Proceedings of the IEEE
International Conference on Automation and Logistics
Shenyang, China August 2009

Further Studies on Zhang Neural-Dynamics and Gradient Dynamics for Online Nonlinear Equations Solving*

Yunong Zhang[1], Peng Xu[1], Ning Tan[2]
[1]*School of Information Science and Technology,* [2]*School of Software*
Sun Yat-Sen University (SYSU), Guangzhou 510275, China
zhynong@mail.sysu.edu.cn, ynzhang@ieee.org

***Abstract*— By following Zhang *et al*'s neural-network design-method, a special kind of neural dynamics is generalized, developed and investigated in this work for online solution of nonlinear equation $f(x) = 0$. Different from conventional gradient-based dynamics (or termed, gradient-dynamics, GD), the resultant Zhang neural-dynamics (or termed, Zhang dynamics, ZD) is designed based on the elimination of an indefinite error-function (rather than the elimination of a square-based positive energy-function usually associated with gradient-based approaches). For comparative purposes, the gradient dynamics is developed and exploited as well for solving online such nonlinear equations. Conventionally and geometrically speaking, the gradient dynamics evolves along the surface descent direction (specifically, the tangent direction) of the square-based energy-function curve; but, how does Zhang neural-dynamics evolve? Together with our previous studies on gradient dynamics and Zhang dynamics, in this paper we further analyze, investigate and compare the characteristics of such two dynamics. Computer-simulation results via three illustrative examples might show us some interesting implications, in addition to the efficacy of Zhang dynamics on nonlinear equations solving.**

***Index Terms*— Gradient, Neural dynamics (ND), Error function, Energy function, Nonlinear equation, Convergence**

I. Introduction

The problem of solving online nonlinear equations [1] in form of $f(x) = 0$ is considered to be one of the basic and inspiring issues widely arising in science and engineering fields. Many numerical algorithms have thus been employed for solving online such nonlinear equations [2]–[4]. However, it may not efficient enough for most numerical algorithms due to their serial-processing nature performed on digital computers [5]. In recent years, due to the in-depth research in artificial neural network, numerous neural-dynamic and/or analogue solvers have been proposed and investigated [6]–[17]. They are now regarded as a powerful alternative to solving online various mathematical problems [5]–[17] because of their high-speed processing and parallel-distributed properties. However, it is worth mentioning that most existing computational-schemes were theoretically/intrinsically designed for time-invariant (or termed, static, constant) problems solving and/or related to gradient methods [6]–[9].

Different from gradient-based approaches [6]–[9], a special kind of recurrent-neural-networks and neural-dynamics has recently been proposed by Zhang *et al* (formally since March 2001) for the online time-varying (or termed, dynamic, moving, nonstationary) problems solving, such as, the time-varying matrix inversion [6], time-varying matrix square roots finding [7], time-varying Sylvester equation solving [12] and time-varying matrix pseudoinverse [17]. Especially in [1], for a simplified problem-formulation and ND solvers, we proposed, examined and compared the Zhang dynamics (with conventional gradient-dynamics) for online solution of nonlinear equation $f(x) = 0$. Following that, this paper is going to model, investigate and compare further the convergent performance of these two special neural-dynamics. The study results are shown in the ensuing sections.

II. Problem Formulation and ND Solvers

Let us consider the nonlinear equation of a general form,

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

where $f(\cdot) : \mathbb{R} \to \mathbb{R}$ is assumed continuously-differentiable. Our starting point in this work is to exploit ZD and GD to find a root $x \in \mathbb{R}$ such that the above nonlinear equation (1) holds true. For ease of presentation, let x^* denote such a theoretical solution (also termed, root, zero) of nonlinear equation (1). In the ensuing subsections, Zhang dynamics and gradient dynamics are both generalized, modeled and exploited (but comparatively) to solve nonlinear equation (1).

A. Zhang Neural-Dynamics

To monitor and control the solution process of nonlinear equation (1), following Zhang *et al*'s design method (having been proposed since March 2001, e.g., in [1], [7], [11]–[16]), we could firstly define the following indefinite error-function (of which the word "indefinite" means that such an error-function can be negative or even lower-unbounded):

$$e(t) := f\big(x(t)\big),$$

where, evidently, if the error-function $e(t)$ converges to zero, then $x(t)$ converges to theoretical solution (i.e., root) x^*.

Next, let the time derivative $\dot{e}(t)$ of error function $e(t)$ be chosen and forced mathematically such that $e(t)$ converges to zero; in math, we have to choose $\dot{e}(t)$ such that $\lim_{t\to\infty} e(t) = 0$. A suggested ZD (or termed ZNN, Zhang neural networks) design-formula for simply choosing a general form of the error-function's time-derivative $\dot{e}(t)$ can be the following (see [1],

*This work is supported by Program for New Century Excellent Talents in University (NCET-07-0887).

[7], [11]–[16] as well):

$$\frac{de(t)}{dt} = -\gamma\phi\big(e(t)\big), \text{ i.e., } \frac{df(x)}{dt} = -\gamma\phi\big(f(x)\big), \quad (2)$$

where design-parameter $\gamma > 0$ is used to control the convergence rate of the ND solution, and activation function $\phi(\cdot) : \mathbb{R} \to \mathbb{R}$ is assumed monotonically-increasing and odd. Expanding the ZD-design formula (2), we could have the following differential equation as of Zhang dynamics:

$$\frac{\partial f(x)}{\partial x}\frac{dx}{dt} = -\gamma\phi\big(f(x)\big), \text{ i.e., } f'(x)\dot{x} = -\gamma\phi\big(f(x)\big),$$
$$\text{or equivalently [if } f'(x) \neq 0], \ \dot{x}(t) = -\gamma\frac{\phi\big(f(x)\big)}{f'(x)}, \quad (3)$$

where $x(t)$, starting from randomly-generated initial condition $x(0) \in \mathbb{R}$, is the state and output of ZD (3) corresponding to theoretical root x^* of nonlinear equation (1).

In addition, it is worth pointing out here that, similar to usual neural-dynamic approaches, design parameter γ in (3) and hereafter, being the reciprocal of a capacitance parameter, can be set as large as hardware permits (e.g., in analog circuits or VLSI [18]) or selected appropriately (e.g., between 10^3 and 10^8) for experimental and/or simulative purposes. Moreover, it follows from ZD (3) that, different ZD performances can be achieved by using different design-parameter γ and activation function $\phi(\cdot)$. In general, any monotonically-increasing odd function can be chosen as the activation function of the neural dynamics. Since March 2001 [11], we have introduced and used four types of activation functions (i.e., linear activation function, power activation function, sigmoid activation function and power-sigmoid function) for the proposed ZD (or to say, ZNN) models (for more details, see [1], [6], [7], [9], [12], [13], [15], [16]).

For ZD (3) solving the nonlinear equation $f(x) = 0$, we have the following proposition about its convergence [1].

Proposition. Consider a solvable nonlinear equation $f(x) = 0$, where function $f(\cdot)$ is continuously differentiable (i.e., with at least the first-order derivative at some interval containing x^*). If a monotonically-increasing odd activation function $\phi(\cdot)$ is employed, then the neural state $x(t)$ of ZD (3), starting from randomly-generated initial condition $x(0) \in \mathbb{R}$, could converge to theoretical root x^* of nonlinear equation $f(x) = 0$ [of which the specific value of x^*, in the situation of not less than two zeros existing, depends on the sufficient closeness of initial state $x(0)$ to x^*].

B. Gradient Dynamics

The conventional ND schemes generally belong to the gradient-descent method in optimization [6]–[9]. For readers' convenience, we show the GD design procedure below.

Firstly, a square-based nonnegative energy-function such as $\mathcal{E}(x) = f^2(x)/2$ can be constructed so that its minimum points are the roots of the nonlinear equation $f(x) = 0$.

Secondly, the solution algorithm is designed to evolve along a descent direction of this energy function until a minimal point is reached. The typical descent direction is the negative gradient of this energy function $\mathcal{E}(x)$.

Thirdly, by using the negative-gradient method to construct a neural-dynamic model for solving nonlinear equation $f(x) = 0$, we could have the following dynamic equation as of the conventional linear gradient-dynamic solver:

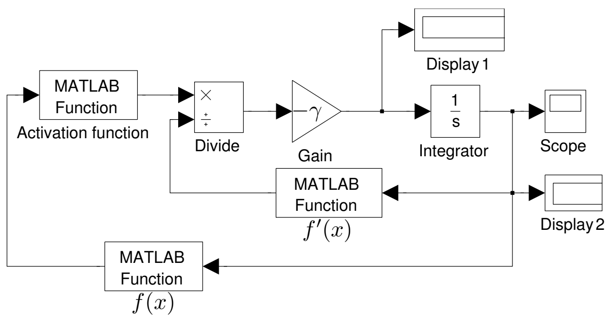
$$\frac{dx(t)}{dt} = -\gamma\frac{\partial\mathcal{E}(x)}{\partial x} = -\gamma f(x)\frac{\partial f(x)}{\partial x} = -\gamma f(x)f'(x).$$

As an extension of the above GD design method and model (which is inspired by [12]–[17] as well), we can generalize a nonlinear form of gradient-based neural-dynamics as the following by exploiting nonlinear activation function $\phi(\cdot)$:

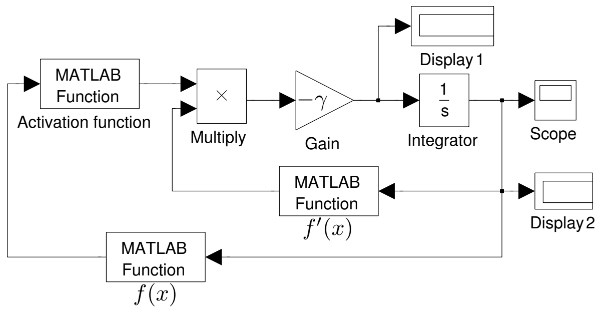
$$\dot{x}(t) = -\gamma\phi\big(f(x)\big)f'(x), \quad (4)$$

where design parameter $\gamma > 0$ and activation function $\phi(\cdot) : \mathbb{R} \to \mathbb{R}$ are defined to be the same as those of ZD (3).

Furthermore, according to the neural-dynamic equations (3) and (4) and the block diagrams of the neural-dynamic solvers depicted in [1], the Simulink-based models representation of Zhang dynamics and gradient dynamics can be seen from Fig. 1. The difference between ZD (3) and GD (4) for this specific problem $f(x) = 0$ solving is evidently interesting: ZD has a division term of $f'(x)$, whereas GD has a multiplication term of $f'(x)$. In addition to the difference, in view of the fact that both ZD and GD solvers could work effectively for this problem solving [1], people may ask why and how $f'(x)$ is needed as well as how ZD and GD solvers differ from each other. For more detailed, complete and systematic comparisons, please refer to [1], [6]–[17].



(a) a Simulink-based model representation of ZD (3)



(b) a Simulink-based model representation of GD (4)

Fig. 1. Simulink-based models representation about ZD (3) and GD (4).

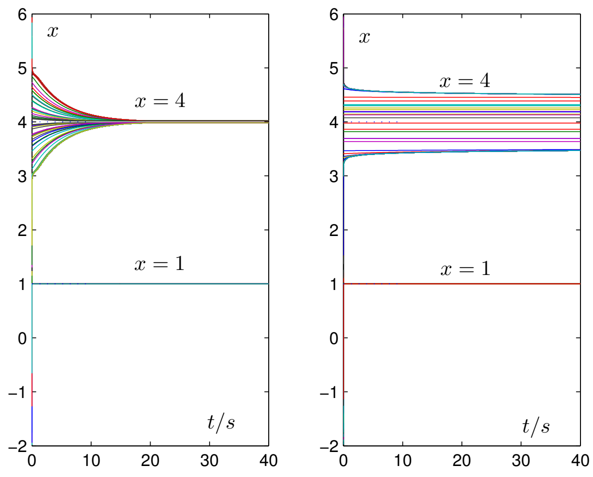


Fig. 2. Online solution to $(x-4)^{10}(x-1)=0$ by ZD (3) and GD (4) with design parameter $\gamma = 1$. Left: ZD (3); Right: GD (4).

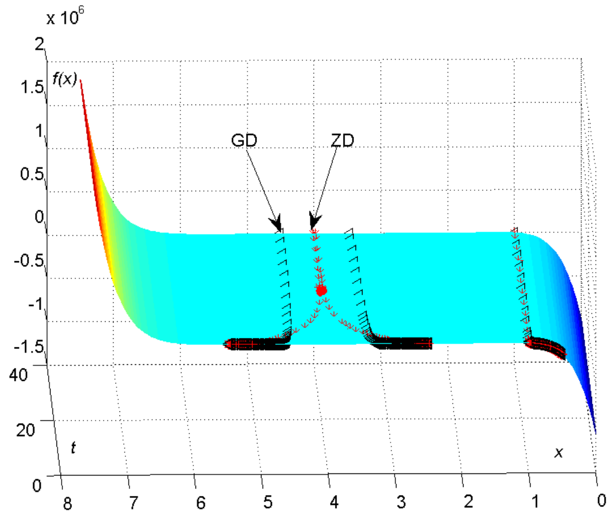


Fig. 3. 3-dimensional convergent performance of ZD (3) and GD (4) with design parameter $\gamma = 1$ for online solution of $(x-4)^{10}(x-1)=0$.

III. Modeling, Simulation and Verification

Section II presents the general forms of Zhang dynamics and gradient dynamics for zero finding, together with theoretical results about ZD convergence. In this section, several illustrative examples are presented to investigate and show the convergent performance of the two kinds of neural dynamics, which are based on the use of the power-sigmoid activation function with design parameters $\xi = 4$ and $p = 3$ [1].

A. Example 1

Let us consider the following nonlinear equation:

$$f(x) = (x-4)^{10}(x-1) = 0 \qquad (5)$$

To check the ND-solution correctness, the theoretical roots to the above nonlinear equation can be given simply as $x_1^* = 4$ (a multiple root of order 10) and $x_2^* = 1$ (a simple root).

It can be seen from the left subgraph of Fig. 2 that the neural state $x(t)$ of ZD (3), starting from 200 initial states randomly generated within [-2,6], could converge to a theoretical root of nonlinear equation (5), either $x_1^* = 4$ or $x_2^* = 1$. In contrast, by applying gradient-based neural dynamics (4), some less-correct solutions (in other words, different from $x_1^* = 4$ and $x_2^* = 1$) have also been generated, which can be seen clearly in the right subgraph of Fig. 2. In addition, the 3-dimensional convergent performance of ZD (3) and GD (4) solving nonlinear equation (5) is shown in Fig. 3. It can be seen from the figure that, starting from initial states [e.g., $x(0) = 0.5$ here] close enough to the simple root $x^* = 1$, the neural states $x(t)$ of both ZD (3) and GD (4) converge to theoretical root $x^* = 1$. On the other hand, for the case of multiple root $x^* = 4$ of order 10, if the initial states [e.g., $x(0) = 2.5$ or $x(0) = 5.5$ here] are close enough to this multiple root, the neural state $x(t)$ of ZD (3) could converge well to the theoretical root $x_1^* = 4$; but, when applying GD (4) under the same conditions, the GD neural-state $x(t)$ will mostly converge to a less-correct solution (or to say, an approximate solution with $0 < |x(40) - 4| < 1$).

Moreover, we have also investigated the convergent performance of the two dynamics solving nonlinear equation similar to (5) but with a multiple root of order less than 10 (specifically, 3 and 2); i.e., $f(x) = (x-4)^3(x-1) = 0$ and $f(x) = (x-4)^2(x-1) = 0$. The results are presented now and here. It follows from Figs. 4 and 5 that, starting from 200 initial states randomly generated within [-2,6] and sufficiently close to simple root $x^* = 1$, the neural state $x(t)$ of both ZD (3) and GD (4) could converge well to such a theoretical root $x^* = 1$. On the other hand, for the case of the multiple root $x^* = 4$, the ZD neural-state $x(t)$ can (almost) always converge well to such a theoretical root $x^* = 4$; but, similar to Fig. 2, GD (4) may still generate less correct (or approximate) solutions. In addition, the corresponding 3-dimensional convergent performance of the two kinds of neural-dynamics is shown in Figs. 6 and 7 (note that the three initial states involved are 0.5, 2.5 and 5.5).

It can be seen through Figs. 2 to 7 that, for the case of the multiple root (i.e., $x^* = 4$), the neural state $x(t)$ of ZD (3) can (almost) always converge accurately to the theoretical root regardless of its order. The efficacy is possibly owing to the division term of $f'(x)$ exploited in ZD (3). In contrast, GD (4) may generate less correct results (or to say, approximate solutions). Moreover, comparing the right subgraphs of Figs. 2, 4 and 5 (corresponding to Figs. 3, 6 and 7, respectively), we can observe that, with the increase of the order of the multiple root $x^* = 4$, the convergent performance of GD (4) to it becomes (much) worse. This increasing inability, as we have investigated, is possibly owing to the multiplication term of $f'(x)$ exploited in GD (4). For more information, please expand these ND equations for such specific functions $f(x)$.

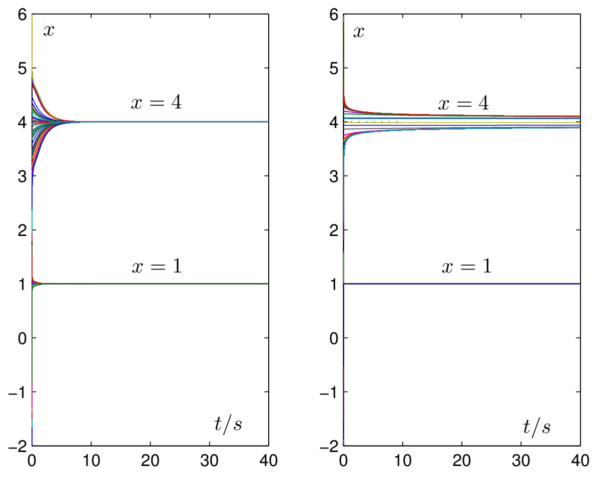


Fig. 4. Online solution to $(x-4)^3(x-1)=0$ by ZD (3) and GD (4) with design parameter $\gamma=1$. Left: ZD (3); Right: GD (4).

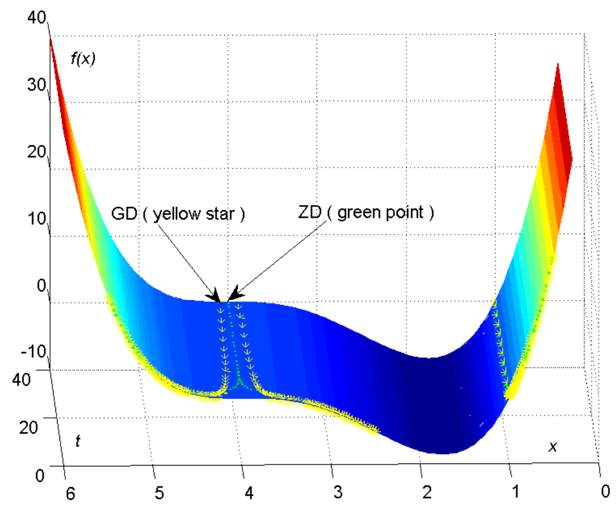


Fig. 6. 3-dimensional convergent performance of ZD (3) and GD (4) with design parameter $\gamma=1$ for online solution of $(x-4)^3(x-1)=0$.

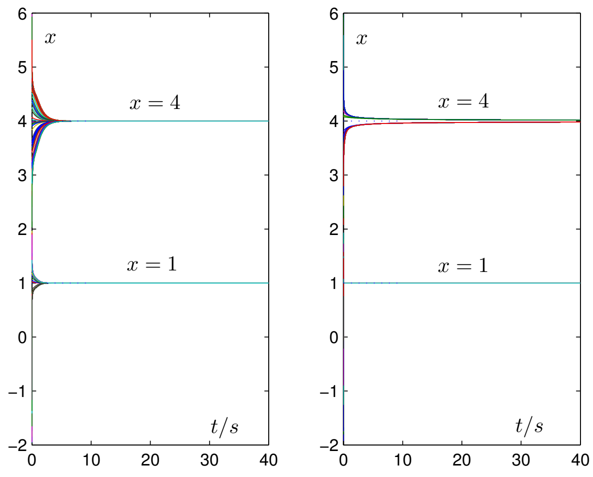


Fig. 5. Online solution to $(x-4)^2(x-1)=0$ by ZD (3) and GD (4) with design parameter $\gamma=1$. Left: ZD (3); Right: GD (4).

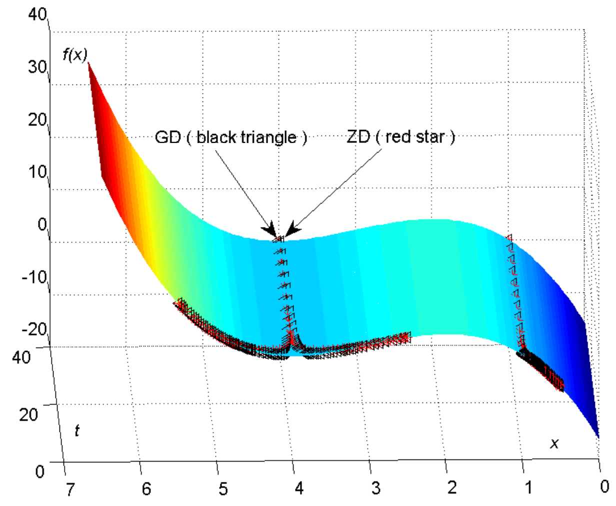


Fig. 7. 3-dimensional convergent performance of ZD (3) and GD (4) with design parameter $\gamma=1$ for online solution of $(x-4)^2(x-1)=0$.

B. Example 2

Let us consider the following nonlinear equation:

$$f(x)=0.01(x+7)(x-1)(x-8)+\sin x+2.4=0, \quad (6)$$

with its three theoretical roots denoted by "root 1", "root 2" and "root 3" in Fig. 8. By using ZD (3) to solve the above nonlinear equation (6), we can see from Fig. 8 that, starting from different initial states close enough, the neural state $x(t)$ could converge to their corresponding theoretical roots. For example, if initial state $x(0)$ is -9.8 or -5.0 (corresponding to "initial state 1" or "initial state 2" in the figure), the neural state $x(t)$ of ZD (3) converges to the "root 1" as depicted in Fig. 8. Similarly, starting from initial state $x(0)=1.0$ or 9.5 (corresponding to "initial state 5" and "initial state 6" in the figure), the neural state $x(t)$ would respectively converge to "root 2" and "root 3" as depicted in Fig. 8.

Correspondingly, let us see another simulation result, i.e., Fig. 9. Starting from 50 initial states randomly generated within [-10,10] and sufficiently close to the theoretical roots (i.e., "root 1", "root 2" and "root 3"), the neural states $x(t)$ of both ZD (3) and GD (4) converge well to them. But, for some initial states [e.g., $x(0)=-4$ or 0.8 corresponding to "initial state 3" or "initial state 4", respectively, in Fig. 8] which are close to a local minimum point, GD (4) will generate a wrong solution, whereas, for this situation, the ZD-state $x(t)$ will move towards the local minimum point and then stop with the following warning information.

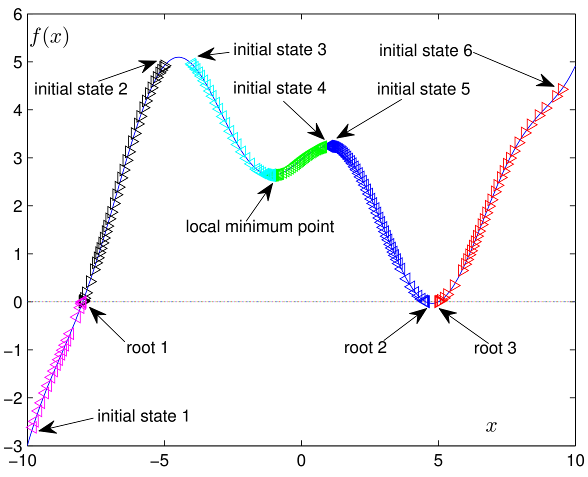


Fig. 8. Convergent performance of the neural state of ZD (3) for online solution of nonlinear equation (6) starting from different initial states.

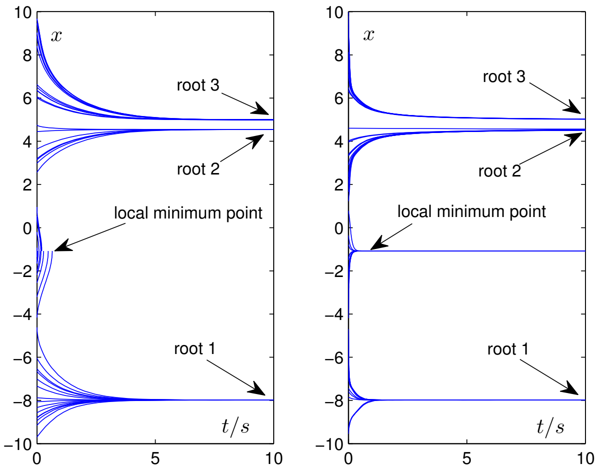


Fig. 9. Online solution of $0.01(x+7)(x-1)(x-8)+\sin x+2.4=0$ by ZD (3) and GD (4) with parameter $\gamma=1$. Left: ZD (3); Right: GD (4).

```
Warning: Failure at t=1.479876e-005. Unable
to meet integration tolerances without redu
-cing the step size below the smallest valu
-e allowed (2.710505e-020) at time t.
> In ode15s at 741  In znnvsgnnft at 20
```

This warning may remind users to re-choose a suitable initial state with which to re-start the ZD-solution process for correct results. In contrast, as mentioned before, for this situation, GD (4) will yield a wrong solution [i.e., the local minimum point with $f'(x)=0$], which may mislead the users into obtaining a "solution" of the nonlinear equation.

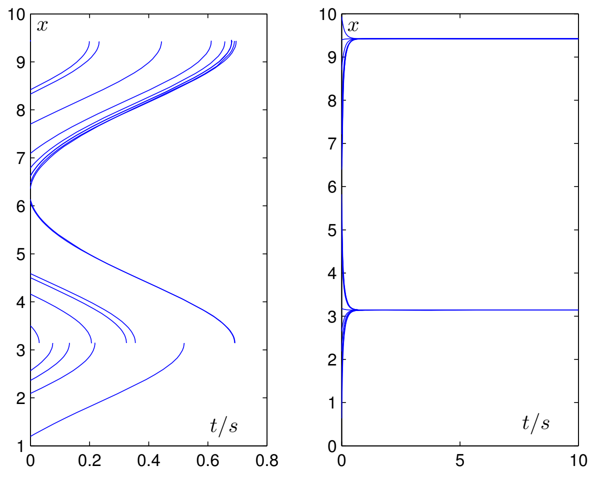


Fig. 10. Online solution of ZD (3) (left) and GD (4) (right) with $\gamma=1$ for nonlinear equation $\cos x+3=0$ which evidently has no theoretical roots.

C. Example 3

Let us consider the following nonlinear equation:

$$f(x)=\cos x+3=0, \tag{7}$$

for another interesting observation. Evidently, the above nonlinear equation (7) has no solution (i.e., no root, no zero). As can be seen from Fig. 10, by applying ZD (3), the neural state $x(t)$, starting from 20 initial states within [0,10], moves towards some values and then stops with a warning as the above. In contrast, GD (4) generates some misleading results again. The specific values involved, as we investigate, are related to the local minimum points satisfying $f'(x)=-\sin x=0$; i.e., $x=k\pi$ with $k=0,\pm1,\pm2,\cdots$. In addition, it is worth mentioning an anonymous reviewer's understanding; i.e., according to his/her experience, the warning given by the simulation tool may not mean "(it) remind(s) users to re-choose a suitable initial state"; on the contrary, it may mean "(we) are using the wrong tools for the job". For $f(x)=\cos x+3=0$ depicted in (7), there is evidently no solution (or to say, no root), and thus no correct tools could complete the root-finding job. So, we believe, with many thanks to the reviewer, that a warning given by the simulation tool in our context means "it reminds users to re-choose a suitable initial state or tells users no solution around (the current initial state)". Besides, the authors suggest the readers checking the simulation manuals for more and exact information when conducting such simulations, in addition to having a right understanding of the warning messages.

In summary, from the above three illustrative examples and [1], we have the following observations and facts about the convergent performance of ZD (3) and GD (4) exploited for the online solution of nonlinear equation $f(x)=0$.

1) If the randomly-generated initial states are close enough to a theoretical root, for the case of a simple root, ZD (3) and GD (4) could both effectively converge to the

theoretical simple root; and, for the case of a multiple root, with the increase of the order of the multiple root, the neural state $x(t)$ of ZD (3) could still converge well to such a theoretical root [in contrast, GD (4) more probably yields wrong (or approximate) solutions].

2) For a nonlinear equation containing local minimum point(s) (e.g., see Example 2), if a initial state is close enough to the local minimum point, then the neural state $x(t)$ of ZD (3) would move towards the local-minimum point and then stop with a warning. This reminds users to re-choose another initial state and to start the ZD solver again if necessary. However, in this local-minimum-point situation, GD (4) generates wrong results, misleading users into thinking that a correct root is now obtained.
3) For the case of a nonlinear equation having no solution, the neural state of ZD (3) move towards some value(s) [possibly the local minimum point(s) satisfying $f'(x) = 0$] and then stop with the warning information. In contrast, in this no-solution situation, GD (4) would generate wrong/misleading solution(s) again, which appears to be similar to the above observation.

Furthermore, before ending this section, it is worth pointing out that the authors agree very well with an anonymous reviewer about the following thoughts. That is, "in addition to the theoretical results given in the authors' previously published papers, the main points of this paper are in Section III; and, logically speaking, an example is enough to prove something worse (and/or wrong), but far from enough to prove something better (and/or correct)". So, we, the authors, together with some interested readers, may have to devote much more research effort still to this topic.

IV. Conclusions

For online solution of nonlinear equations, a special kind of neural dynamics has been generalized, developed, modeled and compared in this paper by following Zhang *et al*'s method. Different from conventional gradient-based neural-dynamics, the resultant Zhang neural-dynamics has been elegantly driven by an indefinite error-function [instead of usually-exploited lower-bounded energy-function(s)]. For comparative purposes, the gradient-based neural-dynamics has also been applied to the online solution of nonlinear equations. Following the previous studies on gradient dynamics and Zhang dynamics [1], we further investigate and compare the convergent performance of the two neural-dynamics. Computer-modeling and simulation results via three illustrative examples (in addition to [1]) have substantiated further the efficacy of Zhang neural-dynamics on static nonlinear equations solving as well (which is just a byproduct of the ZNN-based time-varying matrix/vector problems solving).

Acknowledgments

This work is supported by the Program for New Century Excellent Talents in University (NCET-07-0887). Before joining SYSU in 2006, Yunong Zhang had been with NUIM, University of Strathclyde, NUS, CUHK, SCUT, and HUST, since 1992. Yunong had been supported by those research fellowships, assistantship and studentships, which had been related to this research directly or indirectly. His web-page is now at http://www.ee.sysu.edu.cn/teacher/detail.asp?sn=129. In addition, with the following words, Yunong would like to thank and encourage the student-coauthors for their important contribution: "Most researchers had once been students, and good students may finally become leading researchers".

References

[1] Y. Zhang, C. Yi, and W. Ma, "Comparison on gradient-based neural dynamics and Zhang neural dynamics for online solution of nonlinear equations," *in LNCS Proceedings of International Symposium on Intelligence Computation and Applications,* vol. 5370, pp. 269–279, 2008.

[2] J. R. Sharma, "A composite third order newton-steffensen method for solving nonlinear equations," *Applied Mathematics and Computation,* vol. 169, pp. 242–246, 2005.

[3] C. Chun, "Construction of Newton-like iteration methods for solving nonlinear equations," *Numerische Mathematik,* vol. 104, pp. 297–315, 2006.

[4] N. Ujevic, "A method for solving nonlinear equations," *Applied Mathematics and Computation,* vol. 174, pp. 1416–1426, 2006.

[5] Y. Zhang, W. E. Leithead, and D. J. Leith, "Time-series Gaussian process regression based on Toeplitz computation of $O(N^2)$ operations and $O(N)$-level storage," *in Proceedings of the 44th IEEE Conference on Decision and Control,* pp. 3711–3716, 2005.

[6] Y. Zhang, K. Chen, W. Ma, and X. Li, "MATLAB simulation of gradient-based neural network for online matrix inversion," *in LNAI Proceedings of International Conference on Intelligent Computing,* vol. 4682, pp. 98–109, 2007.

[7] Y. Zhang and Y. Yang, "Simulation and comparison of Zhang neural network and gradient neural network solving for time-varying matrix square roots," *in Proceedings of the 2nd International Symposium on Intelligent Information Technology Application,* vol. 2, pp. 966–970, 2008.

[8] J. Wang, "A recurrent neural network for real-time matrix inversion," *Applied Mathematics and Computation,* vol. 55, pp. 89–100, 1993.

[9] Y. Zhang, "Revisit the analog computer and gradient-based neural system for matrix inversion," *in Proceedings of IEEE International Symposium on Intelligent Control,* pp. 1411–1416, 2005.

[10] Y. Zhang, "Towards piecewise-linear primal neural networks for optimization and redundant robotics," *in Proceedings of IEEE International Conference on Networking, Sensing and Control,* pp. 374–379, 2006.

[11] Y. Zhang, D. Jiang, and J. Wang, "A recurrent neural network for solving Sylvester equation with time-varying coefficients," *IEEE Transactions on Neural Networks,* vol. 13, pp. 1053–1063, 2002.

[12] Y. Zhang, Z. Fan, and Z. Li, "Zhang neural network for online solution of time-varying Sylvester equation," *in LNCS Proceedings of International Symposium on Intelligence Computation and Applications,* vol. 4683, pp. 276–285, 2007.

[13] Y. Zhang and S. S. Ge, "Design and analysis of a general recurrent neural network model for time-varying matrix inversion," *IEEE Transactions on Neural Networks,* vol. 16, pp. 1477–1490, 2005.

[14] K. Chen, S. Yue, and Y. Zhang, "MATLAB simulation and comparison of Zhang neural network and gradient neural network for online time-varying matrix equation $AXB - C = 0$," *in Proceedings of International Conference on Intelligent Computing,* pp. 68–75, 2008.

[15] Y. Zhang and H. Peng, "Zhang neural network for linear time-varying equation solving and its robotic application," *in Proceedings of the 6th International Conference on Machine Learning and Cybernetics,* pp. 3543–3548, 2007.

[16] Y. Zhang, Z. Chen, K. Chen, and B. Cai, "Zhang neural network without using time-derivative information for constant and time-varying matrix inversion," *in Proceedings of International Joint Conference on Neural Networks,* pp. 142–146, 2008.

[17] Y. Zhang, N. Tan, B. Cai, and Z. Chen, "MATLAB Simulink modeling of Zhang neural network solving for time-varying pseudoinverse in comparison with gradient neural network," *in Proceedings of the 2nd International Symposium on Intelligent Information Technology Application,* vol. 1, pp. 39-43, 2008.

[18] C. Mead, *Analog VLSI and Neural Systems,* Addison-Wesley, Reading, MA, 1989.