An Application of Hamiltonian Neurodynamics Using Pontryagin's Maximum (Minimum) Principle

Takamasa Koshizen, Phil Broadbridge and John Fulcher
Department of Computer Science, University of Wollongong
Northfield Avenue Wollongong NSW 2522, Australia

Abstract Hamiltonians can generate Artificial Neural Dynamical systems dependent on time. Classical methods from optimal control theory, notably Pontryagin's Maximum (Minimum) principle(PMP) can be employed, together with Hamiltonians, to determine the optimal system weights. Today, although several extended-backpropagation methods using optimization theory have been developed based on the well known standard back-propagation method (SBP), multilayered feedfoward neural networks are here employed on differential equations which have characteristics such as admitting neurons and weight vectors dependent on time. It is shown that the new learning rule obtained by weight equations using PMP compares favourably with SBP. The new learning rule can also be applied to feedback networks, in both continuous-time and discrete-time.

Keywords: Hamiltonian, Pontryagin Maximum (Minimum) Principle, Multilayered feedforward neural networks, standard backpropagation, Hamiltonian (affine) control system

1. Introduction

In order to develop the science and technology of neural networks to their full potential it will be important to cultivate a thorough understanding of neurodynamics [1]. Previous work on neurodynamics using Hamiltonians includes that of Ramacher et al. [2] who showed that a single first-order Hamiltonian-Jacobi method was sufficient to derive the various neurodynamical paradigms in use today. Neurodynamics can also be determined by the network topology, input patterns, and by weight dynamics attributed to learning. It is important to note that the differential equations to be obeyed by neurons and weights, respectively, constitute only *one half of* the characteristic equations associated with Ramacher's Hamiltonian-Jocobi method [2]. Moreover, the study of neurodynamics seems to be mixed with several optimization methods derived from an understanding of classical optimal control problems such as the first-order gradient algorithm, second-order algorithm and dynamical programming methods [3][4].

Optimal control problems are mainly applied to determine optimal weights for Artificial Neural Networks (ANNs) whose task is defined as the optimization of a specific performance index as will be shown later. Recently the well-known back-propagation method using steepest gradient-descent minimization of a global error function, called standard back-propagation (SBP), has inspired a great deal of interest in such networks for both cognitive modelling and practical problem solving [5]. However, the disadvantages of SBP is that it converges very slowly, often behaves very badly on large-scale problems, and can become stuck in local minima [6][7][8].

In addition, even though an optimal weight vector exists for a given network structure, the limited learning set will render a *different* optimal solution within the set for weights of optimal solutions [8]. In this paper, we firstly employ PMP , which was first produced by Pontryagin et al. [9] for control problems, to find important characteristics of weight solutions. Apart from PMP being useful for minimization theory, it can also be applied to several practical control calculations, such as motor control and multiple-link robot arms using control variables [10]. Secondly, we generate a new learning algorithm for determining optimal weight equations, in n-dimensional continuous-time and discrete-time systems, using PMP. This new algorithm is then compared with SBP in order to analyse its performance quantitatively. In Appendix I, Hamiltonian control systems [11] are introduced, and we see how PMP is eminently suited to describing such control systems.

2. Neural Network models

Although inspired by physical and neurophysiological research, the various neural network models proposed can, to date, be viewed as computing paradigms in their own right. We here discuss one class of ANN, namely the Back-propagation network (Rumenhart et al., 1986) as a computational learning model. The Back-propagation of error gradients has proven its usefulness in feedforward and feedback neural networks in tackling a large number of classification and function mapping problems [8]. This is the case despite its being susceptible to becoming stuck in local minima. The PMP learning method presented in this paper is based on the standard Back-propagation method.

2.1 Mathematical Neuron Model

We consider a mathematical neuron , comprising n inputs (i.e., input vector $\mathbf{X} = (x_1, \cdots, x_n)$) and a signal output z. $\mathbf{W} = (w_1, \cdots, w_n)$ are weights (a weight vector) applied to the inputs , h is a threshold, and g: $\mathbf{R} \rightarrow \mathbf{R}$, acts as an activation function. In addition, the activation function has been traditionally chosen to be the sigmoidal nonlinearity, as will be shown later. Thus, the neuron's input-output relation is given by

$$z = g(\sum_{i=1}^{n} w_i x_i - h) \tag{1}$$

2.2 Backpropagation and its network models

In general, the standard back-propagation method is applied to feedforward networks in order to evaluate the derivatives of the network error with respect to its connections strengths. The basic component in a feedforward network is the single " neuron " model of Section **2.1**. The feedforward neural network is constructed by interconnecting several number of neurons so as to form a network in which all connections are made in the forward direction from input to output without feed*back* loops. Such networks are described by the following equations, in the discrete-time case:

$$x_i(t+1) = g(y_i) \tag{2}$$

where

$$y_i = \sum_{j=1}^{n} w_{ij} x_j(t) \tag{3}$$

subject to (1), which is the total input to the j-th neuron, x_i is the output of the i-th neuron.

The types of training problem which can be presented to feedback (recurrent) neural networks are more numerous compared with those which make sense for feedforward networks. Moreover, it is possible to train a recurrent temporally continuous generalization of back-propagation networks. Such networks are described by the following differential equations [12], in the continuous-time case:

$$\tau \dot{x}_i(t) = -x_i(t) + g(y_i) \tag{4}$$

Alternatively, using linear approximation:

$$\tau \dot{x}_i(t) = -x_i(t) + \sum_{j=1}^{n} w_{ij}(t) g(x_j(t)) \tag{4$'$}$$

3. Optimal Control Problems and ANNs

Classical control system design is generally a trial-and-error process in which various methods of analysis are used iteratively to determine the design parameters of an "acceptable" system [15]. The objective of optimal control theory is to determine the control signals that cause a process to satisfy the physical constraints and at the same time minimize (or maximize) some performance index. In summary, an optimal control problem can be formulated using the following information:

(a) state and output equations
(b) control variables or vectors
(c) problem constraints
(d) performance index
(e) system parameters

We next show that Optimal Control Problems can be reformulated into a form more suited to ANN models. Moreover, a mathematical definition of an optimal control problem for both continuous-and discrete-time systems will be given.

3.1 The form of state equations in continuous-time and discrete-time systems

Ramacher [2] considered the following neuron differential equation:

$$\dot{\mathbf{X}}(t) = F(in(t),\ \mathbf{X}(t),\ \mathbf{W}(t);\ t)$$

where in(t) represents the neuron inputs, $\mathbf{X}(t)$ the neuron outputs, and $\mathbf{W}(t)$ the weights.

We consider an n-dimensional continuous-time neural network with describing state equation as follows

$$\dot{\mathbf{X}}(t) = F(\mathbf{X}(t),\ \mathbf{W}(t);\ t) \tag{5}$$

with initial condition:

$$\mathbf{X}(0) = \mathbf{X}_0 \tag{6}$$

Moreover, (5) is a quite general form, so that it will extend into ANNs , considering the descriptions of (2), (4) and (4)' as

$$\dot{\mathbf{X}}(t) = f(\mathbf{X}(t)) + \mathbf{W}(t)g(\mathbf{X}(t)) \tag{7}$$

or

$$\dot{x}_i(t) = f(x_i(t)) + \sum_{j=1}^{n} w_{ij}(t)g(x_j(t)) \tag{8}$$

Similarly, for an n-dimensional discrete-time system, the state equation can be represented as

$$\mathbf{X}(k+1) = f(\mathbf{X}(k)) + \mathbf{W}(k)g(\mathbf{X}(k)) \tag{9}$$

or

$$x_i(k+1) = f(x_i(k)) + \sum_{j=1}^{n} w_{ij}(k)g(x_j(k)) \tag{10}$$

where, $f(\cdot)$ and $g(\cdot)$ possess continuous first-order derivatives, and $\mathbf{W}(t)$ are control variables for the weights. In addition, the subscript k denotes the kth time interval.

3.2 The cost function in Continuous-time ANN systems

Generally speaking, an optimal control problem is formulated as the minimization of the following performance index J [13]:

$$J = h(\mathbf{X}(t_f)) + \int_{t_0}^{t_f} \psi(\mathbf{X}(t),\ \mathbf{W}(t))dt \tag{11}$$

$\mathbf{X}$ represents the state variables and $\mathbf{W}$ the controls as weight vectors. The performance index, which includes the objective function $h(\mathbf{X}(t_f))$, is to be minimized at final time t_f.

If H represents the Hamiltonian, defined as

$$H \equiv H(\mathbf{X}(t),\ \mathbf{W}(t),\ \lambda(t)) = \lambda^T(t)\dot{\mathbf{X}}(t) + \psi(\mathbf{X}(t),\ \mathbf{W}(t)) \tag{12}$$

where, λ^T is a set of Lagrange multipliers (the costate vector) [14], the cost function ψ can be interpreted as the potential which forces the system to move along the desired trajectory [15], as will be shown later.

Furthermore, when H is also defined as in (12), the performance index (13) becomes:

$$J = h(\mathbf{X}(t_f)) + \int_{t_0}^{t_f} [H(\mathbf{X}(t),\ \mathbf{W}(t),\ \lambda\ (t)) - \lambda^T(t)\mathbf{X}(t)]dt \tag{13}$$

3.3. The cost function in Discrete-time ANN systems

Similarly, we can extend this treatment to optimal control problems in discrete-time systems, as follows

$$J = h(\mathbf{X}(T)) + \sum_{k=0}^{T-1} \psi(\mathbf{X}(k),\ \mathbf{W}(k)) \tag{14}$$

$\mathbf{X}$ represents the state variables and $\mathbf{W}$ the controls as weight vectors. The performance index, which includes the objective function $h(\mathbf{X}(T))$, is to be minimized at final time T.

If H represents the Hamiltonian, defined as

$$H \equiv H(\mathbf{X}(k+1),\ \mathbf{X}(k),\ \mathbf{W}(k),\ \lambda(k)) = \lambda^T(k)\mathbf{X}(k+1) + \psi(\mathbf{X}(k),\ \mathbf{W}(k)) \tag{15}$$

where, λ^T is a set of Lagrange multipliers (the costate vector) [14], then the cost function ψ can be interpreted as the potential which will force the system to move along the desired trajectory [15], as will be shown later.

Furthermore, when the Hamiltonian is defined as in (15), the performance index becomes:

$$J = h(\mathbf{X}(T)) + \sum_{k=0}^{T-1} [H(\mathbf{X}(k+1),\ \mathbf{X}(k),\ \mathbf{W}(k),\ \lambda(k)) - \lambda^T(k)\mathbf{X}(k+1)] \tag{16}$$

4. Learning Potential Functions and PMP

The solution of an optimal control problem can be obtained through two basic approaches. The first is by the application of the *Hamiltonian-Jacobi equation* which constitutes a sufficiency condition. The other approach is *PMP* which constitutes necessary conditions for optimality. However, the *Hamiltonian-Jacobi equation* is another statement of the *PMP* (Appendix II). Rumacher applied a partial differential equation (PDE) of *Hamiltonian-Jacobi* type in order to be neurodynamics [2], whereas we employ *PMP* to find important characteristics of the optimal solution. It yields a necessary condition that an optimal control must satisfy, and has become the basic method for computing optimal controls [10]. In a sense, it can also be thought of as an extension of the method of Lagrange multipliers to dynamic optimisation and control problems. Firstly, we attempt to apply the quadratic cost function ψ as the learning potential function, proposed by Amari [16] for ANNs.

4.1 Squared error function

Let $\psi(\mathbf{X},\ \mathbf{W})$ be a loss function. When signal $\mathbf{X}$ is processed by a layered net of feedforward connections, we generally apply the following squared error as the cost function [16]:

$$\psi \equiv \mathrm{E} = \frac{1}{2\mathrm{P}}\sum_{\mathrm{p}=1}^{\mathrm{P}} \| \mathbf{Y}_{\mathrm{p}} - \mathbf{T}_{\mathrm{P}} \| \tag{17}$$

$\mathbf{Y}_{\mathrm{p}}$ is the value of the output vector, and $\mathbf{T}_{\mathrm{p}}$ the target vector.

If we regard the steepest gradient method as weight equations with (17), we have the standard backpropagation learning algorithm as will be shown later.

4.2 The Learning potential function in Continuous-time and Discrete time systems

We will define the following quadratic cost function as the potential function [13][17], in order to apply to optimal control problems, namely quadratic linear problems in the PMP to ANNs. It will be noticed that the potential function has already been defined, as $\psi(\mathbf{X},\ \mathbf{W})$ [16].

In the continuous-time case, the learning potential is

$$\psi^{\mathrm{C}} = \frac{1}{2}[\mathbf{X}^{\mathrm{T}}(\mathrm{t})Q\mathbf{X}(\mathrm{t}) + \mathbf{W}^{\mathrm{T}}(\mathrm{t})\Re\mathbf{W}(\mathrm{t})] \tag{18}$$

subject to (7):

$$\dot{\mathbf{X}}(\mathrm{t}) = \mathrm{f}(\mathbf{X}(\mathrm{t})) + \mathbf{W}(\mathrm{t})\mathrm{g}(\mathbf{X}(\mathrm{t}))$$

Similarly, in the discrete-time case,

$$\psi^{\mathrm{D}} = \frac{1}{2}[\mathbf{X}^{\mathrm{T}}(\mathrm{k})Q\mathbf{X}(\mathrm{k}) + \mathbf{W}^{\mathrm{T}}(\mathrm{k})\Re\mathbf{W}(\mathrm{k})] \tag{19}$$

subject to (9):

$$\mathbf{X}(\mathrm{k}+1) = \mathrm{f}(\mathbf{X}(\mathrm{k})) + \mathbf{W}(\mathrm{k})\mathrm{g}(\mathbf{X}(\mathrm{k}))$$

where, $\Re$ is a real symmetric positive definite matrix, and Q is a real symmetric positive semi-definite matrix.

4.3 PMP necessary conditions and weight equations

In continuous-time systems, the necessary conditions for optimal weights are

$$\dot{\mathbf{X}}(\mathrm{t}) = \frac{\partial \mathrm{H}(\mathbf{X}(\mathrm{t}),\ \mathbf{W}^{\mathrm{OPT}}(\mathrm{t}),\ \lambda(\mathrm{t}))}{\partial \mathbf{X}(\mathrm{t})} \tag{20}$$

$$\dot{\lambda}(\mathrm{t}) = -\frac{\partial \mathrm{H}(\mathbf{X}(\mathrm{t}),\ \mathbf{W}^{\mathrm{OPT}}(\mathrm{t}),\ \lambda(\mathrm{t}))}{\partial \mathbf{X}(\mathrm{t})} \tag{21}$$

$$\mathrm{H}(\mathbf{X}(\mathrm{t}), \mathbf{W}^{\mathrm{OPT}}(\mathrm{t}), \lambda(\mathrm{t})) \le \mathrm{H}(\mathbf{X}(\mathrm{t}), \mathbf{W}(\mathrm{t}), \lambda(\mathrm{t})) \quad \forall \mathrm{t} \in (0, \mathrm{t_f}) \tag{22}$$

It should be noted that $\mathbf{W}^{\mathrm{OPT}}(\mathrm{t})$ in (22) is a control that causes $\mathrm{H}(\mathbf{X}(\mathrm{t}), \mathbf{W}(\mathrm{t}), \lambda(\mathrm{t}))$ to assume its global minimum(maximum) by Pontryagin's principle [9].

In addition, the following equation corresponds to the boundary condition:

$$\lambda(\mathrm{t_f}) = \frac{\partial \mathrm{h}(\mathbf{X}(\mathrm{t_f}))}{\partial \mathbf{X}(\mathrm{t_f})} \tag{23}$$

For $\mathbf{W}^{\mathrm{OPT}}(\mathrm{t})$ to minimize the Hamiltonian it is necessary (but not sufficient) that

$$\frac{\partial \mathrm{H}(\mathbf{X}(\mathrm{t}), \mathbf{W}^{\mathrm{OPT}}(\mathrm{t}), \lambda(\mathrm{t}))}{\partial W(\mathrm{t})} = 0 \tag{24}$$

Notice that equations (21) and (24) are well known Euler-Lagurange equations, giving the necessary condition for optimal weights to minimize performance index J.

If (24) is satisfied, and the matrix:

$$\frac{\partial^2 \mathrm{H}(\mathbf{X}(\mathrm{t}), \mathbf{W}^{\mathrm{OPT}}(\mathrm{t}), \lambda(\mathrm{t}))}{\partial W^2(\mathrm{t})} (\ge 0) \tag{24'}$$

is positive definite, this is sufficient to guarantee that $\mathbf{W}^{\mathrm{OPT}}(\mathrm{t})$ causes the Hamiltonian to become a local minimum [13]. Moreover, (24)′ is known as the Legendre-Clebsch condition [14].

The Hamiltonian can be defined in quadratic form as

$$\mathrm{H}(\mathbf{X}(\mathrm{t}), \mathbf{W}(\mathrm{t}), \lambda(\mathrm{t})) = \lambda^{\mathrm{T}}(\mathrm{t})\dot{\mathbf{X}}(\mathrm{t}) + \frac{1}{2}[\mathbf{X}^{\mathrm{T}}(\mathrm{t})Q\mathbf{X}(\mathrm{t}) + \mathbf{W}^{\mathrm{T}}(\mathrm{t})\Re\mathbf{W}(\mathrm{t})] \tag{25}$$

subject to (7).

Thus the optimal equation for continuous-time ANN systems that we should obtain is

$$\mathbf{W}^{\mathrm{OPT}}(\mathrm{t}) = -\Re^{-1}\mathrm{g}^{\mathrm{T}}(\mathbf{X}(\mathrm{t}))\lambda(\mathrm{t}) \tag{26}$$

which minimizes (globally) the Hamiltonian. This derivation is presented fully in Appendix III.

(26) has a general formulation for Linear-Regulator problems in optimal control theory. In addition, (26) should satisfy the necessary conditions (21) and (23) respectively.

If $\lambda\ (\mathrm{t})$ is chosen to satisfy the boundary condition (23), then

$$\lambda\ (\mathrm{t}) = \frac{\partial \mathrm{h}(\mathbf{X}(\mathrm{t}))}{\partial \mathbf{X}(\mathrm{t})} \tag{27}$$

Thus, (26) can be expressed as follows

$$\mathbf{W}^{\mathrm{OPT}}(\mathrm{t}) = -\Re^{-1}\frac{\partial \mathrm{h}(\mathbf{X}(\mathrm{t}))}{\partial \mathbf{X}(\mathrm{t})}\mathrm{g}(\mathbf{X}(\mathrm{t})) \tag{28}$$

Similarly, the weight-equations in the discrete-time case can be obtained, i.e. in discrete-time systems, the necessary conditions for optimal weights are

$$\mathbf{X}(\mathrm{k}+1) = \frac{\partial \mathrm{H}(\mathbf{X}(\mathrm{k}), \mathbf{W}^{\mathrm{OPT}}(\mathrm{k}), \lambda(\mathrm{k}))}{\partial \lambda(\mathrm{k}+1)} \tag{29}$$

$$\lambda(\mathrm{k}) = \frac{\partial \mathrm{H}(\mathbf{X}(\mathrm{k}), \mathbf{W}^{\mathrm{OPT}}(\mathrm{k}), \lambda(\mathrm{k}))}{\partial \mathbf{X}(\mathrm{k})} \tag{30}$$

$$\mathrm{H}(\mathbf{X}(\mathrm{k}), \mathbf{W}^{\mathrm{OPT}}(\mathrm{k}), \lambda(\mathrm{k})) \le \mathrm{H}(\mathbf{X}(\mathrm{k}), \mathbf{W}(\mathrm{k}), \lambda(\mathrm{k})) \quad \forall \mathrm{t} \in (0, \mathrm{T}) \tag{31}$$

It should be noted that $\mathbf{W}^{\mathrm{OPT}}(\mathrm{k})$ in (31) is a control that causes $\mathrm{H}(\mathbf{X}(\mathrm{k}), \mathbf{W}(\mathrm{k}), \lambda(\mathrm{k}))$ to assume its global minimum(maximum) by Pontryagin's principle [9].

In addition, the following equation corresponds to the boundary condition:

$$\lambda^{\mathrm{T}}(\mathrm{T}) = \frac{\partial \mathrm{h}(\mathbf{X}(\mathrm{T}))}{\partial \mathbf{X}(\mathrm{T})} \tag{32}$$

For $\mathbf{W}^{\mathrm{OPT}}(\mathrm{k})$ to minimize the Hamiltonian it is necessary (but not sufficient) that

$$\frac{\partial \mathrm{H}(\mathbf{X}(\mathrm{k}), \mathbf{W}^{\mathrm{OPT}}(\mathrm{k}), \lambda(\mathrm{k}))}{\partial \mathbf{W}(\mathrm{k})} = 0 \tag{33}$$

If (33) is satisfied, and the matrix:

$$\frac{\partial^2 \mathrm{H}(\mathbf{X}(\mathrm{k}), \mathbf{W}^{\mathrm{OPT}}(\mathrm{k}), \lambda(\mathrm{k}))}{\partial \mathbf{W}^2(\mathrm{k})}(\ge 0) \tag{33)'$$

is positive definite, this is sufficient to guarantee that $\mathbf{W}^{\mathrm{OPT}}(\mathrm{k})$ causes the Hamiltonian to become a local minimum. Moreover, (33)' is known as the Legendre-Clebsch conditions [14].

The Hamiltonian can be defined in quadratic form as

$$\mathrm{H}(\mathbf{X}(\mathrm{k}), \mathbf{W}(\mathrm{k}), \lambda(\mathrm{k})) = \lambda^{\mathrm{T}}(\mathrm{k})\mathbf{X}(\mathrm{k}+1) + \frac{1}{2}[\mathbf{X}^{\mathrm{T}}(\mathrm{k})Q\mathbf{X}(\mathrm{k}) + \mathbf{W}^{\mathrm{T}}(\mathrm{k})\Re\mathbf{W}(\mathrm{k})] \tag{34}$$

subject to (9).

Thus the optimal equation for discrete-time ANN systems that we should obtain is

$$\mathbf{W}^{\mathrm{OPT}}(\mathrm{k}) = -\Re^{-1}\mathrm{g}^{\mathrm{T}}(\mathbf{X}(\mathrm{k}))\lambda(\mathrm{k}) \tag{35}$$

which minimizes (globally) the Hamiltonian.

Equation (35) is a general formulation for Linear-Regulator problems in control theory. In addition, (35) should satisfy the necessary conditions (30) and (32).

If $\lambda\ (\mathrm{k})$ is chosen to satisfy the boundary condition (32), then

$$\lambda\ (\mathrm{k}) = \frac{\partial \mathrm{h}(\mathbf{X}(\mathrm{k}))}{\partial \mathbf{X}(\mathrm{k})} \tag{36}$$

Thus, (36) can be expressed as follows

$$\mathbf{W}^{\mathrm{OPT}}(\mathrm{k}) = -\Re^{-1}\frac{\partial \mathrm{h}(\mathbf{X}(\mathrm{k}+1))}{\partial \mathbf{X}(\mathrm{k}+1)}\mathrm{g}(\mathbf{X}(\mathrm{k})) \tag{37}$$

5. Resulting optimal weight equations and the new learning algorithm

In Section **4**, the optimal equations for weights that we obtained was contrasted with the weight equations obtained by the steepest gradient descent back-propagation algorithm (SBP). The obtained weight equation comprises both the learning algorithm kernel as well as that of back-propagation, because the aim of these learning algorithms is to minimise the quadratic cost function, giving rise to a modification for each connection weight. First of all, the conventional SBP weight equations will be reviewed.

5. 1 Optimal weight equations

The generalized Delta Rule (GDR) for obtaining optimal weights proposed by Rumelhart et al [18], the SBP learning rule, has become very popular since its initial publication (1986). The rule for updating the weights $\mathbf{W}$ is given by the differential equation [18]:

$$\Delta\mathbf{W} = -\eta\frac{\partial \psi(\mathbf{X},\mathbf{W})}{\partial \mathbf{W}} = -\eta\frac{\partial \mathrm{E}(\mathbf{Y},\mathbf{T})}{\partial \mathbf{W}} \tag{38}$$

where, E is the squared error function, and η is a constant (or a positive definite matrix) and has the role of learning rate coefficient (a learning gain term).

Alternatively,

$$\Delta w_{ij} = -\eta \nabla_{W_{ij}} E \qquad (38)'$$

where, $\nabla_{W_{ij}}$ is the partial derivative with respect to w_{ij}, connecting the *i*th node in a layer to the *j*th node in the subsequent layer.

Notice that if we employ (38) into layered feedforward network connections with the squared function $\mathbf{E}$, then this becomes the conventional backpropagation learning algorithm.
This provides us with the insight that $\Re$ in both (26) and (35) will have the following analogous relationships with the learning rate coefficient.

If it is assumed, with respect to both (26) and (35):

$$\mathbf{W}^{\mathrm{opt}} \cong \Delta\mathbf{W},\ \Re^{-1} \cong \mu \tag{39}$$

then we obtain

$$\Delta\mathbf{W} = -\mu \frac{\partial \mathrm{h}(\mathbf{X})}{\partial \mathbf{X}} \mathrm{g}(\mathbf{X}) \tag{40}$$

which can represent as follows

$$\Delta w_{ij} = -\mu \nabla_{x_i} h(\mathbf{x}) g(x_j) \tag{41}$$

where μ is a positive definite matrix and corresponds to the learning coefficient in (38) and the n state variables $\mathbf{x} = [x_1, \cdots, x_n]^{\mathrm{T}}$.

5.2 The obtained learning equation

An early learning equation was qualitatively formulated by Hebb in 1949, and constitutes the basic mechanism of neural learning [19]. This equation may be quantified simply using the formula:

$$w_{ji}(k) = w_{ij}(k-1) + \Delta w_{ij} \tag{42}$$

As with the previous work by Rumelhart et al [18], the following learning rule is obtained:

$$w_{ij}(k) = w_{ij}(k-1) - \eta \nabla_{W_{ij}} E \tag{43}$$

Thus, we obtain the following new learning equation from (41) and (43):

$$w_{ij}(k) = w_{ij}(k-1) - \mu \nabla_{x_i} h(\mathbf{x}) g(x_j) \tag{44}$$

Which can represent as follows

$$w_{ij}(k) = w_{ij}(k-1) - \mu \frac{\partial h(\mathbf{x})}{\partial x_i} g(x_j) \tag{45}$$

Notice that the learning equation is hence a stochastic difference or differential equation. In such a case, this type of learning is an Ordinary Differential Equation (ODE).

The objective function $h(\mathbf{x})$ is to find $\mathbf{x}^*$ so that

$$h(\mathbf{x}^*) = \min_{\mathbf{x}} h(\mathbf{x})$$

where $h(\mathbf{x})$ has continuous second order derivatives (or twice-continuously differentiable), so the object can have a gradient vector [20]:

$$\nabla_{x_i} h(\mathbf{x}) = [\frac{\partial}{\partial x_1} h(\mathbf{x}), \cdots, \frac{\partial}{\partial x_n} h(\mathbf{x})]^{\mathrm{T}} \tag{46}$$

In addition, the gradient vector has the best rate of convergence with the following matrix, called the Hessian or Jacobian matrix:

$$\nabla^2{}_{x_i} h(\mathbf{x}) = \begin{bmatrix} \frac{\partial^2}{\partial^2 x_1} h(\mathbf{x}) \cdots \frac{\partial^2}{\partial x_1 \partial x_n} h(\mathbf{x}) \\ \frac{\partial^2}{\partial x_n \partial x_1} h(\mathbf{x}) \cdots \frac{\partial^2}{\partial^2 x_n} h(\mathbf{x}) \end{bmatrix} \tag{47}$$

Notice that second order deriatives are used in the numerical determination of the optimal [20].

6. Numerical example and performance evaluations

In this section, the performance of the PMP learning algorithm is evaluated by comparison with classical SBP by ways of a quantitative implementation using the C-language on a Sun SPARC station. In order to realise this, we need first to define the form for the continuous second-order derivatives $\mathrm{h}(\cdot)$. Generally speaking, for second-order derivatives in ANNs, Newton's method using the Hessian matrix is well known as the basic local method [21]. It is also far superior with respect to learning time (iterations) [8]. This numerical method might be able to provide benefits by using the PMP learning method with some remarks oriented towards further neurodynamics research.

6.1 The form of second order derivative $\mathrm{h}(\cdot)$

Consider the objective function $\mathrm{h}(\cdot)$ having the following quadratic form:

$$\mathrm{h}(\mathbf{X}) \equiv \frac{1}{2}\mathbf{X}^{\mathrm{T}}\mathrm{C}\mathbf{X} \tag{48}$$

The description of the $l-th$ layer number will be

$$h^l(\mathbf{x}) = \frac{1}{2}\sum_{i=1}^{n}\sum_{k=1}^{n} C^l{}_{ik} x^l{}_i x^l{}_k \qquad\qquad l = 1,2,\ldots,+\infty \tag{49}$$

where, a diagonal symmetric matrix $\mathrm{C} = (C_{ik}),\ C_{ik} = C_{ki}{:}i,k = 1,2,\ldots,n$ can be either positive or negative definite (indefinite). It comprises both the gradient vector $\nabla\mathrm{h}(\mathbf{X}) = \mathrm{C}\mathbf{X}$ and the constant (diagonal) Hessian matrix $\nabla^2\mathrm{h}(\mathbf{X}) = \mathrm{C}$, based on the availability of eigenvalues and eigenvectors. Sufficient conditions for a strong local minimum to the point $\mathbf{X}^*$ are that $|\mathrm{h}(\mathbf{X}^*)| = 0$. Notice that if the Hessian matrix is positive definite and the quadratic model is correct, *one* iteration only is sufficient to reach the minimum. However, even if the Hessian is not positive definite , it can be shown that the quadratic form can be effective on such minimization problems [21].

6.2 The application of the continuous first order derivative $\mathrm{g}(\cdot)$

We apply the objective $\mathrm{g}(\cdot)$ as the sigmoid type function nonlinearity [22][23][24]:

$$\mathrm{g}(\mathbf{X}) = \frac{1}{1+\exp(-\mathbf{X})} \tag{50}$$

which varies smoothly from 0 at $-\infty$ to 1 at ∞ . Moreover, the nonlinearity (50) generally provides two computationally useful properties, in contrast to linear systems such as $\mathrm{g}(\mathbf{X}) = \mathbf{X}$ [25].

One of these properties is a guarantee that the node outputs remain within a bounded range (0,1), that is, the node activation in the non-linear model always has to be within that range. The other is the ability to approximate a very general class of system. It can also be shown that a feedforward neural network model can approximate essentially *any* mapping, given enough hidden nodes [26].

6.3 Numerical results

In order to evaluate the performance of the proposed learning algorithm, we compare it with the classic SBP algorithm for solving the XOR problem. The network architecture has three layers as shown in Figure 1. Each input node in Figure 1 has a signal strength or activation of the sigmoid type function already described in Section **6.2**. For the current implementation, it consists of eleven input nodes, two hidden nodes and one output node. The network is also fully connected with weights initialized randomly prior to learning. The value of any initial weight w is such that $0.3 \leq |w|$. In addition, the learning coefficient for the two methods is $\eta = 0.5,\ \mu = 0.0005$ respectively.

As for the XOR problem, two binary (i.e., false or true) inputs are presented to the network, and the output is trained to reproduce the logical *exclusive-or* truth table.

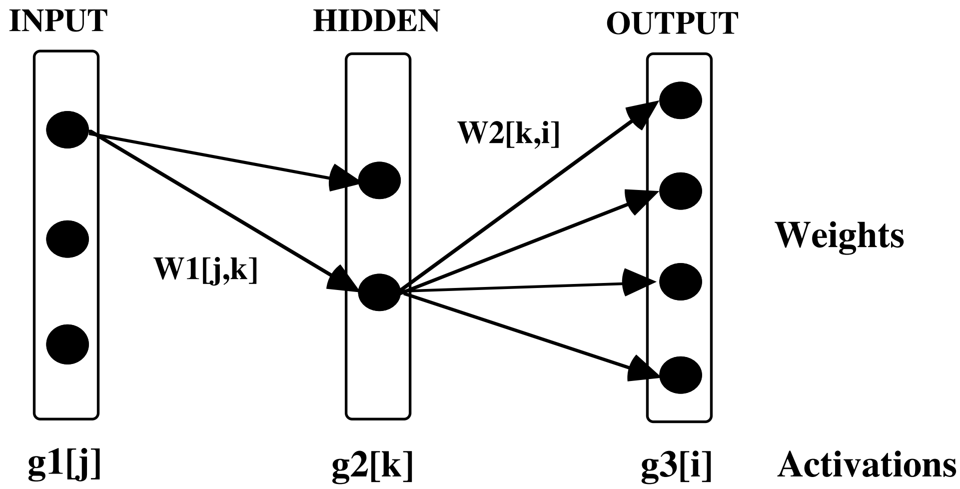

Figure 1: Three-Layer Back Propagation with Feed-forward network (3-2-4)

The performance of the proposed algorithm depend on the choice of the constant Hessian matrix $\mathrm{C} = (C_{ik})$ previously described in Section **6.1**, which is adjusted in order to obtain the best performance for this task . We here apply the following simple Hessian matrix:

$$C^l{}_{ik} = |\kappa| \times \delta^l{}_{ik} \qquad \kappa\text{: constants}$$

where, the rule $\delta^l{}_{ik}$ take the following form:

$$\delta^l{}_{ik} \equiv \begin{cases} 1 & (i = k) \\ 0 & (i \neq k) \end{cases} \qquad i,k = 1,2,\ldots,n$$

A recommended strategy for changing the rule Hessian in order to reach the minimum is, in fact, that of summing to it a simple diagonal matrix of the form $|\kappa| I$ (I being the identity matrix) as above [21]. Effective use of the optinum quadratic forms of Hessian have been analyzed by [27][28].

The results are shown in Figure 2. Convergence to a tolerance of 0.001 took 640 and 1012 iterations for the new PMP learning algorithm and classic SBP respectively. It clearly shows that traditional SBP requires more than 1.5 times as many iterations as the new method.

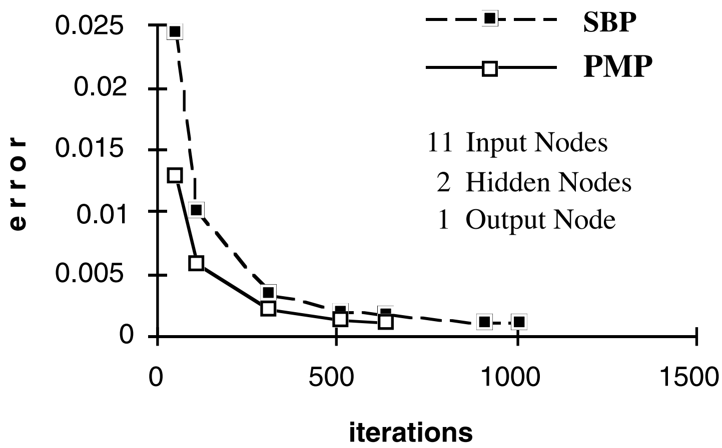

Figure 2: Comparison of the new algorithm and the SBP $(\kappa = 1.0 \times 10^4)$

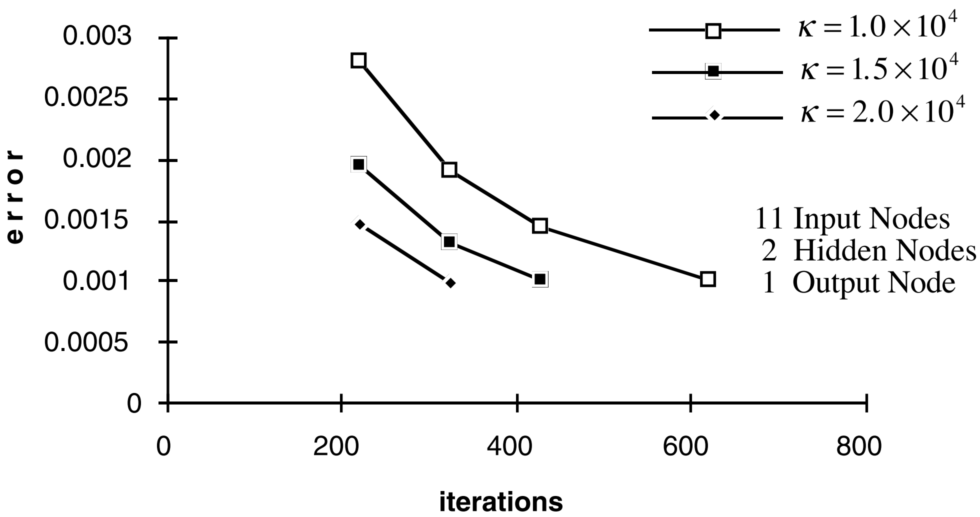

Figure 3: Relationships of error and iterations as changing the constants (κ)

We next changed the positive constants κ of the Hessian matrix (Figure 3) . We see that larger values of κ result in faster convergence for an error tolerance of 0.001.

Moreover, the results in Figure 4 and table 1 are based on training using the same rule $\delta^l{}_{ik}(i,k = 1,2,\ldots,n)$, but different constants κ , as shown below. In particular, notice that for $\kappa = 10 \times 10^4$ the network converges 220 times faster than for $\kappa = 1.5 \times 10^4$.Thus as κ increases, the number of iterations to convergence decreases exponentially.

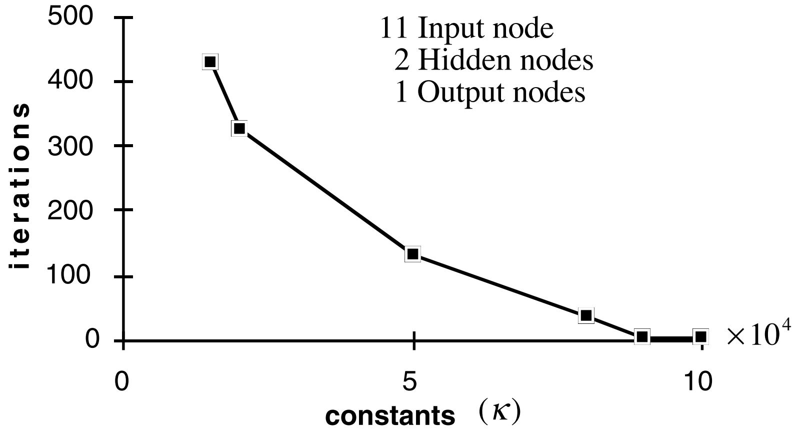

Figure 4: Description of iterations as changing constants (κ), Iteration to convergence (0.001)

Table 1

Constants (κ)	Iterations
15000	429
20000	324
50000	131
80000	37
90000	2
100000	2

Table 1 shows the number of iterations at convergence (0.001) when the constants (κ) are applied for training, based on Figure 4.

7. Summary and Further work

We have shown that a method based on classical optimal control theory, notably PMP, is suited to ANN Hamiltonian systems. In Section **3**, we showed that neurons and weights, being control variables in the network, are able to obey a *continuous-time* differential equation, generally called the state equations. Moreover, the state equations for recurrent (feedback) network models , could be linearly approximated to *discrete-time* feedforward networks because *continuous-time* state equations can be extended to *discrete-time* systems in an optimal control problem. As a result, the ANN system equations, in both continuous-time and discrete-time, can yield optimal weights equations, using the new PMP learning method. In Section **4**, we showed that it is possible to describe the learning potential function as cost functions in the ANN Hamiltonian system. However, developing criteria for selecting appropriate learning potential functions is left for further research. Finally, the new learning algorithm was numerically compared with the traditional SBP algorithm. As a result, we can say the new learning method achieves excellent results. This result also suggests that we could obtain better rules for obtaining optimal network topology or weight dynamics attributed to learning. Nevertheless the quadratic rule that we have already demonstrated leads to several successful results for some networks, other forms for the second order derivatives that have been discussed in Section **6.1** might be needed in further examination of the new PMP learning algorithm. Moreover, we expect applications using Hamiltonian systems to lead to further useful results.

APPENDIX I
Hamiltonian control system and PMP

Here we show how classical optimal control theory such as the PMP is organized under Hamiltonian (*affine*) control systems in *"mechanical nonlinear control system"* [11][29]. In short, the well known Euler-Lagulangian or Hamiltonian equations of motion have been described in terms of "*Classical mechanics*" by [30][31], but here will be shown can be elegantly defined in terms of "*mechanical nonlinear control systems*". Firstly, we review the Euler-Lagrangian and Hamiltonian equations of motion from "*Classical mechanics*". Then, we extend this control theory into Hamiltonian (*affine*) control systems.

Consider a mechanical system with n degrees of freedom, the following equations of motion are derived with n generalized configuration (position) coordinates $x=(x_1,x_2,\ldots,x_n)$:

$$\frac{\mathrm{d}}{\mathrm{dt}}\left[\frac{\partial \mathrm{L}_0}{\partial \dot{x}_i}\right]-\frac{\partial \mathrm{L}_0}{\partial x_i}=0 \qquad i\in n \tag{51}$$

Here, $\mathrm{L}_0=\mathrm{L}_0(x_i,\dot{x}_i)$. (51) is defined as the Lagrangian equation of motion, which are a set of n second order differential equations with $\dot{x}_i=(\dot{x}_1,\dot{x}_2,\ldots,\dot{x}_n)$ being the generalized velocities. Moreover, the Hamiltonian equations of motion in a mechanical system[30] are given by

$$\dot{x}_i=\frac{\partial \mathrm{H}_0}{\partial \lambda_i} \tag{52.a}$$

$$i\in n$$

$$\dot{\lambda}_i=-\frac{\partial \mathrm{H}_0}{\partial x_i} \tag{52.b}$$

$\mathrm{H}_0=\mathrm{H}_0(x_i,\lambda_i)$ is defined as the Hamiltonian or canonical equation of motion, with $\lambda=(\lambda_1,\lambda_2,\ldots,\lambda_n)$ the generalized momenta.

We see that the Euler-Lagrangian or Hamiltonian equations in general mechanics as is Section **7.1** can be extended to control systems, more precisely, mechanical nonlinear control systems [11].

Consider a mechanical control system with n degrees of freedom with a Lagrangian $\mathrm{L}(x,\dot{x},w)$ depending directly on w, being the controls , then the euqations of motion:

$$\frac{\mathrm{d}}{\mathrm{dt}}\left[\frac{\partial \mathrm{L}(x,\dot{x},w)}{\partial \dot{x}_i}\right]-\frac{\partial \mathrm{L}(x,\dot{x},w)}{\partial x_i}=0 \qquad i\in n \tag{53}$$

subject to

$$\mathrm{L}(x,\dot{x},w)=\mathrm{L}_0(x,\dot{x})+\sum_{j=1}^{n}x_j w_j \tag{54}$$

We call (53) a Langrangian control system. In addition, defining $\mathrm{T}(x,\dot{x})$ as the kinetic energy of the system and $\mathrm{V}(x)$ being the potential energy function, we may obtain the $\mathrm{L}_0(x,\dot{x})$ as follows[30]

$$\mathrm{L}_0(x,\dot{x}) = \mathrm{T}(x,\dot{x}) - \mathrm{V}(x) \tag{55}$$

As for the Lagrangian control system of (53), we define the generalized momenta as follows

$$\lambda_i = \frac{\partial \mathrm{L}(x,\dot{x},w)}{\partial \dot{x}} \qquad i \in n \tag{56}$$

so that the Hamiltonian function $\mathrm{H}(x,\lambda,w)$ can be defined as the Legendre transform of $\mathrm{L}(x,\dot{x},w)$:

$$\mathrm{H}(x,\lambda,w) = \sum_{i=1}^{n} \lambda_i \dot{x}_i - \mathrm{L}(x,\dot{x},w) \tag{57}$$

where $\dot{x}$ and λ are related by (56). Moreover, it is well-known that with (56) and (57) the Euler-Lagrange equations of motion:

$$\begin{aligned} \dot{x}_i &= \frac{\partial \mathrm{H}(x,\lambda,w)}{\partial \lambda_i} \\ & \qquad\qquad\qquad i \in n \\ \dot{\lambda}_i &= -\frac{\partial \mathrm{H}(x,\lambda,w)}{\partial x_i} \end{aligned} \tag{58}$$

We call (58) a Hamiltonian control system. We now define the natural outputs of the Hamiltonian control system, as being

$$\mathrm{y}_j = -\frac{\partial \mathrm{H}(x,\lambda,w)}{\partial w_j} \qquad j \in n \tag{59}$$

In particular, if we confine ourselves to Hamiltonian (*affine*) control systems, equation (58) is defined with the following Hamiltonian form:

$$\mathrm{H}(x,\lambda,w) = \mathrm{H}_0(x,\lambda) - \sum_{j=1}^{n} \mathrm{H}_j(x,\lambda) w_j \tag{60}$$

Then we obtain the Hamiltonian (*affine*) control system as follows [11]:

$$\dot{x}_i = \frac{\partial \mathrm{H}_0(x,\lambda)}{\partial \lambda_i} - \sum_{j=1}^{n} \frac{\partial \mathrm{H}_j(x,\lambda) w_j}{\partial \lambda_i}$$

$$i \in n \tag{61}$$

$$\dot{\lambda}_i = -\frac{\partial \mathrm{H}_0(x,\lambda)}{\partial x_i} - \sum_{j=1}^{n} \frac{\partial \mathrm{H}_j(x,\lambda) w_j}{\partial x_i}$$

$$y_j = \mathrm{H}_j(x,\lambda) \qquad j \in n$$

where, defining the Hamiltonain function H_0 as the internal Hamiltonian in [30].

$$\text{Let} \quad \frac{\mathrm{dH}_0(x,\lambda)}{\mathrm{dt}} \tag{62}$$

Thus, we will obtain the following result from (60) and (61):

$$\begin{aligned}
\frac{\mathrm{dH}_0}{\mathrm{dt}} &= \sum_i \left(\frac{\partial \mathrm{H}_0}{\partial x_i} \dot{x}_i + \frac{\partial \mathrm{H}_0}{\partial \lambda_i} \dot{\lambda}_i \right) \\
&= \sum_i \left(\frac{\partial \mathrm{H}_0}{\partial x_i} \frac{\partial \mathrm{H}_0}{\partial \lambda_i} - \frac{\partial \mathrm{H}_0}{\partial \lambda_i} \frac{\partial \mathrm{H}_0}{\partial x_i} \right) + \sum_{i,j} \left(-\frac{\partial \mathrm{H}_0}{\partial x_i} \frac{\partial \mathrm{H}_j}{\partial \lambda_i} + \frac{\partial \mathrm{H}_0}{\partial \lambda_i} \frac{\partial \mathrm{H}_j}{\partial x_i} \right) w_j \\
&= \sum_{i,j} \left\{ \frac{\partial \mathrm{H}_j}{\partial \lambda_i} \left(-\frac{\partial \mathrm{H}_0}{\partial x_i} + \sum_k w_k \frac{\partial \mathrm{H}_k}{\partial x_i} \right) + \frac{\partial \mathrm{H}_j}{\partial x_i} \left(\frac{\partial \mathrm{H}_0}{\partial \lambda_i} + \sum_k w_k \frac{\partial \mathrm{H}_k}{\partial \lambda_i} \right) \right\} w_j \\
&= \sum_j \frac{\mathrm{dH}_j}{\mathrm{dt}} w_j \\
&= \sum_j \dot{y}_j w_j \qquad\qquad k \in m
\end{aligned} \tag{63}$$

If $w_j = 0 \ (j \in n)$, (63) is

$$\frac{\mathrm{dH}_0(x,\lambda)}{\mathrm{dt}} = 0 \tag{64}$$

Therefore, $\mathrm{H}_0(x,\lambda)$ is a *conserved quantity* , expressing the conservation of total energy, which may be represented as $\mathrm{H}_0(x,\lambda) = \mathrm{T}(x,\lambda) + \mathrm{V}(x) = \textit{total energy}$.

Notice that we will be able to obtain the same result with (24) and (33), giving the necessary conditions for PMP, when (59) is equal to 0 (Zero) with respect to the optimal controls $w^{opt}{}_j \ (j \in n)$.

end.

APPENDIX II
PMP and HAMILTONIAN-JACOBI EQUATION

The Hamitonian-Jacobi equation is defined as follows[14]:

$$-\frac{\partial \mathrm{J}^*}{\partial \mathrm{t}} = \mathrm{H}^*(x(t), \lambda^{\mathrm{T}}(t)) \tag{65}$$

where

$$\mathrm{H}^*(x(t), \lambda^{\mathrm{T}}(t)) = \min_{w} \mathrm{H}(x(t), w(t), \lambda^{\mathrm{T}}(t))$$

$$\lambda^{\mathrm{T}}(t) \equiv \frac{\partial \mathrm{J}^*}{\partial x}$$

$$\mathrm{J} = \mathrm{h}(x(t_f)) + \int_{t_0}^{t_f} \varphi(x(t),\ w(t))dt$$

$$\mathrm{H}(x, \lambda, w) = \varphi(x, w) + \lambda^{\mathrm{T}}(t) f(x, w), \quad \dot{x}(x, w) \equiv f(x, w)$$

$$\mathrm{J}^*(x) \equiv \min_{w} \mathrm{J}(x, w)$$

$$\text{Let} \quad \frac{\mathrm{d}\lambda^{\mathrm{T}}}{\mathrm{dt}}$$

then we have

$$\frac{\mathrm{d}\lambda^{\mathrm{T}}}{\mathrm{dt}} \equiv \frac{\mathrm{d}}{\mathrm{dt}}\left(\frac{\partial \mathrm{J}^*}{\partial x}\right) = \frac{\partial^2 \mathrm{J}^*}{\partial x^2} f + \frac{\partial^2 \mathrm{J}^*}{\partial x \partial t} \tag{66}$$

Partial differentiation of the above Hamiltonian-Jacobi equation with respect to x is

$$\begin{aligned} &\frac{\partial}{\partial x}\left(\frac{\partial \mathrm{J}^*}{\partial \mathrm{t}} + \mathrm{H}^*(x(t), \lambda^{\mathrm{T}}(t))\right) \\ &= \frac{\partial^2 \mathrm{J}^*}{\partial x \partial t} + \frac{\partial}{\partial x}\mathrm{H}^*(x(t), \lambda^{\mathrm{T}}(t)) \\ &= 0 \end{aligned}$$

Thus, we obtain the following equation, considering $w^* = w^*(x(t))$ and (65):

$$\begin{aligned} &\frac{\partial^2 \mathrm{J}^*}{\partial x \partial t} + f^{\mathrm{T}} \frac{\partial^2 \mathrm{J}^*}{\partial x^2} + \frac{\partial \varphi}{\partial x} + \frac{\partial \varphi}{\partial w}\frac{\partial w^*}{\partial x} + \frac{\partial \mathrm{J}^*}{\partial x}\left(\frac{\partial f}{\partial x} + \frac{\partial f}{\partial w}\frac{\partial w^*}{\partial x}\right) \\ &= 0 \end{aligned} \tag{67}$$

The PMP necessary condition (24) has been stated previously as being

$$\frac{\partial \mathrm{H}}{\partial w} = \frac{\partial \varphi}{\partial w} + \frac{\partial \mathrm{J}^*}{\partial x}\frac{\partial f}{\partial w} = 0 \tag{68}$$

Now, the coefficient of $\frac{\partial w^*}{\partial x}$ in (67) vanishes on an optimal path according to (68). Concequently, we obtain the following equation:

$$\frac{\partial^2 \mathrm{J}^*}{\partial x \partial t} + f^{\mathrm{T}} \frac{\partial^2 \mathrm{J}^*}{\partial x^2} + \frac{\partial \varphi}{\partial x} + \frac{\partial \mathrm{J}^*}{\partial x} \frac{\partial f}{\partial x} = 0$$

Thus, (67) will beome, taking into account (66)

$$\frac{\mathrm{d}\lambda^{\mathrm{T}}}{\mathrm{dt}} = -\frac{\partial \varphi}{\partial x} - \lambda^{\mathrm{T}} \frac{\partial f}{\partial x} \tag{69}$$

(68) and (69) describe the Euler-Lagrange equation as has been pointed out in Section **4.3**.

end.

APPENDIX III
PROOF OF EQUATION (26)

From (8),

$$\dot{x}_i(t) = f(x_i(t)) + \sum_{j=1}^{n} w_{ij}(t) g(x_j(t))$$

While, (25) can be represented as follows

$$\mathrm{H}(\cdot) = \sum_{i,j} \left[\lambda_i \dot{x}_i + \frac{1}{2} Q_{ij} x_i x_j \right] + \frac{1}{2} \sum_{\alpha,\beta} R_{\alpha,\beta} W_\alpha W_\beta \qquad (25)'$$

where α and β represent multi-indices:

$$\alpha \equiv i_\alpha j_\alpha, \quad \beta \equiv i_\beta j_\beta$$

Thus, (25)' can be represented from (8) as follows

$$\mathrm{H}(\cdot) = \sum_{i,j} \left[\lambda_i (f(x_i) + W_{ij} g(x_j)) + \frac{1}{2} Q_{ij} x_i x_j \right] + \frac{1}{2} \sum_{\alpha,\beta} R_{\alpha\beta} W_\alpha W_\beta \qquad (25)''$$

$$\text{Let} \quad 0 = \frac{\partial \mathrm{H}(\cdot)}{\partial W_\alpha}$$

$$= \sum_{i,j} (\lambda_i g(x_j) \delta_{ii_\alpha} \delta_{jj_\alpha} + R_{\alpha\beta} W_\beta$$

$$= \lambda_{i\alpha} g(x_{j\alpha}) + R_{\alpha\beta} W_\beta$$

Therefore,

$$0 = \lambda_{i\alpha} g(x_{j\alpha}) + R_{\alpha\beta} W_\beta$$

$$\text{(i.e)} \quad W_\beta(t) = -\sum_\alpha (R_{\beta\alpha}{}^{-1} \lambda_{i\alpha}(t) g(x_{j\alpha}(t))) \qquad (26)'$$

Similarly, equation (35) can be obtained as above.

end.

References

[1] M. Y. Choi and B. A. Huberman, "*Dynamic behaviour of nonlinear networks*" , Physical Review A, 28, pp. 1204-1206,1983

[2] U. Ramacher and P. Nachbar, "*The Hamiltonian Approach to neural networks dynamics*", Proc. IJCNN-91, Singapore, Vol. 3, pp. 1930-1936, Nov. 1991

[3] R. Battiti, "*Accelerated backpropagation learning: Two optimization methods*", Complex Systems, Vol. 3, pp. 331-342, 1989

[4] D. M. Goodman et al., "*Backpropagation Learning for Multi-Layer Feed-Forward Neural Networks Using the Conjugate Gradient Method*", International Journal of Neural Systems, Vol. 2, pp. 291-302, 1991

[5] G. Tesauro and Y. He, S. Ahmad, "*Asymptotic Convergence of Backpropagation*", Neural Computation, pp. 382-391, 1989

[6] Moller, M. F., "*A Scaled Conjugate Gradient Algorithm for Fast Supervised Learning*", Neural Networks, Vol. 6, pp. 525-533, 1991

[7] S. Usui et al., "*New accelerated learning algorithm to reduce the oscillation of weights in multilayered neural networks*", Proc. IJCNN-92, Vol. 1, pp. 914-919, 1992

[8] P. Patrick van der Smagt, "*Minimisation Methods for Training Feedforward Neural Networks*", Neural Networks, Vol. 7, pp. 1-11, 1994

[9] L. S. Pontryagin et al., The Mathematical Theory of Optimal Processes, Trans. by K.N.Trirogoff, Interscience Wiley, New York, 1962

[10] R. W. A and V. Vemuri, "*Neural Networks can be used for open-loop, dynamic control*", International Jounal of Neural Networks, Vol. 3, No. 3, pp. 71-83, September 1992

[11] A. J. van der Schaft et al., Nonlinear Dynamical Control Systems., Springer-Verlag New York Inc., 1990.

[12] F. Pineda, "*Generalization of backpropagation to recurrent neural networks*s", Physical Review Letters, Vol. 19, No. 59, pp. 2229-2232, 1987.

[13] D. E. Kirk, Optimal Control Theory, Englewood Cliffs, 1970.

[14] A. E. Bryson and Y. C. Ho., Applied Optimal Control. , New York: Hemisphere, 1975.

[15] R. B. Perez , "*Optimal parameter tracking and control of nonlinear ststems with time varying parameters*", International Conference on Acoustics, Speech, and Signal Processing, Vol.4 , pp. 2224-2227, 1988.

[16] S. Amari, "*Matematical Theory of Neural Learning,*", New Generation Computing, 8(91), pp. 281-294.

[17] H. Tokumaru et al., "*A Nonlinear Regulator Desigh in the Presence of System Uncertainties Using Multilayered Neural Networks*", IEEE Tansactions on Neural Networks. Vol. 2, No. 4, pp. 410-417, July 1991.

[18] D. E. Rummelhart et al., "*Learning internal representations by error propagation, in Parallel Distributed Processing: Explorations in the microstructures of cognition*", Cambridge, MA: MIT press, vol. 1, pp. 318-362, 1986.

[19] E. Davalo et al., Neural Networks, London: Macmillan Education, 1991.

[20] D G. Luenberger, Linear and Nonlinear Programming, 1984.

[21] R. Battiti, "*First- and Second-Order Methods for Learning: Between Steepest Descent and Newton's Method*", Neural Computation, Vol. 4, pp. 141-166, 1992.

[22] M. E. Savran, "*On the Desigh of Dynamic Associate Neural Memories*", IEEE Transaction on Neural Networks. Vol. 5, No. 3, pp. 489-492, May 1994.

[23] M. W. Hirsch, "*Saturation at High Gain in Discrete Time Recurrent Networks*", Neural Networks, Vol. 7, No. 3, pp. 449-453, 1994.

[24] L. G. Sotelino et al., "*Classification of Temporal Trajectories by Continuous-Time Recurrent Nets*", Neural Networks, Vol. 7, No. 5, pp. 767-776, 1994.

[25] R. Rohwer, "*The Time Dimension of Neural Network Models*", SIGART Bulletin, Vol. 5, No. 3, pp. 36-44, July 1994 .

[26] K. Funahashi. "*On the approximate realization of continuous mappings by neural networks*", Neural networks, 2:183, 1989.

[27] J. E. Dennis et al., *"Numerical Methods for Uncontrained Optimization and Nonlinear Equations"*, Prentice Hall, Englewood Cliffs, NJ, 1983.

[28] P. E. Gill et al., Practical Optimization, Academic Press, London, 1981.
[29] A. J. van der Schaft et al., Variational and Hamiltonian Control Systems, Springer-Verlag, , New York, 1987.
[30] H. Goldstein, Classical Mechanics, Addison-Wesley, Reading, Mass., 1950.
[31] E. T. Whittaker, A treatise on the analytical dynamics of particles and rigid bodies, 4th edition, Cambridge University Press, Cambridge, 1959.

(OCT, 1994)