A Dynamic Neural Network Approach for Solving Nonlinear Inequalities Defined on A Graph and Its Application to Distributed, Routing-free, Range-free Localization of WSNs

Shuai Li [a], Feng Qin [b]

Department of Electrical and Computer Engineering, Stevens Institute of Technology, Hoboken, NJ 07030, USA.

[b] *School of Computer Science, Anhui University of Technology, Ma'anshan, Anhui, China.*

Abstract

In this paper, we are concerned with the problem of finding a feasible solution to a class of nonlinear inequalities defined on a graph. A recurrent neural network is proposed to tackle this problem. The convergence of the neural network and the solution feasibility to the defined problem are both theoretically proven. The proposed neural network features a parallel computing mechanism and a distributed topology isomorphic to the corresponding graph. Thus it is suitable for distributed real-time computation. The proposed neural network is applied to range-free localization of wireless sensor networks (WSNs). The analog circuit implementation of the neural network for such an application is also explored. Simulations demonstrate the effectiveness of the proposed method.

Key words: Recurrent neural network, wireless sensor networks, range-free localization, distributed estimation, routing-free localization.

1 Introduction

Large scale networks widely exist in natural or man-made systems. Typical examples include metabolic networks [?], power networks [?], wireless sensor networks (WSNs) [?], robot networks [?], etc. Many problems associated

[1] Email Address: lshuai@stevens.edu (Shuai Li), fqin@ahut.edu.cn (Feng Qin).

with large scale networks, are emerging and receiving intensive studies. For example, the problem of predicting protein folding pathways can be defined as an global minimization problem defined on a graph formed by amino acid sequence [?]; the cooperative control of robot networks can be modeled as a dynamic programming problem defined on a graph constructed by robot interactions [?]. Particularly, there is a class of widely existing problems associated with a large scale network, which can be described by a set of nonlinear inequalities relevant to the interactions between agents. For instance, the communication connectivity maintenance of a robot network is concerned with forming a topological structure, on which the distance between neighbor robots does not exceed the limit of communication range [?]; the range-free localization of WSNs is concerned finding an estimation of sensor node positions, which does not violate the proximity topology constraints [?]. In this paper, we formulate this class of problems in an unified framework as nonlinear inequalities defined on a graph. For problems defined on a graph with a huge number of agents involved, traditional centralized methods are challenged for scenarios with no powerful central computers. In contrast, distributed methods, which allocate the whole complicated task to every agents on the graph and solve the problem cooperatively, demonstrate great attractions.

In recent years, recurrent neural networks have received considerable studies in signal processing [?], pattern classification [?, ?], robotics [?], optimization [?], etc. Particularly, after the invention of the Hopfield neural network [?], which was originally designed for online optimization, recurrent neural networks, resulting from its parallelism and online solving capability, are becoming more and more popular in online optimization [?]. In this paper, we design a recurrent neural network to tackle the formulated nonlinear inequality problem defined on a graph in real time and apply the neural network model to range-free localization of WSNs.

To a constrained optimization problem, early works on recurrent neural networks [?, ?] often design a recurrent neural network evolving along the gradient descent direction of an augmented energy function formed by introducing a constraint-related penalty term into the objective function. However, the solution of the neural network often has a small deviation from the optimal one due to the compromise between the cost function and the constraints. To remedy this, [?, ?] study the problem in the dual space and use a projection function to represent inequality constraints. For the case with linear inequality constraints, it is proven in theory that this type of methods are able to converge to the optimal solution. The results are latter applied to various applications, such as kinematic control of redundant manipulators [?], k-winner-take-all (k-WTA) problem solving [?], L_1 norm estimation [?], to exploit the real-time processing capability of the neural networks. Successive works [?, ?, ?] in this field extend the results to the cases with nonlinear constraints with guaranteed convergence of the recurrent neural network. Although great success has been gained in the field of using recurrent neural networks to solve constrained optimization, difficulties are encountered when using recurrent neural networks to solve nonlinear inequalities defined on a large scale graph as the recurrent neural network approach is essentially a centralize method and is not scalable to a large networked systems. By taking advantage of the local connection between nodes on a graph, a distributed recurrent neural network, whose topology is isomorphic to the graph topology, is elaborately designed to solve nonlinear inequalities defined on the graph in a decentralized fashion.

A prominent feature of the proposed neural network to solve nonlinear inequalities defined on a graph is that its topology is isomorphic to the graph topology, i.e., each neuron in the neural network corresponds to a node on the graph and the neural connections corresponds to edges on the graph, which is in contrast to the conventional neural networks with connections between all pairs of neurons. For applications, this topological feature enables distributed computation, which is often necessary for a scalable large scale network.

The remainder of this paper is organized as follows. In Section 2, we present the neural network model for solving the nonlinear inequality problem defined on a graph. In Section 3, the convergence of the neural network is analyzed and it is proven to be convergent to a feasible solution of the problem. In Section 4, the neural network model is applied to solve range-free localization of WSNs. Its hardware implementation is explored in Section **??**. In Section **??**, simulation examples are given to demonstrate the effectiveness of our method. Section **??** concludes this paper.

2 Model Description

In this section, we are concerned with inequalities defined on a graph $\mathcal{G}(\mathbb{V}, \mathbb{E})$ (where $\mathbb{V}$ and $\mathbb{E}$ denote the vertex set and the edge set, respectively) with the following expression:

$$f_{ij}(x_i, x_j) \leq 0 \quad \text{for } j \in \mathbb{N}(i) \bigcup \{i\}, \forall i \in \mathbb{V} \tag{1}$$

where $\mathbb{N}(i)$ denotes the neighbor set of vertex i, which is defined as $\mathbb{N}(i) = \{j \in \mathbb{V} | \{i, j\} \in \mathbb{E}\}$. $x_i \in \mathbb{R}^m$ is the state variable associated with vertex i. $f_{ij}(x_i, x_j) \in \mathbb{R}$ is a nonlinear convex function with respect to x_i and x_j.

Note that there is no explicit objective function but inequality constraints in problem (1). Mostly, the solution to this problem is not unique. In some applications, such as range-free localization of WSNs [**?**, **?**] (as detailed in Section 4) and communication connectivity maintenance in robot networks [**?**, **?**], researchers are more concerned with finding a feasible solution in real time instead of finding all the feasible solutions. Based on this consideration, we explore finding a feasible solution to problem (1) in real time via a recurrent dual neural network.

Problem (1) is equivalent to the following normal optimization problem by augmenting the original problem with a virtual objective function,

$$\begin{aligned}&\text{minimize} \quad J = 0\\ &\text{subject to} \quad f_{ij}(x_i, x_j) \leq 0 \quad \text{for } j \in \mathbb{N}(i) \bigcup \{i\}, \forall i \in \mathbb{V}\end{aligned} \tag{2}$$

where J is the virtual objective function. According to Karash-Kuhn-Tucker (KKT) conditions [?], the solution to problem (2) satisfies,

$$\lambda_{ii}\frac{\partial f_{ii}(x_i, x_i)}{\partial x_i} + \sum_{j\in\mathbb{N}(i)}\Big(\lambda_{ij}\frac{\partial f_{ij}(x_i, x_j)}{\partial x_i} + \lambda_{ji}\frac{\partial f_{ji}(x_j, x_i)}{\partial x_i}\Big) = 0 \ \forall i \in \mathbb{V} \tag{3a}$$

$$f_{ij}(x_i, x_j)\begin{cases} = 0 & (\lambda_{ij} > 0) \\ < 0 & (\lambda_{ij} = 0)\end{cases} \quad \text{for } j \in \mathbb{N}(i) \bigcup \{i\}, \forall i \in \mathbb{V} \tag{3b}$$

where $\lambda_{ij} \in \mathbb{R}$, $\lambda_{ji} \in \mathbb{R}$ and $\lambda_{ii} \in \mathbb{R}$ are all dual variables. Note that (3b) can be simplified into the following form,

$$g\Big(f_{ij}(x_i, x_j) + \lambda_{ij}\Big) = \lambda_{ij} \quad \text{for } j \in \mathbb{N}(i) \bigcup \{i\}, \forall i \in \mathbb{V} \tag{4}$$

with the function $g(u) = [g_1(u_1), g_2(u_2), ..., g_k(u_k)]^T$ for $u = [u_1, u_2, ..., u_k]^T$ defined as follows for $i = 1, 2, ..., k$,

$$g_i(u_i) = \begin{cases} u_i & \text{if } u_i > 0 \\ 0 & \text{if } u_i \leq 0\end{cases} \tag{5}$$

Up to now, the original inequality problem (1) is equivalently transformed to nonlinear equations consisting of (3a) and (4). We use a recurrent neural network to solve the variables in (3a) and (4) as follows:

$$\begin{aligned}\dot{x}_i &= -\epsilon\Bigg(\lambda_{ii}\frac{\partial f_{ii}(x_i, x_i)}{\partial x_i} + \sum_{j\in\mathbb{N}(i)}\Big(\lambda_{ij}\frac{\partial f_{ij}(x_i, x_j)}{\partial x_i} + \lambda_{ji}\frac{\partial f_{ji}(x_j, x_i)}{\partial x_i}\Big)\Bigg) \ \forall i \in \mathbb{V}\\ \dot{\lambda}_{ij} &= \epsilon g\Big(f_{ij}(x_i, x_j) + \lambda_{ij}\Big) - \epsilon\lambda_{ij} \quad \text{for } j \in \mathbb{N}(i) \bigcup \{i\}, \forall i \in \mathbb{V}\end{aligned} \tag{6}$$

where $\epsilon > 0$ is a scaling factor, the function $g(\cdot)$ is defined in (5), $\lambda_{ij} \in \mathbb{R}$, $\lambda_{ii} \in \mathbb{R}$ and $\lambda_{ji} \in \mathbb{R}$ are dual variables associated with the edge $\{i, j\} \in \mathbb{E}$, the vertex i, and the edge $\{j, i\} \in \mathbb{E}$, respectively. About the distributiveness of the proposed neural network model (6), we have the following remark.

In the dynamic equation (6), the update of x_i, which is associated with the vertex i, requires information on the value of x_i itself, the value of x_j for $j \in \mathbb{N}(i)$, which is associated with i's neighbor vertex j, the value of λ_{ii} associated with the vertex i, the value of λ_{ij} associated with the edge $\{i, j\} \in \mathbb{E}$ and the value of λ_{ji} associated with the edge $\{j, i\} \in \mathbb{E}$. All the above required information comes either from the vertex i itself or from its neighbor vertices or the edges in between. On the other hand, for the update of λ_{ij}, which is associated with the edge $\{i, j\} \in \mathbb{E}$, the required information includes the value of λ_{ij} itself, the value of x_i and the value of x_j, which are associated with the two vertices forming the edge $\{i, j\}$. Clearly, all the required information for the update of λ_{ij} is available in the neighborhood. In this sense, the neural network (6) has an isomorphic topology to the graph $\mathcal{G}(\mathbb{V}, \mathbb{E})$.

Remark 1 *The connection topology of the neural network model (6) is isomorphic to the graph $\mathcal{G}(\mathbb{V}, \mathbb{E})$, on which the problem (1) is defined. This property enables us to implement the neural network in a distributed fashion by assigning the state variables x_i, λ_{ii} and λ_{ij} for all $j \in \mathbb{N}(i)$ to the vertex i. In this way, for the dynamic evolution of the neural network (6), communications only happen between neighbor vertices and neither routing nor cross-hop communication is required, which thoroughly reduces the communication burden especially for the case with a large number of vertices.*

For the convenience of analysis, the neural network dynamic (6) can be equivalently written into a compact form,

$$\begin{aligned} \dot{x} &= -\epsilon\Big(\nabla^T \bar{F}(x)\Big)\bar{\Lambda} \\ \dot{\bar{\Lambda}} &= \epsilon\bigg(g\Big(\bar{F}(x) + \bar{\Lambda}\Big) - \bar{\Lambda}\bigg) \end{aligned} \tag{7}$$

where x is a mn dimensional vector with $n = |\mathbb{V}|$ denoting the number of vertices on the graph, m denoting the dimension of x_i for $i \in \mathbb{V}$ and $x = [x_1^T, x_2^T, ..., x_n^T]^T$, $F(x) \in \mathbb{R}^{n\times n}$ is a matrix function of x with the ijth entry defined as follows:

$$F_{ij}(x) = \begin{cases} f_{ij}(x_i, x_j) & \text{for } i \in \mathbb{V}, j \in \mathbb{V} \text{ and } j \in \mathbb{N}(i) \\ 0 & \text{for } i \in \mathbb{V}, j \in \mathbb{V} \text{ and } j \notin \mathbb{N}(i) \end{cases} \tag{8}$$

Similarly, $\Lambda \in \mathbb{R}^{n\times n}$ is defined as:

$$\Lambda_{ij} = \begin{cases} \lambda_{ij} & \text{for } i \in \mathbb{V}, j \in \mathbb{V} \,\text{and}\, j \in \mathbb{N}(i) \\ \mu_{ij} & \text{for } i \in \mathbb{V}, j \in \mathbb{V} \,\text{and}\, j \notin \mathbb{N}(i) \end{cases} \tag{9}$$

where $\mu_{ij} \in \mathbb{R}$ and is always initialized to be zero, i.e., $\mu_{ij}(0) = 0$. In (7), $\bar{F}(x)$ is defined to be a n^2 dimensional vector by stacking the columns of $F(x)$ into a single column vector and $\bar{\Lambda}$ is so defined by stacking the columns of Λ into a single column vector.

Note that μ_{ij} for $i \in \mathbb{V}, j \in \mathbb{V} \,\text{and}\, j \notin \mathbb{N}(i)$ is a redundant variable designed for the compactness of (7). Since $F_{ij}(x)$ is zero and μ_{ij} is initialized to be zero for $j \notin \mathbb{N}(i)$, μ_{ij} will be zero all the time after initialization according to the dynamics of (7). With this in mind, the equivalence of the dynamic of the neural network (6) and its compact form (7) can be verified by expanding the right-hand side of (7) in entrywise and comparing it with the right-handside of (6).

3 Theoretical Results

In this section, we study the convergence of the neural network (6) and the solution feasibility to the original problem (1). Before starting up, we first state some useful conclusions for the proof.

Lemma 1 *The equilibrium point x_i^*, λ_{ij}^* for all $i \in \mathbb{V}$, $j \in \mathbb{N}(i) \bigcup i$ of the recurrent neural network (6) is a solution to the problem (1).*

Proof: The equilibrium point satisfies:

$$0 = \lambda_{ii}^* \frac{\partial f_{ii}(x_i^*, x_i^*)}{\partial x_i} + \sum_{j \in \mathbb{N}(i)} \Big(\lambda_{ij}^* \frac{\partial f_{ij}(x_i^*, x_j^*)}{\partial x_i} + \lambda_{ji}^* \frac{\partial f_{ji}(x_j^*, x_i^*)}{\partial x_i} \Big) \; \forall i \in \mathbb{V} \tag{10a}$$

$$0 = g\Big(f_{ij}(x_i^*, x_j^*) + \lambda_{ij}^*\Big) - \lambda_{ij}^* \quad \text{for}\, j \in \mathbb{N}(i) \bigcup \{i\}, \forall i \in \mathbb{V} \tag{10b}$$

With the definition of $g(\cdot)$ in (5), (10b) is equivalent to the following,

$$f_{ij}(x_i^*, x_j^*) \begin{cases} = 0 & (\lambda_{ij}^* > 0) \\ < 0 & (\lambda_{ij}^* = 0) \end{cases} \quad \text{for}\, j \in \mathbb{N}(i) \bigcup \{i\}, \forall i \in \mathbb{V} \tag{11}$$

(11) and (10a) together form the KKT condition for problem (2). As the function J and the constraint $f_{ij}(x_i, x_j) \leq 0$ for $j \in \mathbb{N}(i) \bigcup \{i\}$, $i \in \mathbb{V}$ are convex, the KKT condition is necessary and sufficient for problem (2) and the equilibrium point x_i^*, λ_{ij}^* for all $i \in \mathbb{V}$, $j \in \mathbb{N}(i) \bigcup i$ is an optimal solution to the

problem. Further, resulting from the equivalence of problem (1) and problem (2), we conclude that the equilibrium point x_i^*, λ_{ij}^* for all $i \in \mathbb{V}$, $j \in \mathbb{N}(i) \bigcup i$ indeed satisfies problem (1). This completes the proof. ■

For a general projection neural network, the following lemma holds,

Lemma 2 ([?]) *Assume that $\nabla h_1(x)$ is positive semi-definite on Ω and $h_2(x)$ is convex on Ω. Then the following dynamic system (12) with any initial point $(x_1(0), x_2(0), x_3(0)) \in \Omega \times {\mathbb{R}_+}^m \times \mathbb{R}^r$ is stable in the sense of Lyapunov and converges exponentially to its equilibrium point.*

$$\begin{aligned}
\dot{x}_1 &= -\epsilon\Bigg(x_1 - P_\Omega\Big(x_1 - h_1(x_1) - (\nabla^T h_2(x_1))x_2 - D^T x_3\Big)\Bigg) \\
\dot{x}_2 &= -\epsilon\Bigg(x_2 - g\Big(x_2 + h_2(x_1)\Big)\Bigg) \\
\dot{x}_3 &= -\epsilon(Dx_1 - d) \qquad (12)
\end{aligned}$$

where x_1, x_2 and x_3 are state variables of the dynamic system, $P_\Omega(x)$ is the Euclidean projection of x onto the set Ω defined as $P_\Omega(x) = argmin_{y\in\Omega}\|x - y\|$ with $\|\cdot\|$ denoting the Euclidean norm, $\epsilon > 0$ is a scaling factor, x_1, x_2 and x_3 are vector variables of appropriate sizes, d and D are vector and matrix of appropriate sizes, respectively, $g(\cdot)$ is as defined in (5).

The dynamics of (7) is identical to (12) by choosing $D = 0$, $d = 0$, $h_1(\cdot) = 0$, $h_2(\cdot) = \nabla\bar{F}(\cdot)$, $\Omega = \mathbb{R}^k$. Based on this observation, we have the following theorem,

Theorem 1 *The recurrent neural network (6), with $f_{ij}(x_i, x_j)$ convex for $j \in \mathbb{N}(i) \bigcup \{i\}, \forall i \in \mathbb{V}$, $\epsilon > 0$ and the initial value of $\lambda_{ij}(0) \leq 0$ for all $i \in \mathbb{V}$ and $j \in \mathbb{E} \bigcup \{i\}$, exponentially converges to a solution of problem (1).*

Proof: By choosing $D = 0$, $d = 0$, $h_1(\cdot) = 0$, $h_2(\cdot) = \nabla\bar{F}(\cdot)$, $\Omega = \mathbb{R}^k$, with noting that $P_\Omega(x) = x$ in this case, the dynamic system (12) reduces to the following,

$$\begin{aligned}
\dot{x}_1 &= -\epsilon\Big(\nabla^T \bar{F}(x_1)\Big)x_2 \\
\dot{x}_2 &= \epsilon\Bigg(g\Big(x_2 + h_2(x_1)\Big) - x_2\Bigg) \\
\dot{x}_3 &= 0 \qquad (13)
\end{aligned}$$

Note that the function $\bar{F}(x_1)$ is convex relative to x_1 resulting from the convexity of $f_{ij}(x_i, x_j)$ for $j \in \mathbb{N}(i) \bigcup \{i\}, \forall i \in \mathbb{V}$. For system (13), $\nabla h_1(x) = 0$ is positive semi-definite on $\Omega = \mathbb{R}^k$. According to Lemma 2, the dynamic system

(13), with the initial value of $\lambda_{ij}(0) \leq 0$ for all $i \in \mathbb{V}$ and $j \in \mathbb{E} \bigcup i$, is stable in the sense of Lyapunov and converges exponentially to its equilibrium point. Noticing that the dynamic of x_1 and x_2 have no dependence on x_3 in (13), we conclude the following system (14), which describes the dynamic of x_1 and x_2, is also stable and exponentially converges to its equilibrium point under the condition that $\lambda_{ij}(0) \leq 0$.

$$\begin{aligned}\dot{x}_1 &= -\epsilon\Big(\nabla^T \bar{F}(x_1)\Big)x_2 \\ \dot{x}_2 &= \epsilon\Bigg(g\Big(x_2 + h_2(x_1)\Big) - x_2\Bigg)\end{aligned} \tag{14}$$

Replacing the variable x_1, x_2 with x and $\bar{\Lambda}$, system (14) changes to system (7) and we conclude the Lyapunov stability and exponential convergence of (7) and so does its equivalent system (6). With Lemma 1, we conclude that the neural network (6), with the initial value of $\lambda_{ij}(0) \leq 0$ for all $i \in \mathbb{V}$ and $j \in \mathbb{E} \bigcup i$, exponentially converges to a solution of problem (1). This completes the proof. ■

As claimed in Remark 1, communication only happens between neighbor nodes for information exchange. As such, no cross-hop communication exists in the network and the communication protocol can be thoroughly tailored to save time for communication. For the case without taking communication time into account, the iterative time is purely dependent on the neural network structure and can be reduced by increasing the value of the scaling factor ϵ. For the case that the communication time is not negligible, we have the following remark.

Remark 2 *For the case with non-negligible communication time, the communication time introduces delay in the dynamics of the neural network as the received information from neighbors is the version τ seconds ago (say τ is the communication time in seconds) instead of the current one. In such a situation, the convergence conditions can be obtained by solving linear matrix inequalities as done in [?]. Coarsely speaking, the total iterative time in the case with non-negligible communication time can be approximately measured by the summation of the communication time for the required number of iterations and the computation time taken by the neural network.*

4 Solving Range-free Localization of WSNs

To demonstrate the effectiveness of the proposed neural network, we apply this model to solve the range-free localization problem in WSNs. WSN Localization refers to determine the positions of blind sensor nodes (locations of the blind nodes are unknown) with known positions of beacon nodes (the positions of beacon nodes are known perhaps by GPS or deliberate placement) and sensing information, such as time of arrival (TOA) [?], time difference of arrival (TDoA) [?], received signal strength (RSS) [?], angle of arrival (AoA) [?]. Range-based localization needs range measurements and uses range information between neighbor nodes for location inferences. In contrast, range-free localization, which relies on the proximity information provided by the communication topology for localization and does not require any extra ranging sensors, thoroughly reduces the hardware cost and has attracted intensive studies. In [?], a novel range-free based convex method, which can effectively solve the problem when infeasible points occur, is presented to solve position estimation problems. Based on heading data generated by a variety of orientation-tracking sensors, a Monte Carlo sampling technique is used in [?] to solve the range-free localization problem of WSNs. In [?], a sequential greedy optimization algorithm is proposed to solve both range-based localization and range-free localization in a distributed way. A range-free cooperative localization algorithm for mobile sensor networks is presented in [?] by combining hop distance measurements and particle filtering. We refer the reader to paper [?,?] and citations therein for a complete knowledge on range-free localization in WSNs. According to the difference of places for algorithm execution, existing methods can be categorized into centralized methods and distributed methods. For centralized methods, such as the linear matrix inequality (LMI) based method [?], proximity information from every nodes is routed to a base station and the base station processes all the data to output the position estimation of every nodes. As all the computations are run on the base station, the base station is required to be powerful in computation for real-time localization. In addition, this kind of method is vulnerable to the failure of the base station. For distributed methods, such as the DV-HOP method [?], APIT method [?], the computation is distributed to every nodes in the network. Therefore, the computation burden for each node is relatively low and the method is robust to node failures. Routing or even flooding of proximity information is required for centralized methods and some distributed methods (e.g., the DV-HOP method [?]). This greatly increases communication burdens and also increases energy consumptions. In contrast, the proposed neural network approach for WSN localization is a routing-free method, for which each node only needs information from its one-hop neighbors but still completes the task ideally as does by the centralized method.

In this section, we take the following range-free localization modeling of the problem [?]:

$$\|x_i - x_j\| \leq R \Longleftrightarrow \begin{bmatrix} I_k R & x_i - x_j \\ (x_i - x_j)^T & R \end{bmatrix} \geq 0 \quad \text{for}\, j \in \mathbb{N}(i) \tag{15}$$

where R is the maximum communication range, x_i, x_j are the position of node i and node j, respectively, I_k is a $k \times k$ identity matrix with k representing the dimension of variable x. This model is proposed and solved numerically in [?] and has received insensitive attentions in the past ten years. In [?], the problem is solved via a centralized LMI based method and some of its later improvements [?, ?, ?] also bear a centralized computation fashion. In this part, we put (15) into the framework of problem (1) by properly defining the function f_{ij} and solve the problem using the proposed neural network model (6). Defining the graph $\mathcal{G}(\mathbb{V}, \mathbb{E})$ as the communication topology constructed by the WSN with $\mathbb{V}$ being the set composed of all sensor nodes and $\mathbb{E}$ being all connected communication links in the network, and defining the function $f_{ij}(x_i, x_j) = (x_i - x_j)^T(x_i - x_j) - R^2$ for $j \in \mathbb{N}(i), \forall i \in \mathbb{V}$ and $f_{ii}(x_i, x_i) = 0$ for $\forall i \in \mathbb{V}$, the problem (1) reduces to problem (15). We summarize the relation between the problem (1) and the problem of range-free localization of WSNs in the following remark,

Remark 3 *By choosing $f_{ij}(x_i, x_j) = (x_i - x_j)^T(x_i - x_j) - R^2$ in (1), the problem of range-free localization defined in (15) falls into the unified framework given in problem (1) and thereby can be efficiently solved using the proposed recurrent neural network approach. Other applications with networked constraints can be solved in a similar procedure if they fall into the framework presented in (1).*

Noting that $\frac{\partial f_{ij}(x_i,x_j)}{\partial x_i} = \frac{\partial f_{ji}(x_j,x_i)}{\partial x_i} = 2(x_i - x_j)$, the neural network (6) for solving this particular problem has the following form,

$$\begin{aligned} \dot{x}_i &= 2\epsilon \sum_{j\in\mathbb{N}(i)} (\lambda_{ij} + \lambda_{ji})(x_i - x_j) \ \forall i \in \mathbb{V} \\ \dot{\lambda}_{ij} &= \epsilon g\Big((x_i - x_j)^T(x_i - x_j) - R^2 + \lambda_{ij}\Big) - \epsilon\lambda_{ij} \quad \text{for}\, j \in \mathbb{N}(i) \bigcup\{i\}, \forall i \in \mathbb{V} \end{aligned} \tag{16}$$

Directly following Theorem 1, we have the convergence conclusion for the neural network (16) stated as follows.

Corollary 1 *The recurrent neural network (16), with $\epsilon > 0$ and the initial value of $\lambda_{ij}(0) \leq 0$ for all $i \in \mathbb{V}$ and $j \in \mathbb{E} \bigcup\{i\}$, exponentially converges to a solution of the WSN range-free localization problem (15).*

Proof: Apparently, the function $f_{ij}(x_i, x_j) = (x_i - x_j)^T(x_i - x_j) - R^2$ is convex for $j \in \mathbb{N}(i), \forall i \in \mathbb{V}$, so the condition for Theorem 1 holds. This leads to the conclusion. ■

5 Hardware Implementation of the Neural Network

The neural network is composed of many modules, each of which is associated with a blind sensor node in the network. The modules interact with their neighbors and all the modules together perform the localization task and solve the problem (15). Different from traditional iterative methods for localization of WSNs, which run an algorithm in series, the proposed neural network can be implemented in analog circuits and accordingly processes signals in parallel and solve the problem in real time. As an example, Fig. 1 sketches the implementation architecture with analog devices of the neural module associated with the first node (i.e., $i = 1$), where $j1$, $j2$, ..., jk denote the neighbors of the first node. From Fig. 1, we can see that summators, multipliers, linear amplifiers, nonlinear amplifiers (for the implementation of the function $g(\cdot)$) and integrators are employed in the implementation. The neural module gets input from its neighbor nodes and outputs its own position estimation. Equipped with such a hardware module, sensor nodes are able to localize themselves with proximity information.

On the complexity of the proposed approach, we have the following remark.

Remark 4 *Each neural module associates with a sensor node. It only interacts with its neighbors. The introduction of extra sensor nodes outside the neighborhood has no direct impact on this particular neural module. Consequently, the convergence of the neural dynamics of a particular neural module is independent of the total number of sensors in the network (instead, it has dependance on the number of neighbors, i.e., the degree of a node on the graph in the terminology of graph theory). Therefore, the time complexity of the proposed approach is $O(1)$ for its circuit implementation. As to the spatial complexity, we can find that each neural module (as shown in Fig.* **??***) consists of a limit number of analog devices and therefore the spatial complexity is $O(n)$ (n denotes the number of nodes in the sensor network), meaning that the total consumption of analog devices is linear dependent on the number of sensor nodes in the network.*

6 Simulation Results

In this section, two simulation examples are performed to show the effectiveness of the proposed approach for WSN localization. Without loss of generality, we consider the 2-dimensional case.