_d and I_{dref};

x_2=one-time step delayed error signal for I_d;

x_3=two-time step delayed error signal for I_d;

x_4=error signal between I_q and I_{qref};

x_5=one-time step delayed error signal for I_q;

x_6=two-time step delayed error signal for I_q;

$X(t)$=[x_1, x_2, x_3, x_4, x_5, x_6].

The above delayed error signals (1 time-step and 2 time-step delayed signals) are fed into the ADP controller in order for it to work properly. This would ensure that the high magnitude of compensating control actions are produced during the transients, and less magnitude of compensating control actions during the steady state conditions.

The action network has a similar multi-layer perceptron neural network architecture as the critic network. However, input neurons and output neuron numbers are different.

The inputs to the action network are the measured system state vector, $X(t)$, and the output of the action network is the online learning based control signal, $u(t)$. The principle in adapting the action network is to indirectly back-propagate the error between the desired ultimate objective, denoted by U_c, and the approximate J function from the critic network. Since "0" is defined as the reinforcement signal for "success," U_c is set to "0" in the design paradigm. In the action network, the state measurements are used as inputs to create a control as the output of the network. In turn, the action network can be implemented by either a linear or a non-linear network, depending on the complexity of the problem. The action and critic networks weights can be trained by algorithm as shown in chapter 4, section 4.2.

5.2.1 Training Procedures for ADP Controller

The objective of the ADP controller is to provide an optimal control signal that exhibits fast dynamics in tracking I_d and I_q current references. The weights of the two networks are updated at most N_a and N_c times for action, and critic networks respectively within each time step. It will also be stopped once the internal training error threshold T_a and T_c have been met, using the gradient descent algorithm described in Section III. The typical learning process of the ADP controller includes two trials as described in [104]. In the first trial, NN is initialized with random weights. The simulation is repeated until no further improvement in THD has been obtained. At the end of this process, the ADP would learn a considerable amount of information about the system and state-action pairs. This is called offline learning. In the second trial, or online learning, fully trained weights are used. The online training procedure for the ADP controller is shown in Algorithm 1.

Algorithm 1 Online learning process of ADP controller

1: Initialize ADP with *i*=0, online learning based control action, $u^{(0)}(t) = [0,0]$, use weights from offline training, and $J^{(0)} = 0$.
2: Apply $u^{(i)}(t)$ in addition to the output of PI controller in power inverter and collect data for state, $X(t)$ from 3 most recent samples for ADP controller.
3: Obtain the cost function $J^{(0)}(t)$ by using Eq. (4.15).
4: Obtain the online learning based controller function, $u^{(i)}(t)$ by using Eq. (4.8).
5: Update critic and action NN weights using Eq. (4.16) and (4.9) respectively.
6: $i = i + 1$. Repeat from Step 2 to 5 with until end of simulation.
7: Observe THD and repeat from Step 1 to 6 until no further improvement.
8: Save NN weights.

The weights evolution of the ADP controller is saved for the run with the best performance in terms of THD. Fig. 5.6 shows the trajectories for a typical training process. Fig. 5.6(a) shows the weight evolution of the action network from 6 input nodes to 1 hidden node. Similarly, Fig. 5.6(b) shows the weight evolution of the action network from 12 hidden nodes to 1 output node. Here, the power inverter and ADP are connected at 0.11s at which there is a major adjustment of the weights because of transients in the system. In addition, Fig. 5.6(c) shows the convergence of the $J(t)$ function, and Fig. 5.6(d) shows the convergence of the reinforcement signal, $r(t)$. The convergence of both w_{a1} and w_{a2} indicates the convergence of the learning process. The weights are saved at the end of the simulation process and fine-tuned until there is no further improvement in the THD.

5.2.2 Stability Discussion of ADP Controller

There are several major approaches to analyze the stability of ADP controller. A detailed Lyapunov stability analysis of the ADP based controller is presented in [70] to support the ADP structure from a theoretical point of view. The authors demonstrated that the auxiliary error and the error in the weights estimates are uniformly ultimately bounded (UUB) using the Lyapunov stability construct. In [71], the proportional-integral-derivative

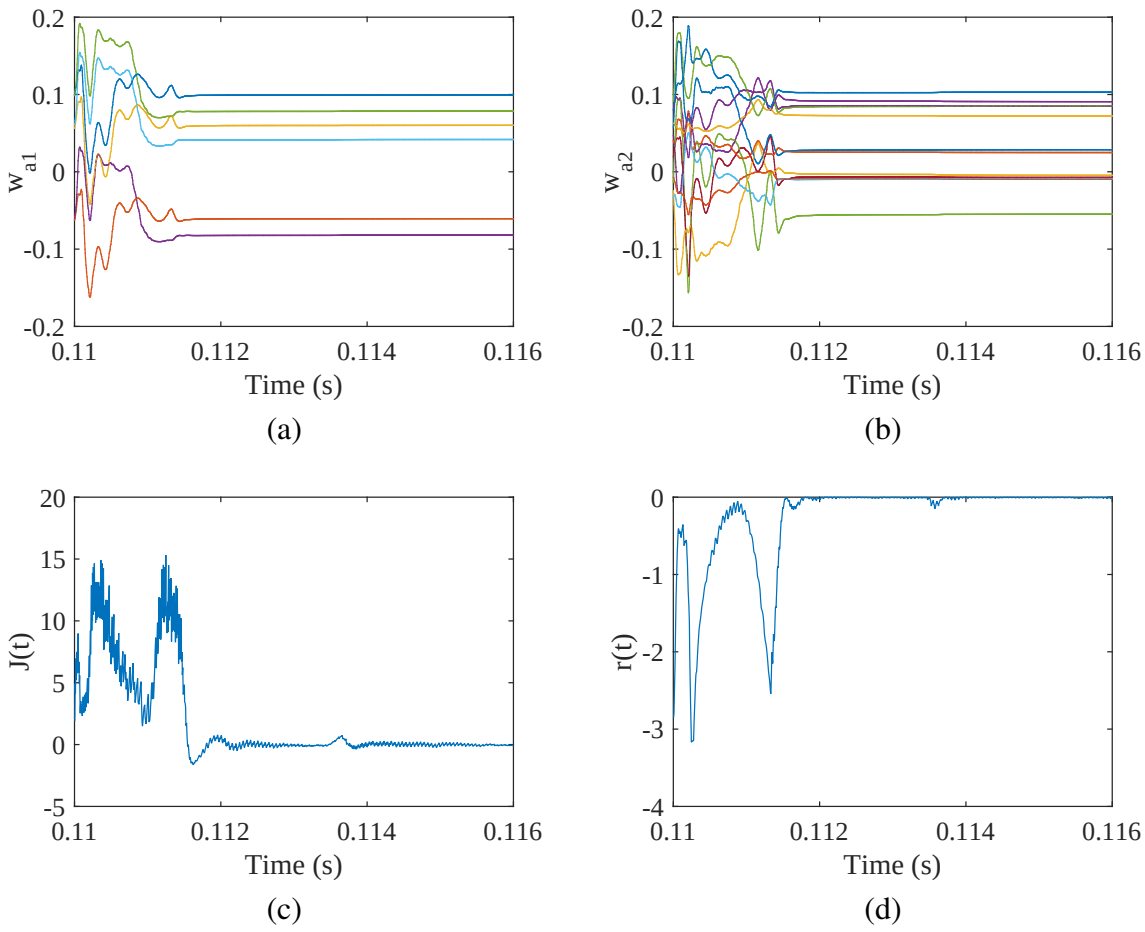


Figure 5.6. The training process of ADP controller: (a) weights trajectories from 6 inputs to 1 hidden node in action network, (b) weights trajectories from 12 hidden to 1 output node in action network, (c) trajectory of output of critic network, $J(t)$, and (d) reinforcement signal, $r(t)$ during the training process.

(PID) control rule is incorporated into neural networks (NNs) and new results of UUB are provided using a Lyapunov stability construct. The monotonic convergence of optimality is discussed for the goal representation ADP control design in [72], and theoretical proof of convergence is given in terms of both the internal reinforcement signal and the performance index. In [73, 74, 141, 142], stability of the ADP controller is presented, where the authors demonstrated the theoretical analysis that the estimation errors of NN weights are UUB by the Lyapunov stability construct. The similar technique can be

followed to conduct the analysis. The similar algorithm is followed as in [73], and set the parameters accordingly. From Corollary 4.4 in [73], the error in the NN weights estimate are UUB, provided that the conditions are met. For this case, the discount factor is set as, α=0.95, number of hidden nodes for both action and critic network=12, and learning rates, l_a=l_c=0.01. With these parameters, all criteria as described by Corollary 4.4 in [73] are met. Thus, estimation errors of NN weights are UUB for the ADP controller designed in this chapter. Experimentally, it can be observed from Fig. 5.6 that the weights and parameters of the ADP controller are quickly converged and bounded after the system transients.

5.3 Simulation Results

The performance of the learning control approach using ADP was accessed by testing the integrated controller and the benchmark with the ADP embedded in the S-function block in Simulink. Two case studies were performed: the first case study was focused on the improvement of current harmonics by connecting the power inverter with the traditional PI controller in an inner current control loop, and the subsequent performance improvement by the ADP. The second case study was a performance comparison between the traditional PI and the ADP based control approach during a sudden step change in load as well as under different loading conditions. Note that, “without power inverter” in the following subsections refers to the system without the SAF connected in the system. “PI” is used for the traditional PI controller in inner loop of the power inverter and “PI+ADP” is used for the integrated PI and ADP controller in the inner loop of the power inverter.

5.3.1 Parameters Setup for ADP Controller

The following parameters were adjusted for training of the ADP: Inputs were normalized in the range of [-1,1] after dividing the d-axis error by 3 and the q-axis error by 2 since the input current was assumed to vary in the range of 0 to 3 A for a 1 kW inverter. Outputs were amplified by a factor of 100 for the d-axis and 66.67 for the q-axis since output of the PI controller varies from 0 to 400. Here, 25% adjustment was assumed for the online learning based ADP controller. The coefficients of Eq. (5.13) were set as follows: c=1, a_1=0.4, a_2=0.2 and a_3=0.04, a_4=0.4, a_5=0.2 and a_6=0.04. For both action and critic network, weights were initialized in the range of [-0.1,0.1].

The following additional parameters are set in ADP.

N_c = internal cycle of the critic network;

N_a = internal cycle of the action network;

T_a = internal training error threshold for the action network;

T_c = internal training error threshold for the critic network;

l_a = learning rate of the action network=0.01 for training and $l_a(f)$=0.001;

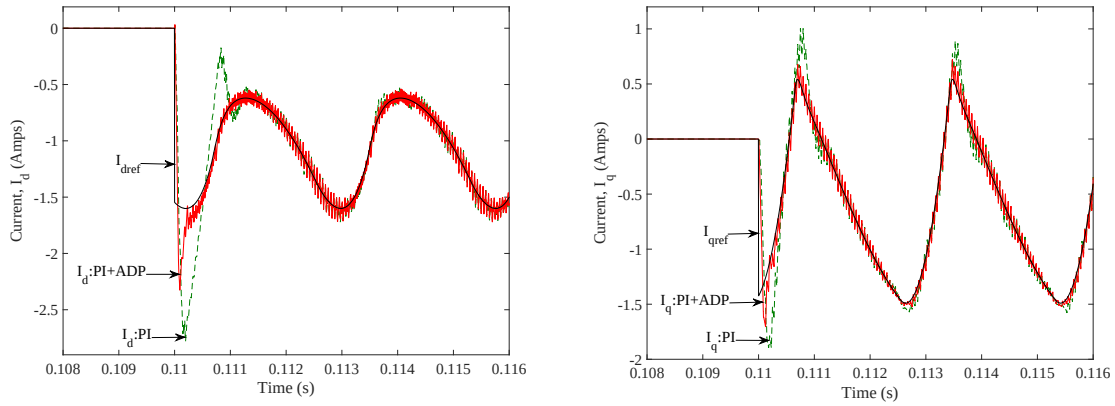
l_c = learning rate of the critic network=0.01 for training and $l_c(f)$=0.001.

For the training process, after the above parameters were set in MATLAB inside the S-function block, the ADP controller was connected as shown in Fig. 5.4 to the power inverter shown in Fig. 5.2. Then, these trained weights were used for improving the transient and steady state operation for both of cases. If major system parameters changes, normalization of input and amplification of output might change or ADP might need training again.

5.3.2 Case Study I: Non-linear Load

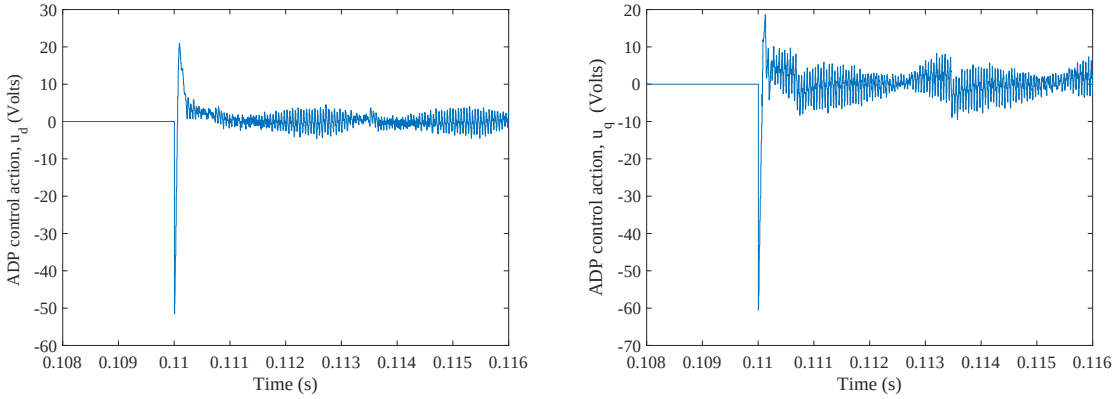
A three-phase diode bridge rectifier was used as the non-linear load (described in Section II) as shown in Fig. 5.2 for case study I. The 700 W load was simulated using 105 Ω load resistance from Eq. (5.8). The three phase AC source with a frequency of 60 Hz and line to line root-mean-square (RMS) voltage of 208 V was connected to this load. Then, THD was calculated by using fast fourier transform (FFT) analysis tool of 'powergui' block in Simulink for 12 cycles of source current waveform. Here, Eq. (5.9) with a sampling time of $1\mu s$ was used for THD calculation. The current (i_S) supplied by the source was distorted and had a high THD of 25.21% before connecting the power inverter. Fig. 5.7 shows the performance comparison between the PI controller and the PI+ADP controller with a tracking curve for the d-axis current, I_d, and q-axis current, I_q. These are denoted by "I_d:PI" and "I_d:PI+ADP" in Fig. 5.7(a), and "I_q:PI" and "I_q:PI+ADP" in Fig. 5.7(b) respectively. "I_{dref}" and "I_{qref}" represent the reference current to be tracked. When the power inverter was connected at 0.11s, the initial charging of the capacitor caused transient behavior for a few cycles as shown in Fig. 5.7. In this case with the traditional PI controller, the THD of the source current was 3.69%. It can also seen that proposed online learning based controller has a lower overshoot and more efficient tracking than the PI controller which reduced the THD in the source current to 2.70%.

The reason why the ADP gave better results than the traditional PI controllers can be explained with the help of Fig. 5.7(a) and Fig. 5.7(b). The ADP controller generated adaptive control actions for the d-axis and q-axis currents, represented by u_d and u_q respectively in Fig. 5.8(a) and Fig. 5.8(b). The transient dips in the d-axis and q-axis



(a) (b)

Figure 5.7. Performance comparison between two control techniques: PI and PI+ADP with tracking curve for (a) direct axis current, I_d, and (b) quadrature axis current, I_q.



(a) (b)

Figure 5.8. Control actions generated by ADP for (a) direct axis current, I_d, and (b) quadrature axis current, I_q.

currents were reduced by the compensating control actions generated by the ADP controller from 0.11s to 0.111s as shown in Fig. 5.8(a) and Fig. 5.8(b). The tracking errors for I_q are clearly visible at 0.1108s and 0.1135s in Fig. 5.7(b). The ADP controller produced appropriate compensating control actions as shown in Fig. 5.8(b) to reduce these tracking errors. These control actions were added to the output of the PI controller so that the appropriate gate signals can be produced for the power inverter. This reduced the error

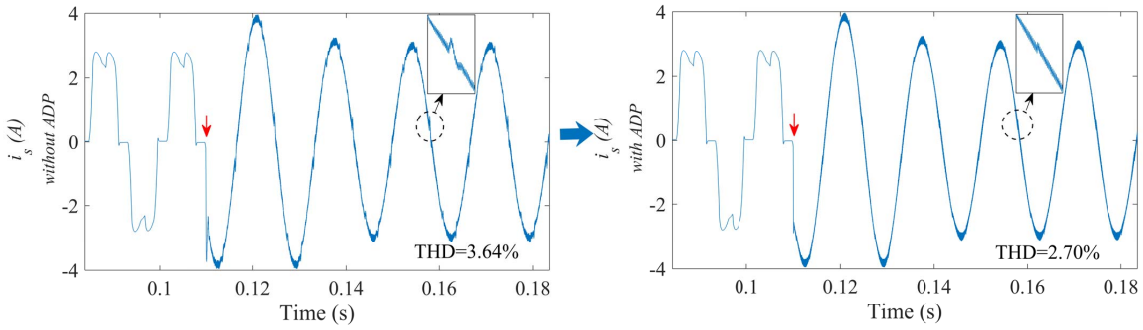


Figure 5.9. Dynamic experiment response of the power inverter connected system. First figure shows source current with the power inverter (PI controller implemented) connected at 0.11s. Second figure shows improved source current with the power inverter (PI+ADP controller implemented) connected at 0.11s. All these currents are for the first A-phase.

between I_q and I_{qref}. This error signal was also set as one of the components for the reinforcement signal (represented by x_4 in Eq. (5.13)), and this helped the ADP controller to generate proper control actions. Similar reasons apply for I_d as well.

The THD of 3.69% with the PI only was due to the non-efficient tracking of reference I_{dref} and I_{qref} currents as shown in Fig. 5.7. Here, 0.11s time was chosen for connecting SAF to observe the best performance of proposed controller since this time had high transients with PI controller alone. With the integration of the ADP controller, the tracking became efficient. Thus, the source current became sinusoidal with less harmonic components as shown in second figure of Fig. 5.9 with THD of 2.70%. Thus from Eq. (5.9), THD was improved with ADP controller in comparison to traditional PI controllers. This was the reason why ADP gave improved total harmonics in terms of THD during the step response and changing of non-linear loads.

5.3.3 Case Study II: Different Loading Conditions

The same trained weights used for ADP from the case study I were used for case study II as well. The non-linear load which was for case study I was reduced from 705 W to 385 W at 0.35s to observe the transient performance of the proposed controller. This was conducted by changing the value of the output resistance R_L from 105 Ω to 192 Ω at 0.35s. The ADP controller generated the similar compensating control actions as Fig. 5.8(a) and Fig. 5.8(b) during the step change in load. Thus, the online learning based ADP controller showed consistent performance and improved transients as well, which can be seen in Fig. 5.9. Figs. 5.9(a) and 5.9(b) show the zoomed-in plots for the comparison of the transient performance of active and reactive power supplied by the source with the power inverter connected at 0.11s. This type of oscillation in the active and reactive power is normal in the case of power electronic circuits. However, the adaptive control approach improved the transients in power drawn from the source. The proposed system with an online learning based ADP controller performed well during such conditions. Because of the efficient tracking of I_d and I_q, the fluctuations and spikes in the reactive power for the power inverter connected system were reduced by the integration of the proposed online learning based ADP control system which can be seen in Fig. 5.9(b) after 0.11s.

The proposed online learning based controller enabled the most efficient tracking of both I_d and I_q current in comparison to the PI controller alone. Thus, the fluctuations in the active and reactive power for the PI controller were reduced by the integration of ADP. The system performed well even during a sudden load change from 705 W to 385 W at

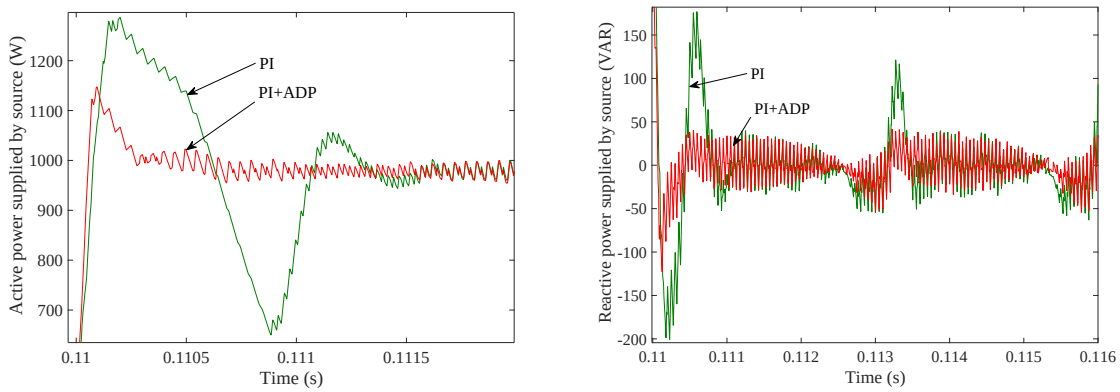


Figure 5.9. Comparative study of PI and PI+ADP controller for (a) Active power transients, and (b) Reactive power transients because of the power inverter connected at 0.11s. These figures are zoomed-in to show superior performance of the ADP controller.

0.35s. THD of the system was reduced for 385 W load from 5.37% (PI controller only) to 4.33% by online learning based ADP.

The further analysis were performed to try different load conditions from 700 W to 100 W; however, the same ADP approaches were used. Fig. 5.10 shows the THD comparison of the source current for above mentioned controllers under different loading conditions. Without the power inverter connected, the THD was poor (above 25%) for all loading conditions. Because, the THD increases with a decrease in load, the online learning based ADP control produced the best THD results for the source current even under low load conditions. Generally, the THD improvement from a high value to low value can be easily achieved by connecting the power inverter. However, further improvement of THD at lower level is a challenging task which is the major contribution of the research in this chapter. For loads from 700 to 100 W, PI and the ADP controller together achieved further THD improvement which proves the effectiveness of the proposed controller under different operating modes unlike passive filters. On average, the

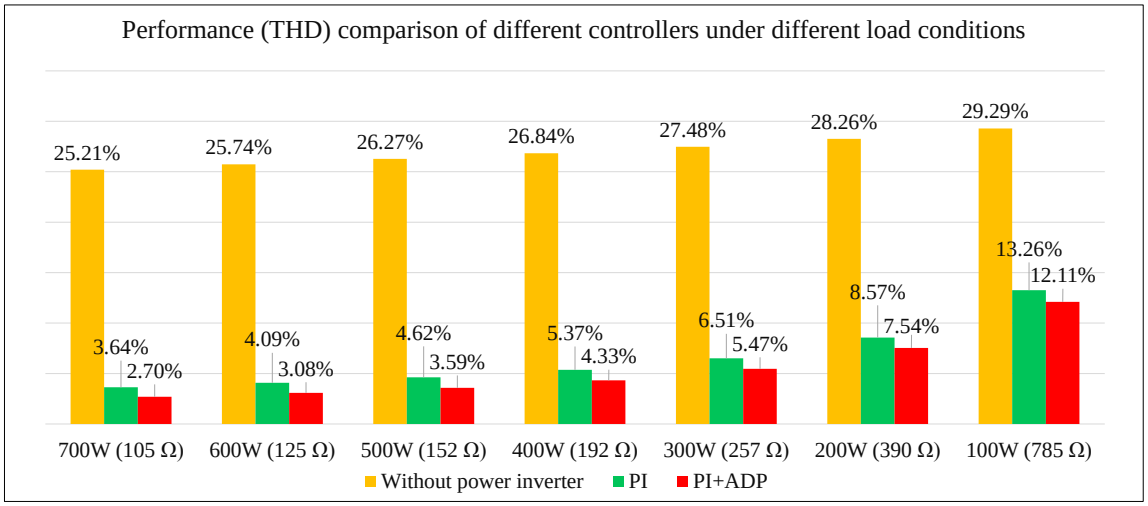


Figure 5.10. Performance comparison of PI and ADP under different loading conditions.

ADP controller reduced THD by 18.41%.

To observe a clear difference between the PI controller and ADP, the worst case among all the above load conditions described previously (i.e., for a load of 100W) was considered for detailed individual harmonic analysis. The individual harmonics of the PI and ADP controllers were compared in Fig. 5.11. The 3^{rd} harmonics is not presented here since it was close to 0 in all cases. It can be seen that ADP outperformed PI even in this worst-case condition and for different harmonic spectrums as well. According to the IEEE recommendations for harmonic control in power systems defined in IEEE Std. 519 [119], requirements on individual odd harmonic components of the source current were satisfied by the PI controller up to the 19^{th} harmonics i.e., it should be less than 1.5%. However, for the 23^{rd} to 31^{st} harmonics, the PI controller violated the standard which mentions that the harmonics should be less than 0.6%. That requirement was met by the proposed online learning based ADP control approach. It can also be observed from Fig. 5.11 that the PI+ADP could achieve significant improvement in harmonics beyond the 23^{rd} harmonics.

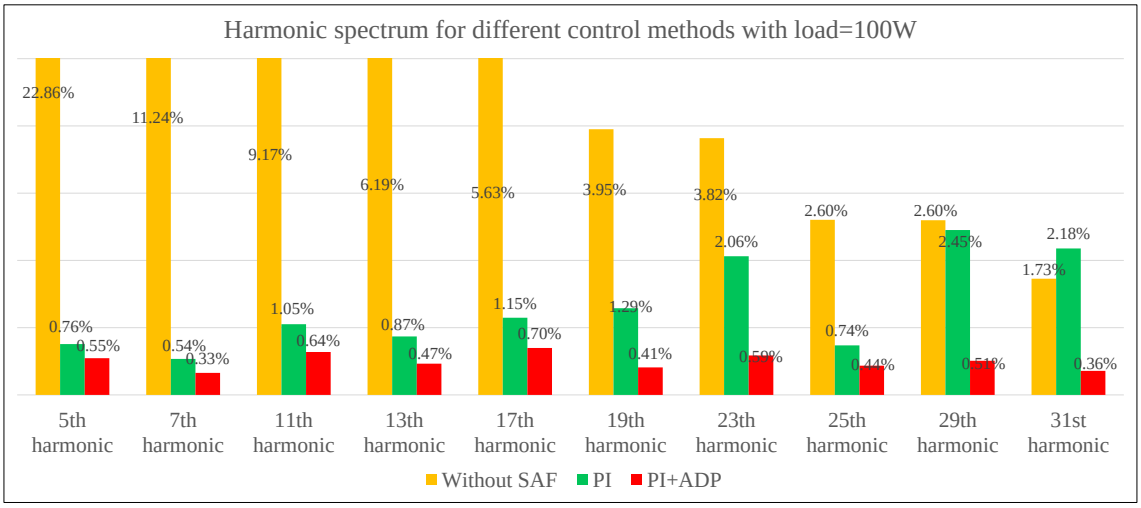


Figure 5.11. Comparison of dominant harmonics for PI and ADP under load=100W. The blue dashed line shows IEEE Std. 519 limits.

5.4 Summary

In this chapter, an online learning based ADP controller for a d-axis and q-axis current control was proposed so that I_d and I_q current effectively track their respective reference currents. The increased speed of the PI controller with the online learning based control improved the harmonic compensating capability of the power inverter. This subsequently reduced the THD of the source current by an average of 18.41% compared to a traditional PI controller alone. With the advantage of NN weights updating rules and a properly defined reinforcement signal, the ADP controller adapted its key parameters when the system was under different conditions. The ADP controller produced the fastest response time, low overshoot and in general, the best performance in comparison to the traditional PI controller. This controller improved the power quality problem of harmonics in case of dynamic non-linear load conditions.

CHAPTER 6 Conclusions and Future Work

First, the history experience was successfully integrated into the traditional ADP design. The key idea of proposed approach was to simplify the prior extensive training and reduce training time associated with the ADP controller thereby preserving the online, model-free learning capability of the ADP. A detailed design architecture and the methodology adapted was presented. In addition, a systematic approach is proposed to integrate history experience in both critic and action networks of ADP controller design. Simulation analysis was performed on two case studies: a cart-pole model and a triple-link inverted pendulum model to demonstrate the superior learning capability of the integrated approach. The statistical results show that the proposed approach can improve the required average number of trials to succeed and also the success rate. In general, the proposed approach improved the required average trial to succeed by 26.5% for cart-pole and 43% for triple-link balancing tasks. There are many future directions along this work. The history experience based ADP can be tested on one of the challenging path planning problems: maze navigation with obstacles. This new architecture also finds applications in smart grid and power electronic converters for improving performance of existing controllers. Currently, the short term memory of the history experience size of 10 is used for simulations. However, the more efficient use of long term memory might further improve the results. The time complexity of the proposed ADP controller can be solved with parallel computing techniques in recent simulation environments which is one of our future research projects.

Second, the prioritized experience replay was successfully integrated into the

traditional ADP design. The statistical results show that the proposed approach can improve the required average number of trials to succeed and the success rate. The proposed training method improved the required average trial to succeed compared to traditional ADP controller by 60.56% for cart-pole and 56.89% for triple-link balancing tasks. This method was far superior than experience replay integrated ADP, which could achieve only 23% improvement for cart-pole and 43% improvement for triple-link case studies. In addition stability of the proposed method has been verified by constructing the Lyanupov function and detailed theoretical analysis has been presented to show that the errors between the optimal network weights and their respective estimations are UUB.

Third, a supplementary ADP controller for d-axis current control is proposed so that I_d current effectively tracks reference current. It plays an important role for the effective application of VSM, not only under sudden step change condition, but also under the fault in three-phase system such as unbalanced single line to ground fault. The ADP controller produced the fastest response time, low overshoot and in general, the best performance in comparison to the traditional PI controller. With the advantage of NN weights updating rules, the ADP controller adopted its key parameters when the system is under different faults. More robust results were obtained by ADP controller in comparison with the conventional PI controller. This VSM system with adaptive control has long-term benefit to micro-grids and data center applications. The coordinated control for both inner current loop and outer frequency loop of VSM is expected to be more effective for the real application. One of the future work might be on the supplementary ADP design for coordinated control in this problem.

Finally, an online learning based ADP controller for a d-axis and q-axis current

control was proposed so that I_d and I_q current effectively track their respective reference currents. The proposed controller works alongside existing proportional integral (PI) controllers to efficiently track the reference currents in the $d-q$ domain. It can generate adaptive control actions to compensate PI controller. We have also included the simulation results without connecting the traditional PI control based power inverter for reference comparison. The proposed system was simulated under different non-linear (three-phase full wave rectifier) load conditions. The grid connected inverter system was simulated in MATLAB/Simulink to analyze transient stability problems. The performance of the proposed approach was compared with the traditional approach. The increased speed of the PI controller with the online learning based control improved the harmonic compensating capability of the power inverter. This subsequently reduced the THD of the source current by an average of 18.41% compared to a traditional PI controller alone. With the advantage of NN weights updating rules and a properly defined reinforcement signal, the ADP controller adapted its key parameters when the system was under different conditions. The ADP controller produced the fastest response time, low overshoot and in general, the best performance in comparison to the traditional PI controller. This controller improved the power quality problem of harmonics in case of dynamic non-linear load conditions. The improvement in the dynamics of the inner current control loop achieved in this work can have a number of applications for other power electronic systems such as grid-connected inverters, STATCOMs, virtual synchronous machines, etc. For future work, a hardware experiment system for the verification of these simulation results is in planning for development.

REFERENCES

[1] R. S. Sutton and A. G. Barto, *Reinforcement learning: An introduction*, 1st ed. MIT press, Cambridge, MA, 1998.

[2] F. L. Lewis and D. Vrabie, "Reinforcement learning and adaptive dynamic programming for feedback control," *IEEE circuits and systems magazine*, vol. 9, no. 3, pp. 32–50, 2009.

[3] T. M. Mitchell, "Machine learning. 1997," *Burr Ridge, IL: McGraw Hill*, vol. 45, 1997.

[4] S. Lange, M. Riedmiller, and A. Voigtländer, "Autonomous reinforcement learning on raw visual input data in a real world application," in *The 2012 International Joint Conference on Neural Networks (IJCNN)*, 2012, pp. 1–8.

[5] T. Matiisen, "Guest Post (Part I): Demystifying Deep Reinforcement Learning - Nervana," 2016. [Online]. Available: https://www.nervanasys.com/demystifying-deep-reinforcement-learning/

[6] V. Mnih, K. Kavukcuoglu, D. Silver, A. A. Rusu, J. Veness, M. G. Bellemare, A. Graves, M. Riedmiller, A. K. Fidjeland, G. Ostrovski *et al.*, "Human-level control through deep reinforcement learning," *Nature*, vol. 518, no. 7540, pp. 529–533, 2015.

[7] Y. B. Yann LeCun and G. Hinton, "Deep learning," *Nature*, vol. 521, pp. 436–444, 2015.

[8] V. Mnih, K. Kavukcuoglu, D. Silver, A. Graves, I. Antonoglou, D. Wierstra, and M. Riedmiller, "Playing atari with deep reinforcement learning," in *NIPS Deep Learning Workshop*, 2013.

[9] S. Gu, T. Lillicrap, I. Sutskever, and S. Levine, "Continuous deep q-learning with model-based acceleration," *arXiv preprint arXiv:1603.00748*, 2016.

[10] H. Van Hasselt, A. Guez, and D. Silver, "Deep reinforcement learning with double q-learning," *arXiv:1509.06461*, 2015.

[11] P. J. Werbos, "Backpropagation through time: What it does and how to do it," *Proceedings of the IEEE*, vol. 78, no. 10, pp. 1550–1560, 2002.

[12] S. Lange and M. Riedmiller, "Deep auto-encoder neural networks in reinforcement learning," in *The 2010 International Joint Conference on Neural Networks (IJCNN)*, 2010, pp. 1–8.

[13] S. Levine, C. Finn, T. Darrell, and P. Abbeel, "End-to-end training of deep visuomotor policies," *Journal of Machine Learning Research*, vol. 17, no. 39, pp. 1–40, 2016.

[14] D. Ernst, R. Marée, and L. Wehenkel, "Reinforcement learning with raw image pixels as input state," in *Advances in Machine Vision, Image Processing, and Pattern Analysis*. Springer, 2006, pp. 446–454.

[15] S. Gu, E. Holly, T. Lillicrap, and S. Levine, "Deep reinforcement learning for robotic manipulation," *arXiv preprint arXiv:1610.00633*, 2016.

[16] L. Tai, S. Li, and M. Liu, "A deep-network solution towards modelless obstacle avoidance," in *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2016.

[17] F. Lewis and D. Liu, Eds., *Reinforcement Learning and Approximate Dynamic Programming for Feedback Control*. Piscataway, NJ, USA: Wiley-IEEE Press, 2013.

[18] J. Si, A. G. Barto, W. B. Powell, and D. C. Wunsch, *Handbook of learning and approximate dynamic programming*, 1st ed. IEEE Press, Hoboken, NJ, 2004.

[19] W. Powell, *Approximate Dynamic Programming: Solving the curses of dimensionality*. Wiley-Blackwell, 2007.

[20] H. He, *Self-Adaptive Systems for Machine Intelligence*. Wiley, 2011.

[21] F.-Y. Wang, H. Zhang, and D. Liu, "Adaptive dynamic programming: An introduction," *IEEE Computational Intelligence Magazine*, vol. 4, no. 2, pp. 39–47, 2009.

[22] D. V. Prokhorov and D. C. Wunsch, "Adaptive critic designs," *IEEE Trans. on Neural Netw.*, vol. 8, no. 5, pp. 997–1007, 1997.

[23] H.-G. Zhang, X. Zhang, L. Yan-Hong, and Y. Jun, "An overview of research on adaptive dynamic programming," *Acta Automatica Sinica*, vol. 39, no. 4, pp. 303–311, 2013.

[24] J. Si and Y.-T. Wang, "Online learning control by association and reinforcement," *IEEE Transactions on Neural Networks*, vol. 12, no. 2, pp. 264–276, 2001.

[25] H. He, Z. Ni, and J. Fu, "A three-network architecture for on-line learning and optimization based on adaptive dynamic programming," *Neurocomputing*, vol. 78, no. 1, pp. 3–13, 2012.

[26] Z. Ni, H. He, and J. Wen, "Adaptive learning in tracking control based on the dual critic network design," *IEEE Trans. on Neural Networks and Learning Systems*, vol. 24, no. 6, pp. 913–928, 2013.

[27] D. Liu, D. Wang, D. Zhao, Q. Wei, and N. Jin, "Neural-network-based optimal control for a class of unknown discrete-time nonlinear systems using globalized dual heuristic programming," *IEEE Transactions on Automation Science and Engineering*, vol. 9, no. 3, pp. 628–634, 2012.

[28] D. Wang, D. Liu, Q. Wei, D. Zhao, and N. Jin, "Optimal control of unknown nonaffine nonlinear discrete-time systems based on adaptive dynamic programming," *Automatica*, vol. 48, no. 8, pp. 1825–1832, 2012.

[29] S. Ray, G. K. Venayagamoorthy, B. Chaudhuri, and R. Majumder, "Comparison of adaptive critics and classical approaches based wide area controllers for a power

system," *IEEE Trans. on Syst. Man, Cybern., Part B*, vol. 38, no. 4, pp. 1002–1007, 2008.

[30] W. Qiao, G. Venayagamoorthy, and R. Harley, "DHP-based wide-area coordinating control of a power system with a large wind farm and multiple FACTS devices," in *Proc. IEEE Int. Conf. Neural Netw.*, 2007, pp. 2093–2098.

[31] D. Wang, C. Mu, H. He, and D. Liu, "Event-driven adaptive robust control of nonlinear systems with uncertainties through NDP strategy," *IEEE Transactions on Systems, Man, and Cybernetics: Systems*, 2016.

[32] D. Wang, C. Li, D. Liu, and C. Mu, "Data-based robust optimal control of continuous-time affine nonlinear systems with matched uncertainties," *Information Sciences*, vol. 366, pp. 121–133, 2016.

[33] F. Liu, J. Sun, J. Si, W. Guo, and S. Mei, "A boundedness result for the direct heuristic dynamic programming," *Neural Networks*, vol. 32, pp. 229–235, 2012.

[34] A. Al-Tamimi, F. L. Lewis, and M. Abu-Khalaf, "Discrete-time nonlinear hjb solution using approximate dynamic programming: Convergence proof," *IEEE Transactions on System, Man and Cybernetics, Part B*, vol. 38, no. 4, pp. 943–949, 2008.

[35] H. G. Zhang, Q. L. Wei, and Y. H. Luo, "A novel infinite-time optimal tracking control scheme for a class of discrete-time nonlinear systems via the greedy hdp iteration algorithm," *IEEE Transactions on System, Man and Cybernetics, Part B*, vol. 38, no. 4, pp. 937–942, 2008.

[36] H. G. Zhang, Y. H. Luo, and D. Liu, "Neural-network-based near-optimal control for a class of discrete-time affine nonlinear systems with control constraints," *IEEE Transactions on Neural Networks*, vol. 20, no. 9, pp. 1490–1503, 2009.

[37] H. Modares, F. L. Lewis, and M.-B. Naghibi-Sistani, "Integral reinforcement learning and experience replay for adaptive optimal control of partially-unknown constrained-input continuous-time systems," *Automatica*, vol. 50, no. 1, pp. 193–202, 2014.

[38] S. Adam, L. Busoniu, and R. Babuska, "Experience replay for real-time reinforcement learning control," *IEEE Transactions on Systems, Man, and Cybernetics, Part C (Applications and Reviews)*, vol. 42, no. 2, pp. 201–212, 2012.

[39] C. W. Anderson, M. Lee, and D. L. Elliott, "Faster reinforcement learning after pretraining deep networks to predict state dynamics," in *2015 International Joint Conference on Neural Networks (IJCNN)*, 2015, pp. 1–7.

[40] B. Wang, D. Zhao, J. Cheng, Y. Xu, and Y. Li, "A general adaptive dynamic programming approach with experience replay," in *Neural Networks (IJCNN), 2016 International Joint Conference on*, 2016, pp. 3550–3555.

[41] D. Zhao, Q. Zhang, D. Wang, and Y. Zhu, "Experience replay for optimal control of nonzero-sum game systems with unknown dynamics," *IEEE transactions on cybernetics*, vol. 46, no. 3, pp. 854–865, 2016.

[42] R. Bellman, *Dynamic Programming*, 1st ed. Princeton University Press, 1957.

[43] S. Sharma, I. Umar, L. Ospina, D. Wong, and H. Tizhoosh, "Stacked autoencoders for medical image search," *arXiv preprint arXiv:1610.00320*, 2016.

[44] R. Salakhutdinov and G. Hinton, "Semantic hashing," *International Journal of Approximate Reasoning*, vol. 50, no. 7, pp. 969–978, 2009.

[45] P. Mirowski, M. Ranzato, and Y. LeCun, "Dynamic auto-encoders for semantic indexing," in *Proceedings of the NIPS 2010 Workshop on Deep Learning*, 2010, pp. 1–9.

[46] G. E. Hinton and R. R. Salakhutdinov, "Reducing the dimensionality of data with neural networks," *Science*, vol. 313, no. 5786, pp. 504–507, 2006.

[47] C. Harrigan, "Deep reinforcement learning with regularized convolutional neural fitted Q iteration," *differences*, vol. 14, pp. 1–12.

[48] Y. LeCun, L. Bottou, Y. Bengio, and P. Haffner, "Gradient-based learning applied to document recognition," *Proceedings of the IEEE*, vol. 86, no. 11, pp. 2278–2324, 1998.

[49] J. T. Springenberg, A. Dosovitskiy, T. Brox, and M. Riedmiller, "Striving for simplicity: The all convolutional net," *arXiv preprint arXiv:1412.6806*, 2014.

[50] "Unsupervised feature learning and deep learning tutorial." [Online]. Available: http://ufldl.stanford.edu/tutorial/supervised/ConvolutionalNeuralNetwork/

[51] A. Nair, P. Srinivasan, S. Blackwell, C. Alcicek, R. Fearon, A. D. Maria, V. Panneershelvam, M. Suleyman, C. Beattie, S. Petersen, S. Legg, V. Mnih,

K. Kavukcuoglu, and D. Silver, “Massively parallel methods for deep reinforcement learning,” *CoRR*, vol. abs/1507.04296, 2015.

[52] L.-J. Lin, “Self-improving reactive agents based on reinforcement learning, planning and teaching,” *Machine learning*, vol. 8, no. 3-4, pp. 293–321, 1992.

[53] T. de Bruin, J. Kober, K. Tuyls, and R. Babuška, “The importance of experience replay database composition in deep reinforcement learning,” in *Deep Reinforcement Learning Workshop, NIPS*, 2015.

[54] T. P. Lillicrap, J. J. Hunt, A. Pritzel, N. Heess, T. Erez, Y. Tassa, D. Silver, and D. Wierstra, “Continuous control with deep reinforcement learning,” *CoRR*, vol. abs/1509.02971, 2015.

[55] X. Wang, “Deep reinforcement learning,” Master’s thesis, Technische Universiteit Eindhoven, Eindhoven, 2016.

[56] L. Busoniu, R. Babuska, B. De Schutter, and D. Ernst, *Reinforcement learning and dynamic programming using function approximators*. CRC press, 2010, vol. 39.

[57] P. WawrzyńSki and A. K. Tanwani, “Autonomous reinforcement learning with experience replay,” *Neural Networks*, vol. 41, pp. 156–167, 2013.

[58] M. Riedmiller and H. Braun, “A direct adaptive method for faster backpropagation learning: The RPROP algorithm,” in *IEEE International Conference on Neural Networks*, 1993, pp. 586–591.

[59] V. Mnih, A. P. Badia, M. Mirza, A. Graves, T. P. Lillicrap, T. Harley, D. Silver, and K. Kavukcuoglu, "Asynchronous methods for deep reinforcement learning," *CoRR*, vol. abs/1602.01783, 2016.

[60] T. Tieleman and G. Hinton, "Lecture 6.5-rmsprop: Divide the gradient by a running average of its recent magnitude," *COURSERA: Neural Networks for Machine Learning*, vol. 4, no. 2, 2012.

[61] Z. Wang, N. de Freitas, and M. Lanctot, "Dueling network architectures for deep reinforcement learning," *CoRR*, vol. abs/1511.06581, 2015.

[62] W. Guo, F. Liu, J. Si, D. He, R. Harley, and S. Mei, "Approximate dynamic programming based supplementary reactive power control for DFIG wind farm to enhance power system stability," *Neurocomputing*, vol. 170, pp. 417–427, 2015.

[63] Z. Ni, H. He, X. Zhong, and D. V. Prokhorov, "Model-free dual heuristic dynamic programming," *IEEE transactions on neural networks and learning systems*, vol. 26, no. 8, pp. 1834–1839, 2015.

[64] B. Luo, D. Liu, T. Huang, and D. Wang, "Model-free optimal tracking control via critic-only Q-learning," *IEEE Transactions on Neural Networks and Learning Systems*, vol. 27, no. 10, pp. 2134–2144, 2016.

[65] B. Luo, D. Liu, H.-N. Wu, D. Wang, and F. L. Lewis, "Policy Gradient Adaptive Dynamic Programming for Data-Based Optimal Control," *IEEE Transactions on Cybernetics*, 2016.

[66] S. Poudel, Z. Ni, and N. Malla, “Real-time cyber physical system testbed for power system security and control,” *International Journal of Electrical Power & Energy Systems*, vol. 90, pp. 124–133, 2017.

[67] N. Malla, U. Tamrakar, D. Shrestha, Z. Ni, and R. Tonkoski, “Online learning control for harmonics reduction based on current controlled voltage source power inverters,” *IEEE/CAA Journal of Automatica Sinica*, vol. 4, no. 3, pp. 447–457, 2017.

[68] N. Malla, D. Shrestha, Z. Ni, and R. Tonkoski, “Supplementary control for virtual synchronous machine based on adaptive dynamic programming,” in *2016 IEEE Congress on Evolutionary Computation (CEC)*, 2016, pp. 1998–2005.

[69] B. Xu, C. Yang, and Z. Shi, “Reinforcement learning output feedback NN control using deterministic learning technique,” *IEEE Transactions on Neural Networks and Learning Systems*, vol. 25, no. 3, pp. 635–641, 2014.

[70] Z. Ni, H. He, and J. Wen, “Adaptive learning in tracking control based on the dual critic network design,” *IEEE transactions on neural networks and learning systems*, vol. 24, no. 6, pp. 913–928, 2013.

[71] X. Luo and J. Si, “Stability of direct heuristic dynamic programming for nonlinear tracking control using PID neural network,” in *The 2013 International Joint Conference on Neural Networks (IJCNN)*, 2013, pp. 1–7.

[72] X. Zhong, Z. Ni, and H. He, "A Theoretical Foundation of Goal Representation Heuristic Dynamic Programming," *IEEE Transactions on Neural Networks and Learning Systems*, vol. 27, no. 12, pp. 2513–2525, 2016.

[73] F. Liu, J. Sun, J. Si, W. Guo, and S. Mei, "A boundedness result for the direct heuristic dynamic programming," *Neural Networks*, vol. 32, pp. 229–235, 2012.

[74] L. Yang, J. Si, K. S. Tsakalis, and A. A. Rodriguez, "Direct heuristic dynamic programming for nonlinear tracking control with filtered tracking error," *IEEE Transactions on Systems, Man, and Cybernetics, Part B (Cybernetics)*, vol. 39, no. 6, pp. 1617–1622, 2009.

[75] Y. Zhu and D. Zhao, "Comprehensive comparison of online ADP algorithms for continuous-time optimal control," *Artificial Intelligence Review*, pp. 1–17, 2017.

[76] P. Cichosz, "TD (λ) learning without eligibility traces: a theoretical analysis," *Journal of Experimental & Theoretical Artificial Intelligence*, vol. 11, no. 2, pp. 239–263, 1999.

[77] P. Wawrzyński, "Real-time reinforcement learning by sequential actor–critics and experience replay," *Neural Networks*, vol. 22, no. 10, pp. 1484–1497, 2009.

[78] S. Lange, T. Gabel, and M. Riedmiller, "Batch reinforcement learning," in *Reinforcement learning*. Springer, 2012, pp. 45–73.

[79] J. McCaffrey, "How to use resilient back propagation to train neural networks – visual studio magazine." [Online]. Available: https://visualstudiomagazine.com/Articles/2015/03/01/Resilient-Back-Propagation.aspx?Page=1

[80] H. He, Z. Ni, and J. Fu, "A three-network architecture for on-line learning and optimization based on adaptive dynamic programming," *Neurocomputing*, vol. 78, no. 1, pp. 3–13, 2012.

[81] Z. Ni, H. He, D. Zhao, X. Xu, and D. V. Prokhorov, "GrDHP: A general utility function representation for dual heuristic dynamic programming," *IEEE transactions on neural networks and learning systems*, vol. 26, no. 3, pp. 614–627, 2015.

[82] T. Schaul, J. Quan, I. Antonoglou, and D. Silver, "Prioritized experience replay," *CoRR*, vol. abs/1511.05952, 2015. [Online]. Available: http://arxiv.org/abs/1511.05952

[83] A. W. Moore and C. G. Atkeson, "Prioritized sweeping: Reinforcement learning with less data and less time," *Machine learning*, vol. 13, no. 1, pp. 103–130, 1993.

[84] Y. Li, "Deep reinforcement learning: An overview," *CoRR*, vol. abs/1701.07274, 2017. [Online]. Available: http://arxiv.org/abs/1701.07274

[85] J. Zhai, Q. Liu, Z. Zhang, S. Zhong, H. Zhu, P. Zhang, and C. Sun, "Deep q-learning with prioritized sampling," in *International Conference on Neural Information Processing*. Springer, 2016, pp. 13–22.

[86] "Let's make a DQN: Double Learning and Prioritized Experience Replay," 2017. [Online]. Available: https://jaromiru.com/2016/11/07/lets-make-a-dqn-double-learning-and-prioritized-experience-replay/

[87] "Introduction to prioritized experience replay," 2017. [Online]. Available: http://www.slideshare.net/ssuser07aa33/introduction-to-prioritized-experience-replay

[88] Z. Wang, V. Bapst, N. Heess, V. Mnih, R. Munos, K. Kavukcuoglu, and N. de Freitas, "Sample efficient actor-critic with experience replay," *CoRR*, vol. abs/1611.01224, 2016. [Online]. Available: http://arxiv.org/abs/1611.01224

[89] N. Malla and Z. Ni, "A new history experience replay design for model-free adaptive dynamic programming," *Neurocomputing*, vol. 266, pp. 141–149, 2017.

[90] R. AbouSleiman, "Roulette Wheel Selection - File Exchange - MATLAB Central," 2017. [Online]. Available: https://www.mathworks.com/matlabcentral/fileexchange/45735-roulette-wheel-selection?requestedDomain=www.mathworks.com

[91] A. Gurung, R. Tonkoski, D. Galipeau, and I. Tamrakar, "Feasibility Study of Photovoltaic-Hydropower Microgrids," in *5th International Conference on Power and Energy Systems (ICPS)*, 2014.

[92] Q. C. Zhong and G. Weiss, "Synchronverters: Inverters that mimic synchronous generators," *IEEE Transactions on Industrial Electronics*, vol. 58, no. 4, pp. 1259–1267, 2011.

[93] H. Bevrani, T. Ise, and Y. Miura, "Virtual synchronous generators: A survey and new perspectives," *International Journal of Electrical Power & Energy Systems*, vol. 54, pp. 244–254, 2014.

[94] S. Li, M. Fairbank, C. Johnson, D. C. Wunsch, E. Alonso, and J. L. Proao, "Artificial neural networks for control of a grid-connected rectifier/inverter under disturbance, dynamic and power converter switching conditions," *IEEE Transactions on Neural Networks and Learning Systems*, vol. 25, no. 4, pp. 738–750, 2014.

[95] S. Li, M. Fairbank, X. Fu, D. C. Wunsch, and E. Alonso, "Nested-loop neural network vector control of permanent magnet synchronous motors," in *The 2013 International Joint Conference on Neural Networks (IJCNN)*, 2013, pp. 1–8.

[96] Y. Tang, J. Yang, J. Yan, and H. He, "Intelligent load frequency controller using GrADP for island smart grid with electric vehicles and renewable resources," *Neurocomputing*, vol. 170, pp. 406–416, 2015.

[97] S. Li, D. C. Wunsch, M. Fairbank, and E. Alonso, "Vector control of a grid-connected rectifier/inverter using an artificial neural network," in *The 2012 International Joint Conference on Neural Networks (IJCNN)*, 2012, pp. 1–7.

[98] M. Fairbank, S. Li, X. Fu, E. Alonso, and D. Wunsch, "An adaptive recurrent neural-network controller using a stabilization matrix and predictive inputs to solve a tracking problem under disturbances," *Neural Networks*, vol. 49, pp. 74–86, 2014.

[99] X. Fu, S. Li, M. Fairbank, D. C. Wunsch, and E. Alonso, "Training recurrent neural networks with the Levenberg–Marquardt algorithm for optimal control of a grid-connected converter," *IEEE Transactions on Neural Networks and Learning Systems*, vol. 26, no. 9, pp. 1900–1912, Sep. 2015.

[100] M. Suresh, A. K. Panda, Y. Suresh *et al.*, "Fuzzy controller based 3phase 4wire shunt active Filter for mitigation of current harmonics with combined pq and Id-Iq control strategies," *Energy and Power Engineering*, vol. 3, no. 01, pp. 43–52, 2011.

[101] J. Dai, G. K. Venayagamoorthy, R. G. Harley, Y. Deng, and S. M. Potter, "Adaptive-critic-based control of a synchronous generator in a power system using biologically inspired artificial neural networks," in *2015 International Joint Conference on Neural Networks (IJCNN)*, 2015, pp. 1–8.

[102] Z. Ni, Y. Tang, H. He, and J. Wen, "Multi-machine power system control based on dual heuristic dynamic programming," in *2014 IEEE Symposium on Computational Intelligence Applications in Smart Grid (CIASG)*, 2014, pp. 1–7.

[103] Z. Ni, Y. Tang, X. Sui, H. He, and J. Wen, "An adaptive neuro-control approach for multi-machine power systems," *International Journal of Electrical Power & Energy Systems*, vol. 75, pp. 108–116, 2016.

[104] C. Lu, J. Si, and X. Xie, "Direct heuristic dynamic programming for damping oscillations in a large power system," *IEEE Transactions on Systems, Man, and Cybernetics, Part B (Cybernetics)*, vol. 38, no. 4, pp. 1008–1013, 2008.

[105] W. Guo, F. Liu, J. Si, D. He, R. Harley, and S. Mei, "Online supplementary ADP learning controller design and application to power system frequency control with large-scale wind energy integration," *IEEE Transactions on Neural Networks and Learning Systems*, vol. 27, no. 8, pp. 1748–1761, Aug 2016.

[106] Y. Tang, H. He, Z. Ni, J. Wen, and X. Sui, "Reactive power control of grid-connected wind farm based on adaptive dynamic programming," *Neurocomputing*, vol. 125, pp. 125–133, 2014.

[107] Y. Tang, H. He, Z. Ni, J. Wen, and T. Huang, "Adaptive modulation for DFIG and STATCOM with high-voltage direct current transmission," *IEEE Transactions on Neural Networks and Learning Systems*, vol. 27, no. 8, pp. 1762–1772, Aug 2016.

[108] Y. Tang, H. He, Z. Ni, X. Zhong, D. Zhao, and X. Xu, "Fuzzy-based goal representation adaptive dynamic programming," *IEEE Transactions on Fuzzy Systems*, vol. 24, no. 5, pp. 1159–1175, 2016.

[109] U. Tamrakar, D. Galipeau, R. Tonkoski, and I. Tamrakar, "Improving transient stability of photovoltaic-hydro microgrids using virtual synchronous machines," in *Proceedings of 2015 IEEE Eindhoven PowerTech*, 2015, pp. 1–6.

[110] J. C. Daly, "Overshoot as a function of phase margin," 2003. [Online]. Available: http://www.ele.uri.edu/~daly/535/margin.html

[111] U. Tamrakar, "Improvement of transient stability of photovoltaic-hydro microgrids using virtual synchronous machines," Master's thesis, South Dakota State University, Brookings, SD, USA, 2015.

[112] P. J. Werbos, "Neurocontrol and supervised learning: An overview and evaluation," *Handbook of intelligent control*, vol. 65, p. 89, 1992.

[113] A. Timbus, M. Liserre, R. Teodorescu, P. Rodriguez, and F. Blaabjerg, "Evaluation of current controllers for distributed power generation systems," *IEEE Transactions on Power Electronics*, vol. 24, no. 3, pp. 654–664, 2009.

[114] ECPE Europen Center for Power Electronics, EPE European power Electronics and Drives Association. Position paper on energy efficiency-the role of power electronics, in *European Workshop on Energy Efficiency-the Role of Power Electronics*, February 2007.

[115] D. O. Abdeslam, P. Wira, J. Mercklé, D. Flieller, and Y.-A. Chapuis, "A unified artificial neural network architecture for active power filters," *IEEE Transactions on Industrial Electronics*, vol. 54, no. 1, pp. 61–76, 2007.

[116] F. S. dos Reis, J. Ale, F. D. Adegas, R. Tonkoski, S. Slan, and K. Tan, "Active shunt filter for harmonic mitigation in wind turbines generators," in *Power Electronics Specialists Conference*, 2006, pp. 1–6.

[117] L. Asiminoaei, F. Blaabjerg, and S. Hansen, "Detection is key-harmonic detection methods for active power filter applications," *IEEE Industry Applications Magazine*, vol. 13, no. 4, pp. 22–33, 2007.

[118] L. Marconi, F. Ronchi, and A. Tilli, "Robust nonlinear control of shunt active filters for harmonic current compensation," *Automatica*, vol. 43, no. 2, pp. 252–263, 2007.

[119] "IEEE recommended practice and requirements for harmonic control in electric power systems," *IEEE Std 519-2014 (Revision of IEEE Std 519-1992)*, pp. 1–29, June 2014.

[120] J. Vazquez and P. Salmeron, "Active power filter control using neural network technologies," *IEE Proceedings-Electric Power Applications*, vol. 150, no. 2, pp. 139–145, 2003.

[121] H. Akagi, A. Nabae, and S. Atoh, "Control strategy of active power filters using multiple voltage-source PWM converters," *IEEE Transactions on Industry Applications*, no. 3, pp. 460–465, 1986.

[122] V. Soares, P. Verdelho, and G. D. Marques, "An instantaneous active and reactive current component method for active filters," *IEEE Transactions on Power Electronics*, vol. 15, no. 4, pp. 660–669, 2000.

[123] S. Buso, L. Malesani, and P. Mattavelli, "Comparison of current control techniques for active filter applications," *IEEE Transactions on Industrial Electronics*, vol. 45, no. 5, pp. 722–729, 1998.

[124] L. L. Lai, *Intelligent system applications in power engineering: evolutionary programming and neural networks*. John Wiley & Sons, Inc., 1998.

[125] M. M. A. Radzi and N. A. Rahim, "Neural network and bandless hysteresis approach to control switched capacitor active power filter for reduction of harmonics," *IEEE Transactions on Industrial Electronics*, vol. 56, no. 5, pp. 1477–1484, 2009.

[126] R. Aguilera, P. Acuna, P. Lezana, G. Konstantinou, B. Wu, S. Bernet, and V. Agelidis, "Selective harmonic elimination model predictive control for

multilevel power converters," *IEEE Transactions on Power Electronics*, vol. 32, no. 3, pp. 2416–2426, March 2017.

[127] H. A. Young, M. A. Perez, and J. Rodriguez, "Analysis of finite-control-set model predictive current control with model parameter mismatch in a three-phase inverter," *IEEE Transactions on Industrial Electronics*, vol. 63, no. 5, pp. 3100–3107, 2016.

[128] Q. Wei, D. Liu, Q. Lin, and R. Song, "Adaptive dynamic programming for discrete-time zero-sum games," *IEEE Transactions on Neural Networks and Learning Systems*, to be published, doi: 10.1109/TNNLS.2016.2638863.

[129] Z. Ni, H. He, J. Wen, and X. Xu, "Goal representation heuristic dynamic programming on maze navigation," *IEEE Transactions on Neural Networks and Learning Systems*, vol. 24, no. 12, pp. 2038–2050, 2013.

[130] J. Na and G. Herrmann, "Online adaptive approximate optimal tracking control with simplified dual approximation structure for continuous-time unknown nonlinear systems," *IEEE/CAA Journal of Automatica Sinica*, vol. 1, no. 4, pp. 412–422, 2014.

[131] W. Guo, F. Liu, J. Si, and S. Mei, "Incorporating approximate dynamic programming-based parameter tuning into PD-type virtual inertia control of DFIGs," in *The 2013 International Joint Conference on Neural Networks (IJCNN)*, 2013, pp. 1–8.

[132] Z. Ni, Y. Tang, H. He, and J. Wen, "Multi-machine power system control based on dual heuristic dynamic programming," in *2014 IEEE Symposium on Computational Intelligence Applications in Smart Grid (CIASG)*, 2014, pp. 1–7.

[133] ——, "Multi-machine power system control based on dual heuristic dynamic programming," in *2014 IEEE Symposium on Computational Intelligence Applications in Smart Grid (CIASG)*, Dec. 2014, Orlando, FL, pp. 1–7.

[134] Q. Wei, D. Liu, G. Shi, and Y. Liu, "Multibattery optimal coordination control for home energy management systems via distributed iterative adaptive dynamic programming," *IEEE Transactions on Industrial Electronics*, vol. 62, no. 7, pp. 4203–4214, 2015.

[135] Y. Tang, H. He, J. Wen, and J. Liu, "Power system stability control for a wind farm based on adaptive dynamic programming," *IEEE Transactions on Smart Grid*, vol. 6, no. 1, pp. 166–177, 2015.

[136] L. Dong, Y. Tang, H. He, and C. Sun, "An Event-Triggered Approach for Load Frequency Control With Supplementary ADP," *IEEE Transactions on Power Systems*, vol. 32, no. 1, pp. 581–589, 2017.

[137] Q. Wei, D. Liu, and H. Lin, "Value iteration adaptive dynamic programming for optimal control of discrete-time nonlinear systems," *IEEE Transactions on cybernetics*, vol. 46, no. 3, pp. 840–853, 2016.

[138] Q. Wei, D. Liu, and X. Yang, "Infinite horizon self-learning optimal control of nonaffine discrete-time nonlinear systems," *IEEE transactions on neural networks and learning systems*, vol. 26, no. 4, pp. 866–879, 2015.

[139] Q. Wei, F. L. Lewis, D. Liu, R. Song, and H. Lin, "Discrete-time local value iteration adaptive dynamic programming: Convergence analysis," *IEEE Transactions on Systems, Man, and Cybernetics: Systems*, to be published, doi: 10.1109/TSMC.2016.2623766.

[140] Q. Wei, D. Liu, and Q. Lin, "Discrete-time local iterative adaptive dynamic programming: Terminations and admissibility analysis," *IEEE Transactions on Neural Networks and Learning Systems*, to be published, doi: 10.1109/TNNLS.2016.2593743.

[141] H. Zhang, J. Zhang, G.-H. Yang, and Y. Luo, "Leader-based optimal coordination control for the consensus problem of multiagent differential games via fuzzy adaptive dynamic programming," *IEEE Transactions on Fuzzy Systems*, vol. 23, no. 1, pp. 152–163, 2015.

[142] Q. Wei, R. Song, and P. Yan, "Data-driven zero-sum neuro-optimal control for a class of continuous-time unknown nonlinear systems with disturbance using adp," *IEEE transactions on neural networks and learning systems*, vol. 27, no. 2, pp. 444–458, 2016.