Journal of Global Optimization **28:** 175–195, 2004.
© 2004 *Kluwer Academic Publishers. Printed in the Netherlands.*

Neurodynamical Optimization

LI-ZHI LIAO[1], HOUDUO QI[2] and LIQUN QI[2]
[1]*Department of Mathematics, Hong Kong Baptist University, Kowloon Tong, Kowloon, Hong Kong; E-mail: liliao@hkbu.edu.hk.* [2]*Department of Applied Mathematics, The Hong Kong Polytechnic University, Hung Hom, Kowloon, Hong Kong; E-mail: mahdqi@polyu.edu.hk*

(Received and accepted in revised form 21 January 2003)

Abstract. Dynamical (or ode) system and neural network approaches for optimization have been co-existed for two decades. The main feature of the two approaches is that a continuous path starting from the initial point can be generated and eventually the path will converge to the solution. This feature is quite different from conventional optimization methods where a sequence of points, or a discrete path, is generated. Even dynamical system and neural network approaches share many common features and structures, yet a complete comparison for the two approaches has not been available. In this paper, based on a detailed study on the two approaches, a new approach, termed **neurodynamical approach**, is introduced. The new neurodynamical approach combines the attractive features in both dynamical (or ode) system and neural network approaches. In addition, the new approach suggests a systematic procedure and framework on how to construct a neurodynamical system for both unconstrained and constrained problems. In analyzing the stability issues of the underlying dynamical (or ode) system, the neurodynamical approach adopts a new strategy, which avoids the Lyapunov function. Under the framework of this neurodynamical approach, strong theoretical results as well as promising numerical results are obtained.

Key words: Dynamical system, Neural network, Neurodynamical, Ode System, Optimization

1. Introduction

In this paper, we are interested in finding a local optimum of the following generally constrained optimization problem:

$$\min_{x\in R^n} f(x) \tag{1a}$$

$$s.t. g(x)\leqslant 0, \tag{1b}$$

$$h(x)=0, \tag{1c}$$

where functions $f:R^n\to R^1$, $g:R^n\to R^m$, and $h:R^n\to R^p$ have continuous first order derivatives.

It is very important to observe that the optimization problem (1) itself is posted in the continuous form, i.e., x can be changed continuously. In the literature, the necessary and sufficient conditions of a local optimum are also presented in the continuous form. Furthermore, almost all the theoretical study for problem (1) is in the continuous form. However, it is very interesting to say that when it comes down to the numerical solution of (1), most of the conventional numerical methods, such as the gradient method, Newton's method and quasi-Newton's methods, are addressed in the discrete form. This interesting situation is mainly due to the fact that the computer's computation can be only done discretely. But is it possible to study both the optimization problem and the solution methods in its original form, i.e., continuous form? Can strong and attractive theoretical results be obtained for these new solution methods? How would numerical solutions be obtained in this way? These questions will be addressed, at least partially, in this paper. Further and extensive research is much needed to better understand the continuous solution schemes for (1). It should be noted that we are not the first group to think about these questions. In the literature, there are many papers that have addressed these questions, partially or in some way. These papers can be classified into the following two categories:

- Dynamical system approach. The research in this approach started in 1950s. In the literature, this approach bears many different names, such as gradient process [4], differential equation or ode method [1, 2, 6, 12, 13, 14, 62, 26, 47, 48, 68, 76, 49, 74, 75], curvilinear search method [7, 8, 9, 10], continuous method [16, 22, 58], dynamic method [55, 16], trajectory-following method [61, 52], and gradient-flow method [60], etc. Here we unify these names as dynamical system approach. The essence of this approach is to convert problem (1) into a dynamical system or an ordinary differential equation so that the solution of problem (1) corresponds to a stable equilibrium point of this dynamical system. It is worth mentioning that most of the research in this approach were done by mathematicians. A detailed review on this approach is given in Section 3.
- Neural network approach. The major breakthrough of this approach was due to the seminal work of Hopfield [32, 33] in early 1980s. He introduced an artificial neural network to solve a TSP problem [34]. The mathematical representation of his neural network is an ordinary differential equation which is asymptotically stable at any isolated solution point. A companion of this neural network is an energy function which is a Lyapunov function. He showed that as time evolves, the solution of the ode will converge to the optimum, and in this whole process, the energy function will decrease monotonically in time. The most attractive feature of this approach is that the solution of problem (1) can be obtained on-line in real-time by constructing an electrical circuit which represents an artificial neural network. Following Hopfield's idea, numerous neural networks have been proposed to solve (1) under some convexity assumptions. Interestingly, most of the research in this approach was done by engineers. A detailed review on this approach is given in Section 4.

In each of the two approaches, there are more than 30 research articles and books, however, none of these address, compare and analyze the two approaches in details. In particular, there is no review paper for the dynamical system approach. Motivated by the strong desire to break the wall between the two approaches and combine them into a more powerful approach for (1) based on the merits of the two approaches, this paper attempts to achieve the following three goals:

(i) to review the research results in the dynamical system approach, and identify the merits of this approach;
(ii) to review the research results in the neural network approach, and identify the merits of this approach from the mathematical point of view; and
(iii) to propose a new approach, termed **neurodynamical approach**, which shares all the merits and remove some limitations of the above two approaches.

The rest of the paper is organized as follows. First some preliminary results and definitions are provided in Section 2. Detailed reviews on the dynamical system approach and the neural network approach for optimization problems are presented in Section 3 and Section 4, respectively. Then the new neurodynamical approach is introduced in Section 5. Two neurodynamical systems including both theoretical and simulation results are presented in Section 6. Finally, some concluding remarks including future research directions are addressed in Section 7.

2. Preliminaries

Consider the following simple dynamical system or ordinary differential equation

$$\frac{\mathrm{d}x(t)}{\mathrm{d}t} = \mathrm{d}(x), \tag{2}$$

we first state some classical results on the existence and uniqueness of the solution, and some stability definitions for the dynamical system (2) in [63, 69].

THEOREM 1. [69] *Assume that $d(x)$ is a continuous function from R^n to R^n. Then for arbitrary $t_0 \geqslant 0$ and $x_0 \in R^n$ there exists a local solution $x(t)$ satisfying $x(t_0) = x_0, t \in [t_0, \tau)$ to* (2) *for some $\tau > t_0$. If furthermore $d(x)$ is locally Lipschitz continuous at x_0 then the solution is unique, and if $d(x)$ is Lipschitz continuous in R^n then τ can be extended to ∞.*

DEFINITION 1. (Equilibrium point). *A point $x^* \in R^n$ is called an equilibrium point of* (2) *if $d(x^*) = 0$.*

DEFINITION 2. (Stability in the sense of Lyapunov). *Let $x(t)$ be the solution of* (2). *An isolated equilibrium point x^* is Lyapunov stable if for any $x_0 = x(t_0)$ and any scalar $\varepsilon > 0$ there exists a $\delta > 0$ such that if $\|x(t_0) - x^*\| < \delta$ then $\|x(t) - x^*\| < \varepsilon$ for $t \geqslant t_0$.*

DEFINITION 3. (Convergence). *Let $x(t)$ be the solution of* (2). *An isolated equilibrium point x^* is convergent if there exists a $\delta > 0$ such that if $\|x(t_0) - x^*\| < \delta$, $x(t) \to x^*$ as $t \to \infty$.*

DEFINITION 4. (Asymptotic stability). *An isolated equilibrium point x^* is said to be asymptotically stable if x^* is both Lyapunov stable and convergent.*

DEFINITION 5. (Lyapunov function). [54] *If in a neighborhood of the isolated equilibrium point x^*, there exists a function $V(x)$ which satisfies*: (i) $V(x) > 0$ *if* $x \neq x^*$ *and* $V(x^*) = 0$; (ii) $V(x)$ *has continuous first order partial derivatives*; *and* (iii) *its time derivative along any state trajectory of* (2) *is non-positive, i.e.* $dV/dt \leqslant 0$. *Then* $V(x)$ *is said to be a Lyapunov function for the system* (2) *at equilibrium point* x^*.

Lyapunov function plays a very important role in the stability analysis of dynamical systems and neural network models. Unfortunately, a Lyapunov function may not always exist for any function $d(x)$ in the ode system (2). This limits many applications of dynamical system and/or neural network models. In Section 5, we will show that without using the Lyapunov function, we are still able to conduct the stability analysis under the framework of the neurodynamical approach.

3. Dynamical system approach

The dynamical system approach for the underlying optimization problem is to define a dynamical system or an ordinary differential equation so that the solution of the optimization problem corresponds to a stable equilibrium point of the dynamical system. As a result, it is desired that the solution of this dynamical system forms a continuous path or trajectory which starts at the initial point and ends at the solution of the original optimization problem. The typical forms of such dynamical systems in the literature are as follows:

$$\frac{\mathrm{d}x(t)}{\mathrm{d}t} = \mathrm{d}(x(t)), \tag{3}$$

$$\frac{\mathrm{d}x(t)}{\mathrm{d}t} = s(t) \cdot d(x(t)), \tag{4}$$

$$a(t) \cdot \frac{\mathrm{d}^2 x(t)}{\mathrm{d}t^2} + b(t) \cdot B(x(t)) \cdot \frac{\mathrm{d}x(t)}{\mathrm{d}t} = \mathrm{d}(x(t)), \tag{5}$$

where $d(x)$ is a descent direction for function $f(x)$ in (1a), $B(x) \in R^{n \times n}$ is a positive definite matrix, $a(t)$ and $b(t)$ are scalar functions in t, and $s(t)$ is a positive scalar function in t and bounded above.

The research in the dynamical system approach can be traced back to 1950s, and covers both various types of optimization problems as summarized in Table 1 and different forms of dynamical systems as classified in Table 2. As mentioned in Section 1, the dynamical system approach for optimization is studied mostly by mathematicians. This approach normally consists of the following three steps

(a) to establish an ode system;
(b) to study the convergence of ode solution $x(t)$ as $t \to \infty$; and
(c) to solve the ode system numerically.

Table 1. Classification of articles in terms of problem types

Without constraints	[4], [7], [8], [9], [10], [55], [13], [52]
System of equations	[6], [12], [38], [76], [1], [2], [16], [22], [47]
With equality constraints	[68], [14], [48], [49]
With inequality constraints	[4], [62]
With general constraints	[21], [58], [74], [75]

Table 2. Classification of articles in terms of ode types

Autonomous (3)	Non-autonomous	
	First-order ode (4)	Second-order ode (5)
[4], [6], [12], [7], [21], [8], [9], [10], [58], [68], [76], [16], [13], [14], [26], [22], [47], [48], [49], [74], [75]	[3], [52]	[38], [55], [1], [2], [13]

The establishment of various ode systems in this approach reveals the lack of some physical motivation, i.e., there is no discussion on how such dynamical systems were derived. Even [4] mentioned some economic interpretations, and some ode systems were established based on the classical mechanics [38] and the motion of a particle of unit mass [55], yet most of dynamical systems were derived simply from their discrete counterparts in optimization, such as gradient method, Newton's method, etc.

The convergence study of $x(t)$ as $t \to \infty$ and the stability of the corresponding dynamical system were mostly addressed on a case by case base, no standard theory and/or methodology were given. This phenomenon certainly limits the systematic study of the dynamical system approach and its application potential as well. Two papers are worth mentioning, one by Tanabe [58] which used the stability theory of the dynamical system to study the ode system, and the other one by Yamashita [68] which employed Lyapunov's direct method to study the ode system.

Even though the solutions of ode systems are continuous, yet the actual computation has to be done discretely. In all the dynamical systems (3)–(5), the numerical solutions were mainly solved by either discrete optimization methods or finite difference methods such as Euler, Runge–Kutta methods.

In summary, the main attractiveness of this approach is its simplicity and its originality in pursuing the continuous form. Furthermore, there is no any restrictions on the forms of functions $f(x)$, $g(x)$ and $h(x)$ in (1).

4. Neural network approach

In many engineering and scientific applications, the real-time solutions of optimization problems are demanded. However, traditional algorithms for digital computers may not be able to provide the solutions on-line in real-time. Therefore, the search for real-time on-line solutions in such cases becomes not only important but also essential. In early 1980s, an attractive and very promising approach was introduced to provide real-time on-line solutions for optimization problems. The new approach, which was pioneered by Hopfield [32, 33, 34, 59], is termed as Hopfield neural network or simply neural network. Generally speaking, the neural network approach provides an alternative and attractive way for the solution of optimization problems. The significant and unique feature of the neural network approach to optimization is the realization of simple and real-time hardware implementation. In other words, an electrical circuit can be constructed which generates the on-line solution of certain optimization problems.

Almost all the early work of neural network approach has strong practical application background. In addition, these neural networks are aimed at finding the global minimizer. As a result, the research was mainly focused on linear programming (**LP**), quadratic programming (**QP**) and convex problems as summarized in Table 3. Additional applications can be found in the books by Cichocki and Unbehauen [17] and by Zhang [73].

Table 3. Classification of articles in terms of problem types

LP	[59], [45], [71], [70], [65], [18]
QP	[57], [43], [11], [64], [66]
Convex	[19], [39], [51], [72], [44], [23]

Besides the above types of problems, general optimization problems were also addressed using the neural network approach. The search is no longer for global minimum, but includes local minimum. As a matter of fact, numerous neural network models have been developed for various types of optimization problems. From the optimization point of view, most of existing neural network models for optimization problems could be divided into two classes. One class is gradient-based neural network models, which are used for unconstrained optimization problems. These problems normally come from (a) some kind of transformations from the constrained minimization problem with penalty function methods [39, 51, 44, 15, 42, 23]; and (b) complementarity problems with NCP functions [40, 41]. The other class is projective gradient based neural network models, which are derived from constrained minimization problems and complementarity problems with KKT-conditions [37, 64, 65, 66, 67] and [72]. The neural network models in [64, 65, 66, 67] for solving linear, quadratic and nonlinear convex programming problems are based on KKT systems for optimization problems. Their models correspond to some variants of the projective gradient method for the complementarity problem and the variational inequality problem [24, 28, 29, 31, 50, 56]. It should be noted that for Lagrangian conditions with the nonnegative Lagrange multiplier, if the Lagrange multiplier is penalized or transformed into unrestricted case, the resulting neural network model belongs to the first class. On the other hand, if the nonnegative Lagrange multiplier is enforced by projection, it belongs to the second class.

Generally speaking, the Hopfield neural network is a recurrent neural network (i.e., a neural network with feedback) which is asymptotically stable at any isolated solution point. A Hopfield neural network model consists of the following components:

(a) a Lyapunov function as an energy function;

(b) a neural network which is asymptotically stable at any isolated solution point and is in the form of an ordinary differential equation mathematically; and

(c) a hardware implementation or an architectural neural network diagram with computer simulation.

Different from the dynamical system approach, every Hopfield neural network model must have an energy function, which is one of the most important contributions of Hopfield [32]. In the process of system evolution, this energy function will decrease monotonically and reduce to zero when the system reaches the equilibrium point. This energy function shares the same spirit as the merit function in interior point methods for LP. But here the energy function must be a Lyapunov function.

As mentioned in Section 3 where many dynamical systems normally do not have any physical meaning, the dynamical systems here are all autonomous and simply in the form of (3), where $\mathrm{d}(x(t))$ is normally just the negative gradient of the energy function with respect to x. The physical interpretation of this energy function and the resulting dynamical system can be viewed as the motion of a particle on the energy surface under the influence of gravity. From any initial point, the particle slides downhill until it reaches the bottom of the hill (assume that the particle has little momentum so that it will stay at the local minimum). In this way, every local minimizer of the energy function can be thought as an attractor.

The stability analysis for various neural network models has been addressed in the literature, see [11, 15, 23, 39, 43, 42, 64, 72]. However, all these discussions are based on Lyapunov's direct method which requires the existence of a Lyapunov function. Unfortunately, a Lyapunov function may not always exist. This certainly limits the stability analysis for a general neural network model.

The hardware realization of a neural network ensures that the solution of the underlying optimization problemcan be obtained on-line in real-time. However, only certain functions can be achieved by neural network units due to the hardware limitation. For those problems where nonlinear functions are engaged, certain nonlinear artificial neurons must be required. Table 4 summarizes the basic or typical nonlinear functions, also called activation functions, used in neural networks in the literature. For a more complete list, please see Appendix B in [17].

Table 4. A list of the basic or typical nonlinear functions used in neural networks

Sigmoid function	$\phi(x)=\frac{1-e^{-2\lambda x}}{1+e^{-2\lambda x}}$ or $\phi(x)=\frac{1}{1+e^{-\lambda x}}$
Hard limiter (signum function)	$\phi(x)=\begin{cases}1 & if x\geqslant 0\\ -1 & if x<0\end{cases}$
Saturation limiter	$\phi(x)=\begin{cases}1 & if x\geqslant 1\\ x & if -1<x<1\\ -1 & if x\leqslant -1\end{cases}$
Simple limiter	$\phi(x)=\begin{cases}x & if x\geqslant 0\\ 0 & if x<0\end{cases}$
Quadratic function	$\phi(x)=\begin{cases}x^2 & if x\geqslant 0\\ 0 & if x<0\end{cases}$
Absolute value function	$\phi(x)=\|x\|$
Continuous-time integrator	$y(t)=y(0)+\int_0^t x(\tau)d\tau$

In late 1980s, some significant achievements have been obtained in artificial neural network. Refs. [20] and [36] have shown that continuous functions on compact subset of R^n can be uniformly well approximated by linear combinations of sigmoidal functions. In addition, [35] reported that under very general conditions, networks with sufficiently smooth activation functions are capable of arbitrarily accurate approximation to a function and its derivatives. To further extend the study to a quantitative scale, Barron [5] examined how the approximation error is related to the number of nodes in the network. It is shown in [5] that feedforward networks with one layer of sigmoidal nonlinearities can achieve integrated squared error of order $O(1/N)$, where N is the number of nodes. Certainly, neural network learning, which is becoming a very important part of artificial neural networks, must be employed in these approximations.

Another but very important feature of the neural network approach is that its formulation is naturally suited for massive parallel processing. As a matter of fact, the computation in the neural network approach is normally carried out in parallel. Generally speaking, the number of computational units could be in the same order of the number of unknown variables. Furthermore, this scaling can be done without any additional work.

5. Neurodynamical approach

The term, neurodynamics, is not new. In [27], an entire chapter is devoted to the discussion of neurodynamics. But Lyapunov's direct method was used to address the stability issue. To avoid using the Lyapunov function and combine the merits of both dynamical system and neural network approaches, a new approach, called neurodynamical approach, is introduced in this section to solve problem (1).

Based on the analysis and discussion in previous two sections, we can see that the main merit of the dynamical system approach is that many dynamical systems can be constructed based on optimization methods, i.e., these dynamical systems can be viewed as the continuous realization of many optimization methods. On the other hand, the main mathematical merit of the neural network approach lies in the formulation of the energy function which decreases monotonically in time along the solution of the underlying dynamical system. This energy function enables a systematic study of the stability analysis for dynamical systems.

5.1. FRAMEWORK OF A NEURODYNAMICAL SYSTEM

In the proposing neurodynamical approach, we require each neurodynamical system to have the following features:

Framework of a neurodynamical system

(N1) A merit function, say $V(x)$, which may not be a Lyapunov function but is bounded below, and a dynamical (or ode) system must be constructed.

(N2) The dynamical (or ode) system must be asymptotically stable at any isolated solution point of problem (1).

(N3) $\frac{\mathrm{d}V(x(t))}{\mathrm{d}t} = (\nabla_x V(x(t)))^T \frac{\mathrm{d}x(t)}{\mathrm{d}t} \leqslant 0$ for all $t \geqslant 0$ and $\frac{\mathrm{d}V(x(t))}{\mathrm{d}t} = 0$ if and only if $\frac{\mathrm{d}x(t)}{\mathrm{d}t} = 0$, where $x(t)$ is a solution of the dynamical (or ode) system established in (N1).

(N4) Each equilibrium point of the dynamical (or ode) system must correspond to a (constrained) stationary point of problem (1).

Remarks.

(a) The $V(x)$ and the ode system for each problem are not unique but they must satisfy the above requirements.

(b) The analytical solution $x(t)$ of the ode system is not sought. Only the limit point of $x(t)$ is interested.

(c) The stability analysis in both dynamical system and neural network approaches adopts Lyapunov's direct method, which relies on the Lyapunov function. Here, we remove the restriction of the Lyapunov function, as a result, we are ableto both analyze a much wider range of dynamical systems and simplify the analysis.
(d) In the stability analysis of Lyapunov's direct method, the equilibrium point must be known beforehand. However, in our new neurodynamical system, no Lyapunov function is needed, therefore we do not need any prior information on the equilibrium point. This is very attractive since normally the equilibrium point is the one that we are going to find.
(e) The dynamical (or ode) system is not just limited to any form of (3)–(5). This is because some time-delay dynamical (or ode) systems may also be available as a result of quasi-Newton directions.
(f) The hardware (circuit) implementation is not required in the above features. This is mainly due to the fact that any continuous function on a compact set can be uniformly well approximated by linear combinations of sigmoidal functions as stated in Section 4. Certainly large number of neurons may be required for a very nonlinear function, but this is beyond the scope of this paper.

5.2. GENERAL PROPERTIES

In this subsection, we will explore some general properties for a neurodynamical system. Theorem 1 has guaranteed that if $\mathrm{d}(x)$ is Lipschitz continuous, the solution of (2) exists and is unique. First, let's state the following Barbalat's lemma which is very useful in our stability analysis.

Barbalat's Lemma [54] *If a differentiable function $V(t)$ has a finite limit as $t \to \infty$, and $\frac{dV(t)}{dt}$ is uniformly continuous, then $\frac{dV(t)}{dt} \to 0$ as $t \to \infty$.*

Theorem 2 below states a general property for the ode system (2).

THEOREM 2. *Assume that (i) $V(x) \in C^1(R^n)$ is bounded below, (ii) there exists constants $\beta \geqslant \alpha > 0$ such that $-(\nabla V(x))^T d(x) \geqslant \alpha \|\nabla V(x)\|^2$ and $\|d(x)\| \leqslant \beta \|\nabla V(x)\|$ $\forall x \in R^n$, and (iii) $d(x)$ and $\nabla V(x)$ are Lipschitz continuous in R^n. Then for any $t_0 \geqslant 0$ and $x(t_0) = x_0$, $\nabla V(x(t)) \to 0$. Therefore $d(x(t)) \to 0$ as $t \to \infty$, where $x(t)$ is the solution of* (2).

Proof: Since $d(x)$ is Lipschitz continuous, Theorem 1 ensures that for any $t_0 \geqslant 0$ and $x(t_0) = x_0$, there exists a unique solution to the ode system (2). The assumptions of $V(x)$ being bounded below and $-(\nabla V(x))^T d(x) \geqslant \alpha \|\nabla V(x)\|^2$ imply that $V(x(t))$ is monotonically decreasing along the trajectory $x(t)$ in t if $\nabla V(x(t)) \neq 0$. Therefore, there exists a finite scalar V^* such that $\lim_{t\to\infty} V(x(t)) =$

V^*. Then

$$V(x_0)-V^* = -\int_{t_0}^{\infty} (\nabla V(x(t)))^T \mathrm{d}(x(t))\mathrm{d}t \geqslant \alpha \int_{t_0}^{\infty} \|\nabla V(x(t))\|^2 \mathrm{d}t.$$

Following exactly the same arguments as in the proof of Theorem 4 in [25], we have that there exists a constant L such that

$$\|\nabla V(x(t))\| \leqslant L \quad \forall t \in [t_0, \infty). \tag{6}$$

Then for any $t_1,\ t_2 \in [t_0,\infty)$,

$$\begin{aligned}
\left|\frac{\mathrm{d}V(x(t_1))}{\mathrm{d}t} - \frac{\mathrm{d}V(x(t_2))}{\mathrm{d}t}\right| &= |(\nabla V(x(t_1)))^T \mathrm{d}(x(t_1)) - (\nabla V(x(t_2)))^T \mathrm{d}(x(t_2))| \\
&\leqslant |(\nabla V(x(t_1)))^T \mathrm{d}(x(t_1)) - (\nabla V(x(t_1)))^T \mathrm{d}(x(t_2))| \\
&\quad + |(\nabla V(x(t_1)))^T \mathrm{d}(x(t_2)) - (\nabla V(x(t_2)))^T \mathrm{d}(x(t_2))| \\
&\leqslant L\|\mathrm{d}(x(t_1)) - \mathrm{d}(x(t_2))\| + \beta \cdot L \|\nabla V(x(t_1)) \\
&\quad - \nabla V(x(t_2))\| \\
&\leqslant (L\cdot L_1 + \beta\cdot L\cdot L_2)\|x(t_1)-x(t_2)\| \\
&\leqslant (L\cdot L_1 + \beta\cdot L\cdot L_2)\left\|\int_{t_1}^{t_2} d(x(\tau))d\tau\right\| \\
&\leqslant (L\cdot L_1 + \beta\cdot L\cdot L_2)\beta\cdot L|t_1-t_2|,
\end{aligned} \tag{7}$$

where L_1 and L_2 are the Lipschitz constants for $d(x)$ and $\nabla V(x)$, respectively. So $\frac{dV(x(t))}{dt}$ is uniformly continuous in $[t_0,\infty)$. From Barbalat's lemma and the assumption (ii), we have

$$\lim_{t\to\infty} \nabla V(x(t)) = 0, \qquad \text{therefore} \qquad \lim_{t\to\infty} d(x(t)) = 0. \tag{8}$$

This completes our proof. □

The result of Theorem 2 is quite general. Essentially, it says that if $d(x)$ is a strictly descent direction for $V(x)$, then any limit point of the solution of (2) is a stationary point of $V(x)$ under fairly general assumptions. Theorem 2 is a little short of proving the convergence of $x(t)$. This convergence result is very desirable but it is difficult to obtain. In Section 6, we will state this result in the case of $d(x) = -\nabla V(x)$ for unconstrained problems.

The following Theorem 3 reveals that if $d(x(t_0)) \neq 0$, then $d(x(t)) \neq 0$ for all $t \in [t_0,\infty)$.

THEOREM 3. *Under the same assumptions as Theorem* 2, *we have that if* $d(x(t_0)) \neq 0$, *then* $d(x(t)) \neq 0$ *for all* $t \in [t_0,\infty)$. *Therefore,* $\nabla V(x(t)) \neq 0$ *for all* $t \in [t_0,\infty)$.

Proof. It is easy to see that $d(x(t))$ is continuous in t under the assumptions. Suppose that there exists a $\bar{t} > t_0$ such that $d(x(\bar{t})) = 0$. From the continuity of $d(x(t))$, we can assume that $\bar{t}$ is the smallest t such that $d(x(t)) = 0$.

From the proof of Theorem 2, we know that (6) is true and therefore

$$\|\mathrm{d}(x(t))\| \leqslant \beta \cdot L \quad \forall t \geqslant t_0.$$

Let L_1 be the Lipschitz constant for $d(x)$ and δ be any constant with $\delta \in (0, \frac{1}{2L_1}]$. Now we focus on the interval $[\bar{t}-\delta, \bar{t}]$. Then for any $t_1,\ t_2 \in [\bar{t}-\delta, \bar{t}]$, we have

$$\begin{aligned} \|\mathrm{d}(x(t_1))\| - \|\mathrm{d}(x(t_2))\| &\leqslant L_1 \|x(t_1) - x(t_2)\| \\ &\leqslant L_1 \| \int_{t_1}^{t_2} \mathrm{d}(x(\tau)\mathrm{d}\tau \| \\ &\leqslant L_1 \cdot \delta \max_{\tau \in [\bar{t}-\delta, \bar{t}]} \|\mathrm{d}(x(\tau))\|. \end{aligned} \tag{9}$$

Notice that (9) is true for any $t_1,\ t_2 \in [\bar{t}-\delta, \bar{t}]$ and $\mathrm{d}(x(\bar{t}))=0$, then we have

$$0 = \min_{\tau \in [\bar{t}-\delta, \bar{t}]} \|\mathrm{d}(x(\tau))\| \geqslant (1 - L_1 \cdot \delta) \max_{\tau \in [\bar{t}-\delta, \bar{t}]} \|\mathrm{d}(x(\tau))\|. \tag{10}$$

This implies that $d(x(\tau))=0$ for any $\tau \in [\bar{t}-\delta, \bar{t}]$ which contradicts with the definition of $\bar{t}$. This completes the proof. □

In conventional optimization methods, finite step termination can happen in some solution schemes. But the result of Theorem 3 indicates that finite step termination will not happen in the neurodynamical approach. Consider a very simple function of $f(x) = \frac{1}{2}x^2$ and the ode system $\frac{\mathrm{d}x(t)}{\mathrm{d}t} = -x(t)$. It is easy to get the analytical solution of the ode system $x(t) = x_0 e^{-(t-t_0)}$. Therefore for any $t \geqslant t_0$, $x(t) \neq 0$ if $x_0 \neq 0$.

6. Some neurodynamical systems

In this section, two neurodynamical systems will be presented based on the framework established in Section 5 for both unconstrained and constrained problems.

6.1. A. STEEPEST DESCENT DIRECTION FOR UNCONSTRAINED PROBLEMS

If $V(x) = f(x)$ and $d(x) = -\nabla V(x)$, the right-hand-side of (2) corresponds to the steepest descent direction. This is the most common case. A detailed study of this dynamical system is provided in [25]. The following theorem from [25] guarantees the asymptotic stability of this dynamical system at any isolated solution point.

THEOREM 4 (Theorem 4 in [25].). *If $V(x)$ is bounded below and its gradient $\nabla V(x)$ is Lipschitz continuous in R^n, then for any initial point x_0, the trajectory $x(t)$ of the system (2), satisfying $x(t_0) = x_0$, will converge to an equilibrium point of the neural network (2) as $t \to \infty$.*

The result of Theorem 4 is very strong. It guarantees the convergence of $x(t)$, the solution of (2), as $t \to \infty$. It should be noted that in the above result the infinity point has been viewed as an ordinary point in R^n in such a way that for every d $(\neq 0)$, $x \in R^n$, $x+\alpha d$ approaches to it as $\alpha \to \infty$.

THEOREM 5. *For unconstrained problem* $(1a)$, *if* $f(x)$ *is bounded below and its gradient* $\nabla f(x)$ *is Lipschitz continuous in* R^n, *then the system of* $V(x)=f(x)$ *and* (2) *with* $d(x)=-\nabla V(x)$ *is a neurodynamical system.*

Proof. Since $f(x)$ is bounded below and the ode system (2) is well defined with $d(x)=-\nabla V(x)$, (N1) is satisfied. While Theorem 4 ensures the asymptotic stability of (2) with $\mathrm{d}(x)=-\nabla V(x)$ at any isolated solution point. Therefore (N2) is met.

Finally, due to $\frac{\mathrm{d}V(x)}{\mathrm{d}t}=-\|\nabla V(x)\|^2$ and $V(x)=f(x)$, (N3) and (N4) can be easily verified. This completes the proof. □

6.2. B. NEWTON'S DIRECTION FOR UNCONSTRAINED PROBLEMS

Newton's method is perhaps the most important method in optimization. The study of Newton's method has been quite rich in the literature. However, in the framework of the neurodynamical approach, Newton's direction has surely brought out many new challenges. Let $V(x)=f(x)$, based on Newton's direction, the right-hand vector $d(x)$ in (2) can be defined as

$$\mathrm{d}(x)=\begin{cases} -[\nabla^2 V(x)]^{-1}\nabla V(x) & \text{if } \lambda(\nabla^2 V(x)) \geqslant \delta > 0, \\ -\nabla V(x) & \text{otherwise,} \end{cases} \tag{11}$$

where $\lambda(\cdot)$ denotes the minimum eigenvalue of the underlying matrix. Then two problems emerge:

1. First, the $\mathrm{d}(x)$ in (11) is not continuous. Therefore, the existence of a solution $x(t)$ to (2) needs to be justified.
2. Second, the computation of $\mathrm{d}(x)$ in (11) is much more expensive than the $\mathrm{d}(x)$ in the steepest descent direction. Whether the faster convergence of Newton's direction could compensate this extra computational load remains to be investigated.

To compare the numerical performance of the steepest descent direction and Newton's direction in (11) and also illustrate the numerical implementation of our neurodynamical approach, we apply the system (2) to the following common test examples in the literature.

Example 1. Powell badly scaled function [46]

$$\begin{cases} f(x)=(10^4 x_1 x_2-1)^2+(\exp(-x_1)+\exp(-x_2)-1.0001)^2, \\ x_0=(0,1),\ x^*=(1.098\cdots 10^{-5}, 9.0106\cdots),\ f(x^*)=0. \end{cases}$$

Example 2. Extended Rosenbrock function [46]

$$\begin{cases} f(x)=\sum_{i=1}^{n}\left[100(x_{2i}-x_{2i-1}^2)^2+(1-x_{2i-1})^2\right], \\ [x_0]_{2i-1}=-1.2,\ [x_0]_{2i}=1,\ x^*=(1,\cdots,1)^T,\ f(x^*)=0. \end{cases}$$

In our simulation experiment, we tested $\delta = 10^{-10}$ and 10^{-20} for Newton's direction, and $n = 10$ in Example 2. Our simulation will stop if $\|\nabla f(x)\| \leqslant 10^{-8}$. Our simulation was implemented on a HP workstation (Model J5600) with Matlab 6.0. The ode solver used is *ode23s*. The minimum eigenvalue routine used in (11) is the modified Cholesky factorization scheme in [53]. Our simulation results are summarized in Table 5.

Table 5. Simulation results of Examples 1 and 2

SD	Example 1			Example 2		
	f	$\|\nabla f\|$	CPU (s)	f	$\|\nabla f\|$	CPU (s)
	1.34e-28	1.96e-9	0.5	1.11e-16	9.40e-9	4.3
Newton						
δ=1.e-10	1.34e-28	1.96e-9	1.4	3.27e-18	3.04e-9	30
δ=1.e-20	2.12e-26	7.20e-9	1.4	3.27e-18	3.04e-9	30

The results in Table 6.2 indicate that for the two examples, the matrix operation for Newton's direction really has slowed down the performance of Newton's method. More discussions on the related issues will be addressed in Section 7.

6.3. C. PROBLEMS WITH CONVEX SET CONSTRAINTS

Here we focus on the following form of problems:

$$\min_{x \in R^n} f(x) \tag{12}$$

$$s.t. x \in \Omega, \tag{13}$$

where Ω is a closed and convex set. In particular, the common cases for Ω are: (a) $\Omega = \{x \in R^n |\ a \leqslant x \leqslant b\}$ (bound constraint), (b) $\Omega = \{x \in R^n |\ \|x\|_2 \leqslant c\}$ (ball or l_2 norm constraint), and (c) $\Omega = \{x \in R^n |\ \sum_{i=1}^n |x_i| \leqslant d\}$ (l_1 norm constraint).

To establish a neurodynamical system for (12)–(13), we define $V(x) = f(x)$ and the following dynamical system:

$$\frac{\mathrm{d}x(t)}{\mathrm{d}t} = -[x - P_\Omega(x - \nabla f(x))], \tag{14}$$

where $P_\Omega(\cdot)$ is the projection operator onto Ω. To ease the following discussion, we define

$$e(x) = x - P_\Omega(x - \nabla f(x)). \tag{15}$$

LEMMA 1. *For problem* (12)–(13) *and the ode system* (14), *if* $x(t_0) = x_0 \in \Omega$, *then the solution* $x(t)$ *of* (14) *is always feasible, i.e.* $x(t) \in \Omega \forall t \geqslant t_0$.

Proof. It is sufficient to show that $-e(x)$ is a feasible direction for Ω at x providing $x \in \Omega$.

For any $\varepsilon \geqslant 0$, we define

$$x_\varepsilon = x + \varepsilon(-e(x)).$$

Then

$$x_\varepsilon = (1-\varepsilon)x + P_\Omega[\varepsilon(x - \nabla f(x))]. \tag{16}$$

Since $x,\ P_\Omega(x - \nabla f(x)) \in \Omega$ and Ω is a closed convex set, we know that $(1-\varepsilon)x + \varepsilon P_\Omega(x - \nabla f(x)) \in \Omega\ \ \forall \varepsilon \in [0,1]$. Therefore (16) indicates that $x_\varepsilon \in \Omega\ \ \forall \varepsilon \in [0,1]$. Thus $-e(x)$ is a feasible direction for Ω at x providing $x \in \Omega$. □

The result of Lemma 1 guarantees that the solution $x(t)$ of (14) always stays in Ω. Therefore, in the rest discussion for problem (12)–(13) and the ode system (14), we only concentrate on Ω.

LEMMA 2. *For problem* (12)–(13), *if* $\nabla f(x)$ *is Lipschitz continuous in* Ω, *then* $e(x)$ *is also Lipschitz continuous in* Ω.

Proof. Let L_f be the Lipschitz constant for $\nabla f(x)$. Then from (15), we have

$$\|e(x) - e(y)\| \leqslant \|x-y\| + \|P_\Omega(x - \nabla f(x)) - P_\Omega(y - \nabla f(y))\| \quad \forall x,\ y \in R^n. \tag{17}$$

Since Ω is a closed convex set, from the non-expansive property of the projection operator, we have

$$\|P_\Omega(x) - P_\Omega(y)\| \leqslant \|x-y\| \qquad \forall x,\ y \in R^n. \tag{18}$$

Using (18), (17) becomes

$$\begin{aligned}\|e(x)-e(y)\| &\leqslant \|x-y\| + \|(x-\nabla f(x)) - (y - \nabla f(y))\| \\ &\leqslant 2\|x-y\| + \|\nabla f(x) - \nabla f(y)\| \\ &\leqslant (2+L_f)\|x-y\|.\end{aligned} \tag{19}$$

Therefore $e(x)$ is Lipschitz continuous in Ω. □

Lemma 2 and Theorem 1 guarantee that the ode system (14) has a unique solution $x(t)$ for any initial solution $x_0 \in \Omega$ if $\nabla f(x)$ is Lipschitz continuous in Ω.

LEMMA 3. *For problem* (12)–(13) *and the ode system* (14), *if* $x(t_0) = x_0 \in \Omega$, *then we have*

$$(\nabla f(x))^T e(x) \geqslant \|e(x)\|^2 \geqslant 0. \tag{20}$$

Proof. Since Ω is a closed convex set, from inequality (4) in [30], we have

$$[y-P_{\Omega}(y)]^T[x-P_{\Omega}(y)]\leqslant 0 \quad \forall x\in\Omega,\ \forall y\in R^n. \tag{21}$$

Taking $y=x-\nabla f(x)$ in (21), we have

$$[e(x)-\nabla f(x)]^T e(x)\leqslant 0. \tag{22}$$

This proves (20). □

If x^* is a local minimum point of problem (12)–(13), then the intersection of any feasible direction and any descent direction at x^* is empty. In the case that $f(x)$ is differentiable on an open set which includes Ω, this can be written as

$$x^*\in\Omega, \quad (x-x^*)^T\nabla f(x^*)\geqslant 0 \quad \forall x\in\Omega. \tag{23}$$

The following lemma states an equivalent condition for (23).

LEMMA 4. *For problem* (12)–(13), *x^* satisfies* (23) *if and only if x^* is a zero point of $e(x)$.*

Proof. See the proof of Theorem 1 in [30]. □

It is worth mentioning that in our discussion on the steepest descent direction based neurodynamical system, the level set

$$L(x_0)=\{x\in\Omega|f(x)\leqslant f(x_0)\} \tag{24}$$

could be unbounded. But for problems with constraints, we must require the boundedness of the level set $L(x_0)$. Now we are ready to prove the convergence of $e(x)$.

THEOREM 6. *For problem* (12)–(13) *and the ode system* (14), *if $\nabla f(x)$ is Lipschitz continuous in Ω and $L(x_0)$ is bounded for any $x_0\in\Omega$, then for any initial solution $x(t_0)=x_0\in\Omega$, any limit point of the solution $x(t)$ of the ode system* (14) *is an equilibrium point of* (14), *i.e.*

$$\lim_{t\to\infty} e(x(t))=0. \tag{25}$$

Proof. Since $x(t_0)=x_0\in\Omega$, we know from Lemma 1 that any solution of the ode system (14) will stay in Ω. The Lipschitz continuity of $e(x)$ in Lemma 2 ensures that the solution $x(t)$ of the ode system (14) exists and is unique from Theorem 1. Since $L(x_0)$ is bounded, we know that both $\|\nabla f(x)\|$ and $e(x)$ are bounded in Ω. Following the similar discussions as in the proof of Theorem 2, we can prove that $\frac{\mathrm{d}f(x)}{\mathrm{d}t}$ is uniformly continuous in $[t_0,\infty)$. From Barbalat's lemma, $\frac{\mathrm{d}f(x)}{\mathrm{d}t}=-(\nabla f(x))^T e(x)$, and Lemma 3, we can establish (25). □

The following theorem states that for problem (12)–(13), $V(x)=f(x)$ and the ode system (14) constitute a neurodynamical system.

THEOREM 7. *For problem* (12)–(13), *if its gradient* $\nabla f(x)$ *is Lipschitz continuous in* Ω *and* $L(x_0)$ *is bounded for any* $x_0 \in \Omega$, *then the system of* $V(x)=f(x)$ *and* (14) *is a neurodynamical system.*

Proof. From our assumptions and previous discussions, we know that the ode system (14) is well defined. In addition, From $V(x)=f(x)$, Lemma 1 and $L(x_0)$ being bounded, we can easily see that $f(x)$ is bounded in Ω. Therefore (N1) is true.

Let x^* be any isolated solution point of (12)–(13), then Theorem 6, Lemma 4 and the proof of Theorem 4 in [25] guarantee that x^* is asymptotically stable. Therefore (N2) is met.

Notice that $\frac{\mathrm{d}V(x)}{\mathrm{d}t}=-(\nabla f(x))^T e(x)\leqslant -\|e(x)\|^2\leqslant 0$ (Lemma 3), then (N3) can be established easily.

Finally, Lemma 4 ensures that (N4) is true as well. This completes the proof. □

It should be noted that the assumption of level set $L(x_0)$ being bounded can be removed if Ω is bounded.

7. Concluding remarks

In this paper, based on thorough reviews on both dynamical system and neural network approaches for optimization, a neurodynamical approach is proposed for general optimization problems. The framework set out in Section 5 for the neurodynamical approach combines the merits of the previous two approaches and allows a rigorous and systematic continuous path study for optimization. Using the neurodynamical approach, good and attractive theoretical results have been obtained under very mild assumptions. In addition, two neurodynamical systems are established, one for unconstrained problems and one for convex set constrained problems. These two neurodynamical systems have illustrated many attractive features for the neurodynamical approach.

7.1. FUTURE RESEARCH DIRECTIONS

Our study on the neurodynamical approach for optimization is just starting. There are many interesting and important issues remain to be investigated. We list some of them here:

(i) We anticipate that the solution of any neurodynamical system should converge to an equilibrium point of the underlying ode system. For unconstrained problems, this result has been proved in [25] for the gradient direction. But a general result remains to be investigated.

(ii) General nonlinear constrained problem represents the hard core in optimization. A neurodynamical system for such problem is needed to consolidate the power and attractiveness of the neurodynamical approach. This has been under investigation by the authors.

(iii) Looking at the two neurodynamical systems established in Section 6, there is no matrix operation. This is certainly very attractive for large-scale problems. However, can the numerical ode solvers cope with such feature? This should post new challenges for the large-scale numerical ode solvers.

8. Acknowledgements

The research of Li-Zhi Liao is supported in part by grants FRG/99-00/II-23 and FRG/00-01/II-63 of Hong Kong Baptist University. The research of Liqun Qi is supported in part by the Research Grant Council of Hong Kong.

References

1. Aluffi-Pentini, F., Parisi, V. and Zirilli, F. (1984), A differential-equations algorithm for nonlinear equations, *ACM Trans. on Math. Software* 10 (3), 299–316.
2. Aluffi-Pentini, F., Parisi, V. and Zirilli, F. (1984) Algorithm 617 DAFNE: A differential-equations algorithm for nonlinear equations, *ACM Trans. on Math. Software*, 10 (3), 317–324.
3. Anstreicher, K. M. (1988), Linear programming and the Newton barrier flow, *Math. Prog.* 41, 367–373.
4. Arrow, K. J., Hurwicz, L. and Uzawa, H. (1958), *Studies in Linear and Nonlinear Programming*, Stanford University Press, Stanford, CA.
5. Barron, A. R. (1993), Universal approximation bounds for superpositions of a sigmoidal function, *IEEE Trans. Inform. Theory* 39 (3), 930–945.
6. Boggs, P. T. (1971), The solution of nonlinear systems of equations by A-stable integration techniques, *SIAM J. Numer. Anal.* 8 (4), 767–785.
7. Botsaris, C. A. and Jacobson, D. H. (1976), A Newton-type curvilinear search method for optimization, *JMAA*, 54, 217–229.
8. Botsaris, C. A. (1978), Differential gradient methods, *JMAA* 63, 177–198.
9. Botsaris, C. A. (1978), A curvilinear optimization method based upon iterative estimation of the eigensystem of the Hessian matrix, *JMAA* 63, 396–411.
10. Botsaris, C. A. (1978), A class of methods for unconstrained minimization based on stable numerical integration techniques, *JMAA* 63, 729–749.
11. Bouzerdoum, A. and Pattison, T. R. (1993), Neural network for quadratic optimization with bound constrains, *IEEE Trans. Neural Networks* 4, 293–304.
12. Branin, Jr. F. H., (1972), Widely convergent method for finding multiple solutions of simultaneous nonlinear equations, *IBM Journal of Research and Development* 16, 504–522.
13. Brown, A. A. and Bartholomew-Biggs, M. C. (1989), Some effective methods for unconstrained optimization based on the solution of systems of ordinary differential equations, *JOTA* 62 (2), 211–224.
14. Brown, A. A. and Bartholomew-Biggs, M. C. (1989), ODE versus SQP methods for constrained optimization, *JOTA* 62 (3), 371–386.
15. Chen, Y. H. and Fang, S. C. (1998), Solving convex programming problem with equality constraints by neural networks, *Computers Math. Appl.* 36, 41–68.
16. Chu, M. T. (1988), On the continuous realization of iterative processes, *SIAM Review* 30 (3), 375–387.
17. Cichocki, A. and Unbehauen, R. (1993), *Neural Networks for Optimization and Signal Processing*. Wiley, Chichester.

18. Cichocki, A., Unbehauen, R., Weinzierl, K. and Holzel, R., (1996), A new neural network for solving linear programming problems, *European J. Operational Res.*, 93, 244–256.
19. Chua, L. O. and Lin, G. N. (1984), Nonlinear programming without computation, *IEEE Trans. Circuits Syst.*, 31, 182–188.
20. Cybenko, G. (1989), Approximation by superpositions of a sigmoidal function, *Math. Control Signals Systems*, 2, 303–314.
21. Evtushenko, Yu. G. and Zhadan, V. G., (1978), A relaxation method for solving problems of non-linear programming, *U.S.S.R. Comput. Math. Math. Phys.* 17 (4), 73–87.
22. Diener, I. and Schaback, R., (1990), An extended continuous Newton method, *JOTA* 67 (1), 57–77.
23. Glazos, M. P., Hui, S. and Żak, S., (1998), Sliding modes in solving convex programming problems, *SIAM J. Control Optim.* 36, 680–697.
24. Goldstein, A. A. (1964), Convex programming in Hilbert space, *Bulletin of American Mathematical Society* 70, 709–710.
25. Han, Q., Liao, L.-Z., Qi, H. and Qi, L., (2001), Stability analysis of gradient-based neural networks for optimization problems, *J. Global Optim.* 19 (4), 363–381.
26. Hassan, N. and Rzymowski, W., (1990), An ordinary differential equation in nonlinear programming, *Nonlinear Analysis, Theory, Method & Applications* 15 (7), 597–599.
27. Haykin, S. S., (1994), *Neural Networks: A Comprehensive Foundation*, Prentice-Hall, Englewood Cliffs, NJ.
28. He, B. S., (1994), Solving a class of linear projection equations, *Numerische Mathematik* 68, 71–80.
29. He, B. S., (1997), A class of projection and contraction methods for monotone variational inequalities, *Applied Mathematics and Optimization* 35, 69–76.
30. He, B. S., (1999), Inexact implicit methods for monotone general variational inequalities, *Mathematical Programming*, 86 (1), 199–217.
31. He, B. S. and Yang H., (2000) A neural network model for monotone linear asymmetric variational inequalities, *IEEE Trans. Neural Networks*, 11, 3–16.
32. Hopfield, J. J., (1982), Neural networks and physical systems with emergent collective computational ability, Proc. Natl. Acad. Sci. USA, 79, 2554–2558.
33. Hopfield, J. J., (1984), Neurons with graded response have collective computational properties like those of two-state neurons, Proc. Natl. Acad. Sci., 81, 3088–3092.
34. Hopfield, J. J. and Tank, D. W., (1985), Neural computation of decisions in optimization problems, *Biolog. Cybernetics*, 52, 141–152.
35. Hornik, K., (1991), Approximation capabilities of multilayer feedforward networks, *Neural Networks*, 4, 251–257.
36. Hornik, K., Stinchcombe, M. and White, H., (1989), Multilayer feedforward networks are universal approximators, *Neural Networks*, 2, 359–366.
37. Hou, Z.-G., Wu, C.-P. and Bao, P., (1998), A neural network for hierarchical optimization of nonlinear large-scale systems, *International Journal of Systems Science* 29 (2), 159–166.
38. Incerti, S., Parisi, V. and Zirilli, F., (1979), A new method for solving nonlinear simultaneous equations, *SIAM J. Numer. Anal.* 16, 779–789.
39. Kennedy, M. P. and Chua, L. O., (1988), Neural networks for nonlinear programming, *IEEE Trans. Circuits Syst.* 35, 554–562.
40. Liao, L.-Z. and Qi, H., (1999), A neural network for the linear complementarity problem, *Math. Comput. Modelling* 29 (3), 9–18.
41. Liao, L.-Z., Qi, H. and Qi, L., (2001), Solving nonlinear complementarity problems with neural networks: a reformulation method approach, *JCAM* 131 (12), 343–359.
42. Lillo, W. E., Loh, M. H., Hui S. and Zak, S. H., (1993), On solving constrained optimization problems with neural networks: a penalty method approach, *IEEE Trans. Neural Networks*, 4, 931–940.

43. Maa, C. Y. and Shanblatt, M. A., (1992), Linear and quadratic programming neural network analysis, *IEEE Trans. Neural Networks* 3, 580–594.
44. Maa, C. Y. and Shanblatt, M. A., (1992), A two-phase optimization neural network, *IEEE Trans. Neural Networks* 3, 1003–1009.
45. Mangasarian, O. L., (1993), Mathematical programming in neural networks, *ORSA J. Comput.* 5 (4), 349–360.
46. Moré, J. J., Garbow, B. S. and Hillstrom, K. E., (1981), Testing unconstrained optimization software, *ACM Trans. Math. Software* 7 (1), 17–41.
47. Novaković, Z. R., (1990), Solving systems of non-linear equations using the Lyapunov direct method, *Computers Math. Applic.* 20 (12), 19–23.
48. Pan, P.-Q., (1992), New ODE methods for equality constrained optimization (1) – equations, *JCM* 10 (1), 77–92.
49. Pan, P.-Q., (1992), New ODE methods for equality constrained optimization (2) – algorithms, *JCM* 10 (2), 129–146.
50. Polyak, B. T., (1966), Constrained minimization problems, *USSR Computational Mathematics and Mathematical Physics* 6, 1–50.
51. Rodríguez-Vázquez, A., Domínguez-Castro, R., Rueda, A., Huertas J. L. and Sánchez-Sinencio, E., (1990), Nonlinear switch-capacitor 'neural' networks for optimization problems, *IEEE Trans. Circuits Syst.* 37, 384–398.
52. Schäffler, S. and Warsitz, H., (1990), A trajectory-following method for unconstrained optimization, *JOTA* 67 (1), 133–140.
53. Schnabel, R. B. and Eskow, E., (1990), A new modified Cholesky factorization, *SIAM J. Sci. Stat. Comput.* 11, 1136–1158.
54. Slotine, J.-J. E. and Li, W., (1991), *Applied Nonlinear Control*, Prentice-Hall, Englewood Cliffs, NJ.
55. Snyman, J. A., (1982), A new and dynamic method for unconstrained minimization, *Appl. Math. Modelling* 6, 449–462.
56. Solodov, M. V. and Tseng, P., (1996), Modified projection-type methods for monotone variational inequalities, *SIAM J. Control and Optimization* 34, 1814–1830.
57. Sudharsanan, S. and Sundareshan, M., (1991), Exponential stability and a systematic synthesis of a neural network for quadratic minimization, *Neural Networks* 4, 599–613.
58. Tanabe, K., (1980), A geometric method in nonlinear programming, *JOTA* 30 (2), 181–210.
59. Tank, D. W. and Hopfield, J. J., (1986), Simple neural optimization networks: An A/D convert, signal decision circuit, and a linear programming circuit, *IEEE Trans. Circuits Syst.* 33, 533–541.
60. Teo, K. L., Wong, K. H. and Yan, W. Y., (1995), Gradient-flow approach for computing a nonlinear-quadratic optimal-output feedback gain matrix, *JOTA* 85, 75–96.
61. Vincent, T. L., Goh, B. S. and Teo, K. L., (1992), Trajectory-following algorithms for min-max optimization problems, *JOTA* 75, 501–519.
62. Wilde, N. G., (1969), A note on a differential equation approach to nonlinear programming, *Management Science* 15 (11), 739–739.
63. Williems, J. L., (1970), *Stability Theory of Dynamical Systems*, Nelson.
64. Wu, X., Xia, Y., Li, J. and Chen, W. K., (1996), A high performance neural network for solving linear and quadratic programming problems, *IEEE Trans. Neural Networks*, 7, 643–651.
65. Xia, Y., (1996), A new neural network for solving linear programming problems and its applications, *IEEE Trans. Neural Networks* 7, 525–529.
66. Xia, Y., (1996) A new neural network for solving linear and quadratic programming problems, *IEEE Trans. Neural Networks* 7, 1544–1547.
67. Xia, Y. and Wang, J., (1998), A general methodology for designing globally convergent optimization neural networks, *IEEE Trans. Neural Networks* 9, 1331–1343.

68. Yamashita, H., (1980), A differential equation approach to nonlinear programming, *Math. Prog.* 18, 155–168.
69. Zabczyk, J., (1992), *Mathematical Control Theory: An Introduction*, Birkhauser, Boston.
70. Żak, S. H., Upatising, V. and Hui, S., (1995) Solving linear programming problems with neural networks: a comparative study, *IEEE Trans. Neural Networks* 6, 94–104.
71. Żak, S. H., Upatising, V., Lillo, W. E. and Hui, S., (1994), A dynamical systems approach to solving linear programming problems. In: K. D. Elworthy, W. N. Everitt and E. B. Lee (eds.), *Differential Equations, Dynamical Systems, and Control Science*, Marcel Dekker, New York.
72. Zhang, S. and Constantinides, A. G., (1992), Lagrange programming neural network, *IEEE Trans. Circuits Syst.* 39, 441–452.
73. Zhang, X.-S., (2000), *Neural Network in Optimization*, Kluwer Academic Publishers, Dordrecht.
74. Zhou, Z. and Shi, Y., (1997), An ODE method of solving nonlinear programming, *Computers Math. Applic.* 34 (1), 97–102.
75. Zhou, Z. and Shi, Y., (1998), A convergence of ODE method in constrained optimization, *JMAA* 218 (1), 297–307.
76. Zirilli, F., (1982), The use of ordinary differential equations in the solution of nonlinear systems of equations, Powell, M. J. D., (ed.) *Nonlinear Optimization* 1981, Academic Press, London.