Neural Network Output Optimization Using Interval Analysis

E. de Weerdt, Q. P. Chu, and J. A. Mulder

***Abstract*—The problem of output optimization within a specified input space of neural networks (NNs) with fixed weights is discussed in this paper. The problem is (highly) nonlinear when nonlinear activation functions are used. This global optimization problem is encountered in the reinforcement learning (RL) community. Interval analysis is applied to guarantee that *all* solutions are found to any degree of accuracy with guaranteed bounds. The major drawbacks of interval analysis, i.e., dependency effect and high-computational load, are both present for the problem of NN output optimization. Taylor models (TMs) are introduced to reduce these drawbacks. They have excellent convergence properties for small intervals. However, the dependency effect still remains and is even made worse when evaluating large input domains. As an alternative to TMs, a different form of polynomial inclusion functions, called the polynomial set (PS) method, is introduced. This new method has the property that the bounds on the network output are tighter or at least equal to those obtained through standard interval arithmetic (IA). Experiments show that the PS method outperforms the other methods for the NN output optimization problem.**

***Index Terms*—Feedforward neural networks (FFNNs), global optimization, inclusion function, interval analysis, optimization methods, polynomial set, radial basis function neural networks (RBFNNs), Taylor expansion, Taylor model (TM).**

I. Introduction

NEURAL networks are general function approximators that are widely used in the scientific community. Applications in business [1], [2], pattern recognition and classification [3]–[6], statistical modeling [7], and many other research fields provide an enormous amount of information on neural networks (NNs), which comes in many forms and sizes. There are self-organizing maps, recurrent networks, feedforward neural networks (FFNNs), radial basis function neural networks (RBFNNs), and many others [8]. The origin of artificial NNs lies in 1943 when a first attempt was made by mathematicians McCulloch and Pitts to mimic the neurons in the human brain [9]. This opened a new research field although it was not until 1974 with the invention of error-back propagation learning by Werbos that the NNs became more widely applicable [10]. Since then many new NN forms and learning algorithms have been developed. Although great advances have been reported in the past, there still remain some challenges. An example being the global optimization of network structure and weights given a certain data set. Various methods have been proposed that can be divided into two categories: probabilistic methods and deterministic methods [11]. Probabilistic methods, such as global descent [12] and the hybrid method of Baba *et al.* [13], guarantee to find the global optimal weight set provided that the number of iterations can go to infinity. Since the optimization problem must be solved within finite time, one cannot guarantee to (always) find the global optimum weight set (unless one knows the cost function value of the global minimum). Moreover, it has been shown in [14] that many global optimization methods of probabilistic nature have similar performance to standard gradient methods but have much larger computational costs.

Deterministic optimization method can guarantee to find the global optimum within finite time. However, there are few optimization methods available in literature which are truly deterministic *global* optimization methods. Methods as proposed by Toh [15], which use a penalty function that determines a correct outcome, or the NOVEL method proposed by Shang and Wah [11], which uses search trajectories to circumvent uninteresting regions, do not have an absolute guarantee to find the global minimum. So-called covering methods do have this guarantee and can moreover provide bounds on the found solution(s). The method of Tang and Koehler [16] based on Lipschitz bounds is such a method. Li *et al.* [17] use a method based on interval analysis to provide guaranteed bounds on the solutions. The latter method applies, among other things, gradient inclusion to cutoff uninteresting regions of the search space similar to the method of Tang and Koehler.

Instead of minimizing an error cost, in this paper, we solve the problem of global maximization of NN output, for networks with a fixed set of weights, within a given input space, i.e., find the input $\mathbf{x} \in \mathbf{X}$, which lead to the maximum output value

$$\max_{\mathbf{x}\in\mathbf{X}} \mathrm{NN}(\mathbf{x}, \mathbf{W}) \tag{1}$$

where $\mathrm{NN}(\mathbf{x}, \mathbf{W})$ is an NN where $\mathbf{x}$ is the input vector, $\mathbf{X}$ is the total input space, and $\mathbf{W}$ is the set of network weights. This problem becomes a (highly) nonlinear optimization problem with many local maxima and minima when the NN uses nonlinear activation functions (see Fig. 1). The need to solve this problem stems from the reinforcement learning (RL) topic [18]. RL is an optimal control technique in which the agent (or agents) learns to control a system by interaction with the environment or with a model of the environment [19]. To derive the optimal control law the agent must process and store the gained knowledge. For Q-learning and advantage learning, this knowledge is

Manuscript received November 13, 2007; revised November 03, 2008; accepted December 02, 2008. First published February 24, 2009; current version published April 03, 2009. This work is part of the MicroNed MISAT project.

The authors are with the Control and Simulation Division, Faculty of Aerospace Engineering, Delft University of Technology, 2629 HG Delft, The Netherlands (e-mail: E.deWeerdt@TUDelft.nl; Q.P.Chu@TUDelft.nl; J.A.Mulder@TUDelft.nl).

Digital Object Identifier 10.1109/TNN.2008.2011267

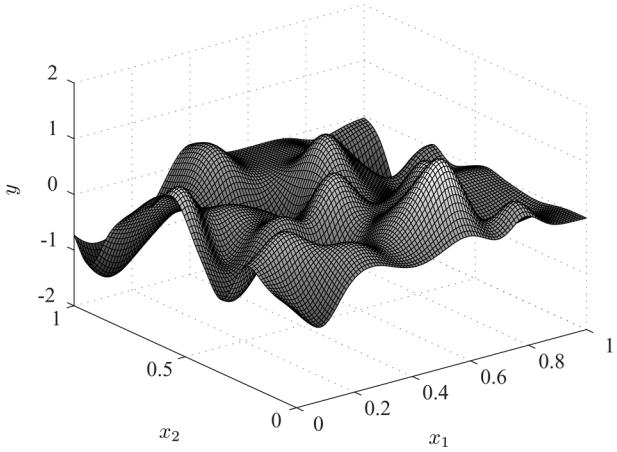

Fig. 1. Example of an input–output mapping of a 2-64-1 radial basis function network.

stored as a mapping from states and actions to expected accumulated future rewards, i.e., the state-action value function [20]

$$Q^{\pi}(\mathbf{s_t}, \mathbf{u_t}) = \mathrm{E}\left\{\sum_{i=t}^{\infty} \gamma^i r_i\right\}$$
$$\text{with} \quad \mathbf{u_i} = \pi(\mathbf{s_i}), \qquad \mathbf{i} = \mathbf{t}, \mathbf{t+1}, \ldots, \infty \tag{2}$$

where $\mathbf{s}$ is the state vector, $\mathbf{u}$ is the action vector, γ is a discount factor $[0, 1]$, r is the reward signal, and π is the policy (mapping between state and actions). For most problems, this function is a nonlinear input–output mapping whose shape is not known beforehand.

NNs are commonly applied in RL techniques [21] to represent the state-action value function. NNs circumvent the "curse of dimensionality" since they are capable of providing a continuous nonlinear approximation of (in theory) any continuous function to any degree of accuracy [22] in an efficient manner, i.e., the number of parameters is limited compared to other storing methods. When applying Q-learning (or advantage learning), the agent uses an update rule to store the gained information in the state-action value function. To compute the weight update, one needs to determine which action set within the action input space U leads to the maximal NN output value given the future state and network weights [18]. Furthermore, when applying RL, the applied policy is (at some stage) a greedy policy meaning that we apply those actions which lead to the maximal function output. It poses the same global optimization problem. In practice, the posed problem has to be solved each time step, which requires an efficient optimization algorithm. The accuracy of the found solution, in both the control input space and the output space, influences the outcome of the Q-learning algorithm and the performance of the agent. Moreover, *all* solutions leading to the maximal network output must be found such that the agent knows the complete set of optimal actions. It also helps in the exploration aspect of RL.

To solve the problem of finding the maximal network output one can use any covering method which provides guaranteed bounds (as presented earlier). In this paper, we propose to use interval analysis (IA) to solve the problem. IA is a global optimization technique developed in the 1960s by Moore [23], which finds *all* solutions to *any required degree of accuracy* with *guaranteed bounds*. These properties make interval analysis very attractive for solving the stated problem. In the past, the applicability of interval analysis was limited because it can be computationally expensive. However, currently, this research field is rapidly growing due to the increasing computation power of personal computers. Examples of applications range from nonlinear aircraft trim [24] to reentry flight clearance [25], integer ambiguity resolution [26], stock market forecasting [27], data analysis [28], and many more (see also the *Journal of Reliable Computing*). In Section II, we will provide more information regarding interval analysis. The inclusion function theory on which interval analysis is based is given and the standard branch-and-bound algorithm is explained. The branch-and-bound algorithm is enhanced using the same improvements as given in [17] such that we can compare our new implementation to their method. In Section II, we also discuss several key drawbacks of interval analysis which, in the specific case of NN output optimization, lead to severe degradation of algorithm efficiency. In Section III, Taylor model (TM) inclusion functions are introduced in an attempt to remove the drawbacks and speed up the process. TMs have been developed by Lanford around 1980, subsequently studied by Eckmann, Koch, Wittwer, Berz, Makino, and Hoefkens [29]–[32], and have been successfully applied to many problems [33], [34]. However, for the current optimization problem, we will show that these models suffer from remainder blowup. We will combine TM with standard interval analysis to prevent the remainder blowup. A different approach to solving the drawbacks of standard interval analysis is given in Section IV, where we introduce another form of polynomial inclusion functions called the polynomial set (PS) method. Unlike TM, the PS method is guaranteed to have bounds equal to or tighter than the bounds provided by standard interval analysis. Simulation results for all described techniques are given in Section V followed by conclusions in Section VI.

For this paper, we handle two types of NNs: single-hidden-layer FFNNs with tangent sigmoidal activation functions and RBFNNs

$$\begin{aligned}
\text{FF} \quad y_k(x) &= \sum_j w_{jk}\left(\frac{2}{1+\exp(-2v_j)} - 1\right) + b_k \\
v_j &= \sum_i \{w_{ij}x_i\} + b_j \\
\text{RBF} \quad y_k(x) &= \sum_j w_{jk}\exp\left(-v_j^2\right) \\
v_j^2 &= \sum_i \left\{w_{ij}^2(x_i - c_{ij})^2\right\}
\end{aligned} \tag{3}$$

where

- y_k output k;
- x_i input i;
- w_{jk} output weight: from hidden-layer neuron j to output k;
- w_{ij} input weight: from input i for hidden-layer neuron j

b_k bias weight for output k;

b_j bias weight for hidden-layer neuron j;

c_{ij} center weight: hidden-layer neuron j for input dimension i.

II. Interval Analysis

In standard mathematics, crisp numbers, either real or imaginary, are used to perform crisp arithmetic operations. With interval mathematics, we use interval numbers that represent a set of crisp numbers, i.e., the interval number $[a, b]$ is the set of crisp numbers x such that $a \leq x \leq b$. Interval numbers become equal to crisp numbers when the interval has zero width, e.g., $x = 6, X = [6, 6]$. Variables representing an interval number are indicated with capital letters whereas variables representing crisp numbers are written in lowercase. When dealing with interval numbers the arithmetic operations change. Some examples are as follows ($X = [a, b]; Y = [c, d]$[1]):

addition

$$X + Y = [a + c, b + d]; \tag{4}$$

subtraction

$$X - Y = [a - d, b - c]; \tag{5}$$

multiplication

$$X \times Y = [\min(ac, ad, bc, bd), \max(ac, ad, bc, bd)]; \tag{6}$$

division (excluding division by an interval Y containing 0)

$$\frac{X}{Y} = X \times \left(\frac{1}{Y}\right) \quad \text{with} \quad \frac{1}{Y} = \left[\frac{1}{d}, \frac{1}{c}\right]. \tag{7}$$

For a complete overview of interval arithmetic (IA) operands, see [35]. The key aspect of interval analysis is the ability of providing bounds on the output of any rational function with inputs within a specified domain. This property of IA follows from the inclusion function theorem given by Moore [23], [36]:

Theorem II.1: If $F(X_1, X_2, \ldots, X_n)$ is a rational expression in the interval variables $X_1, X_2, \ldots, X_n$, i.e., a finite combination of $X_1, X_2, \ldots, X_n$ and a finite set of constant intervals with IA operations, then

$$X_1' \subset X_1, X_2' \subset X_2, \ldots, X_n' \subset X_n \tag{8}$$

implies

$$F(X_1', X_2', \ldots, X_n') \subset F(X_1, X_2, \ldots, X_n) \tag{9}$$

for every set of interval numbers $X_1, X_2, \ldots, X_n$ for which the IA operations in F are defined.

Proof: For the proof of this theorem, the reader is directed to [36]. ■

If we take $X_1', X_2', \ldots, X_n'$ to be crisp numbers $x_1, x_2, \ldots, x_n$ and apply the theorem, we have

$$f(x_1, x_2, \ldots, x_n) \subset F(X_1, X_2, \ldots, X_n) \tag{10}$$

[1]The lowest bound of an interval number is always written first, i.e., $a = \inf(X)$

TABLE I
Standard Branch-and-Bound Algorithm

0.	Initialization: put initial box $\mathbf{X}_0$ into list L
DO WHILE boxes are left in list L	
1.	Take first box $\mathbf{X}$ out of list L and remove it from the list
2.	Perform function evaluation $NN(\mathbf{X}, \mathbf{W})$
3.	If function output fulfills all criteria
	If function output and box fulfill stopping criteria
	Save box in list of solutions and proceed with next box
	Else
	Split box and add all sub-boxes to list L
	End
	Else
	Delete box and go to 1
	End
END	

for $x_1 \subset X_1, x_2 \subset X_2, \ldots, x_n \subset X_n$. This means that if we replace all variables x with their interval counterparts X, we can compute guaranteed bounds on any rational function for $\mathbf{x} \in \mathbf{X}$. Theorem II.1 is the foundation of interval analysis and we can use this important property of function evaluations to obtain global optimization algorithms.

Using Theorem II.1, we can compute guaranteed bounds on the output of an NN for any input space $\mathbf{X}$

$$\mathrm{NN}(\mathbf{x}, \mathbf{W}) \subset \mathrm{NN}(\mathbf{X}, \mathbf{W}) \qquad \forall \mathbf{x} \in \mathbf{X}. \tag{11}$$

The obtained bounds on the output can be used to determine whether the corresponding input space contains the maximal output value.

Corollary II.2: If for $\mathbf{P} \subset \mathbf{X}$ and $\mathbf{Q} \subset \mathbf{X}$, we have $\sup \mathrm{NN}(\mathbf{P}, \mathbf{W}) < \inf \mathrm{NN}(\mathbf{Q}, \mathbf{W})$, then the subset $\mathbf{P}$ is guaranteed *not* to contain any solution of

$$\max_{\mathbf{x} \in \mathbf{X}} \mathrm{NN}(\mathbf{x}, \mathbf{W}). \tag{12}$$

Proof: Theorem II.1 states that $\mathrm{NN}(\mathbf{x}) \in \mathrm{NN}(\mathbf{X}) \; \forall \, \mathbf{x} \in \mathbf{X}$. Therefore, $\sup \mathrm{NN}(\mathbf{P}, \mathbf{W}) < \inf \mathrm{NN}(\mathbf{Q}, \mathbf{W})$ implies $\mathrm{NN}(\mathbf{x}, \mathbf{W}) < \mathrm{NN}(\mathbf{x}, \mathbf{W}) \forall \, \mathbf{x} \in \mathbf{P}, \mathbf{x} \in \mathbf{Q}$ and thus the location of the maximum of $\mathrm{NN}(\mathbf{x}, \mathbf{W})$ cannot lie in $\mathbf{P}$. ■

A. Basic Branch-and-Bound Algorithm

The basic optimization algorithm to find the location and value of the maximal output is called branch-and-bound. The general setup of the basic branch-and-bound algorithm is given in Table I. If the task at hand is the computation of the maximal output of $\mathrm{NN}(\mathbf{x}, \mathbf{W})$, we replace the steps in Table I with the following.

- Step 2: Function evaluation is the computation of $\mathrm{NN}(\mathbf{X}, \mathbf{W})$.
- Step 3: The criterion is that $\sup \mathrm{NN}(\mathbf{X}, \mathbf{W}) > \rho$ where ρ is the current estimate of the maximal output value. It is equal to the highest $\inf \mathrm{NN}(\mathbf{X}, \mathbf{W})$ of all the boxes in the list L. The stopping criterion is commonly defined as $|\mathbf{X}| < \sigma_x$, where σ_x is some minimal width per dimension of $\mathbf{X}$ or when $|\mathrm{NN}(\mathbf{X}, \mathbf{W})| < \sigma_y$.
- Splitting of the box is commonly done by bisecting the box in the dimension that has the largest width.

To clarify the basic algorithm, we will discuss a simple example.

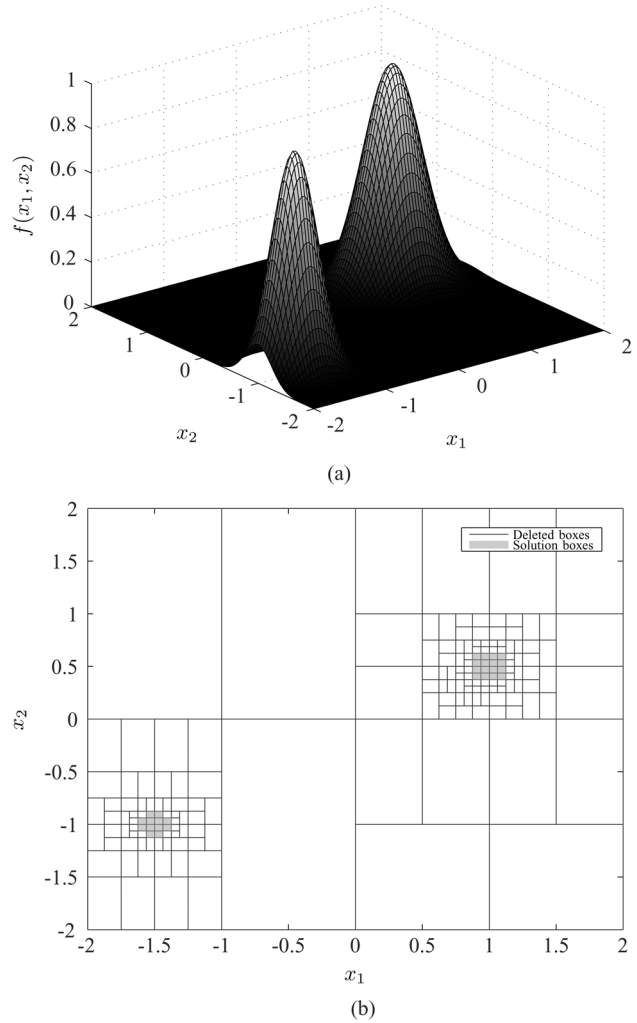

Fig. 2. Function mapping and algorithm output for simple example branch-and-bound algorithm. (a) Function output. (b) Algorithm output.

1) Example Branch-and-Bound: Consider a 2-2-1 radial basis function network

$$\begin{aligned}\mathrm{NN}(x_1, x_2) = &\exp(-4(x_1-1)^2 - 6(x_2-0.5)^2) \\ &+ \exp(-7(x_1+1.5)^2 - 9(x_2+1)^2), \\ &x_1 \in [-2, 2], \qquad x_2 \in [-2, 2]. \end{aligned} \tag{13}$$

The function is plotted in Fig. 2(a). The maximal output value is 1 and the input set $\{x_1, x_2\}$ leading to this value is located at the center of both neurons. If we want to find the maximal value and its location, we apply the branch-and-bound algorithm with initial conditions $\rho = 0, \sigma_y = 0.1$, and $\sigma_x = [0.1, 0.1]$, i.e., the best estimate of the maximal value is zero and the algorithm stops when all solution boxes are smaller than 0.1 in x_1 and x_2. The solution is plotted in Fig. 2(b). The accuracy of the solution can be increased by decreasing σ_x.[2] As we can see, the algorithm can find the maximal value and *both* locations where this value is attained with guaranteed bounds (see Table III).

□

Global optimization algorithms, such as the branch-and-bound algorithm, are based on the principle of box evaluation and box elimination, i.e., discarding regions where the solution is guaranteed not to lie. The key to an efficient algorithm lies therefore in the box evaluation: the higher the accuracy and the more useful information about the output shape within the interval, the more efficient the optimization algorithm will be. This is where the drawbacks of standard interval analysis lie. There are two issues.

1) Standard IA provides an inclusion of the function $f(\mathbf{x})$ on the domain $\mathbf{X}$. Although this inclusion function is critical for the implementation of any optimization algorithm, its information contents is limited: no information regarding the behavior of the function $f(x)$ of the domain $\mathbf{X}$ is given except that its value lies within the interval F. If more information is available, then it could be used to enhance the efficiency of the algorithm, e.g., by enabling faster box elimination or box contractions.
2) IA has the following properties:
   - associative: $X \times (Y \times Z) = (X \times Y) \times Z$;
   - commutative: $X \times Y = Y \times X$;
   - nondistributive: $X \times (Y + Z) \neq X \times Y + X \times Z$ but $X \times (Y + Z) \subset X \times Y + X \times Z$.

   The nondistributive property can lead to the so-called *dependency* effect. When evaluating a function $f(x)$ over domain $\mathbf{X}$, each occurrence of X_i is treated as an independent interval parameter. This can result in an inclusion function $F(\mathbf{X})$, which has nontight bounds, i.e., $\sup F > f(\mathbf{x})$ and $\inf F < f(\mathbf{x})\ \forall \mathbf{x} \in \mathbf{X}$. A simple example is the function $f(x) = x - x$, which is equal to zero on any domain. Interval evaluation leads to

$$X - X = [a - b, b - a] \neq [0, 0], \qquad \text{for } a \neq b. \tag{14}$$

   The dependency effect can expand the bounds of the inclusion function, therefore creating less accurate information, which lowers the efficiency of the algorithm.

Unfortunately, the dependency effect is very hard to eliminate. If possible, one could rewrite the optimization problem such that the number of occurrences of each variable is as low as possible. For the current NN output optimization algorithm, this is not possible. We therefore focus on increasing the amount of useful information obtained during a function evaluation.

B. Enhanced Branch-and-Bound Algorithm

In this paper, we apply three additional steps to enhance the efficiency of the algorithm.

- **Midpoint evaluations**. By evaluating the NN at midpoint in the box $\mathbf{X}$,[3] we can possibly update the current estimate of the maximal output value

$$\rho = \max(\rho, \mathrm{NN}(\mathrm{mid}(\mathbf{X}), \mathbf{W})). \tag{15}$$

  Since we have significant dependency in the interval evaluations, this crisp evaluation step can quickly drive the estimate of the maximal value to its final value.
- **Derivative evaluations**. For the NN, we can evaluate its derivatives with respect to x using IA. The result is a guaranteed inclusion of the derivative value within box $\mathbf{X}$.

[2]We have kept σ_x large to make the example clearer.

[3]One can choose any point in box $\mathbf{X}$, but for convenience, we select the midpoint.

TABLE II
ENHANCED BRANCH-AND-BOUND ALGORITHM

0.	Initialization: put initial box $\mathbf{X}_0$ into list L, set $\rho = -\inf$
DO WHILE boxes are left in list L	
1.	Take first box $\mathbf{X}$ out of list L and remove it form the list
2.	Perform function evaluation $NN(\mathbf{X}, \mathbf{W})$ If $\sup NN(\mathbf{X}, \mathbf{W}) < \rho$; Delete box, goto 1; end
3.	Update ρ: $\rho = \max[\rho, \inf NN(\mathbf{X}, \mathbf{W})]$
4.	Perform mid-point evaluation and update ρ: $\rho = \max[\rho, NN(\text{mid}(\mathbf{X}), \mathbf{W})]$
5.	Compute derivatives $\partial NN/\partial x_i$ and apply rules from equations (16)-(18). If box can be deleted goto 1.
6.	Contract box based using equation (19).
7.	If $\lvert\mathbf{X}_{new}\rvert < \sigma_x$ or $\lvert NN(\mathbf{X}, \mathbf{W})\rvert < \sigma_y$; store box as possible solution and goto 1; end
8.	If $\lvert\mathbf{X}_{new}\rvert < \alpha\lvert\mathbf{X}\rvert$; Add box to list L and goto 1; end
9.	Split box and add all sub-boxes to list L
END	

Since we are searching for a maximum, we can eliminate any box that does not contain zero in one or more of the derivative intervals. Care must be taken when the box lies on the border of the original search space. It could be that the maximal output value lies on the border of dimension i while the first-order derivative with respect to x_i is nonzero. In the following cases, we must keep the box (possibly reduce it).

— Keep entire box when

$$0 \in \left.\frac{\partial \text{NN}}{\partial x_i}\right|_{\mathbf{X}} \forall i. \tag{16}$$

— Keep box and reduce $X_q : X_q = \inf X_q$ when

$$0 \in \left.\frac{\partial \text{NN}}{\partial x_i}\right|_{\mathbf{X}} \forall i \neq q,\ \left.\frac{\partial \text{NN}}{\partial x_q}\right|_{\mathbf{X}} < 0,\quad \inf X_q = \inf X_{q,0}. \tag{17}$$

— Keep box and reduce $X_q : X_q = \sup X_q$ when

$$0 \in \left.\frac{\partial \text{NN}}{\partial x_i}\right|_{\mathbf{X}} \forall i \neq q,\ \left.\frac{\partial \text{NN}}{\partial x_q}\right|_{\mathbf{X}} > 0,\quad \sup X_q = \sup X_{q,0} \tag{18}$$

where $\mathbf{X}_0$ is the original search space. In all other cases, we can remove the box.

- **Box contractions**. By using the inclusion of derivatives and the current estimate of the maximal value, one can possibly contract the box $\mathbf{X}$. The method applied here is a Newton step method [37] and it applies to any dimension

$$\mathbf{X}_i = \mathbf{X}_i \cap \frac{[\rho, \infty] - \text{NN}(x_{i,0}, \hat{\mathbf{X}})}{\partial \text{NN}/\partial x_i|_{\mathbf{X}}} + x_{i,0} \tag{19}$$

where $\hat{\mathbf{X}}$ is the interval set $\mathbf{X}$ minus dimension i and $x_{i,0}$ is the expansion point somewhere in domain X_i. During the experiments, we select $x_{i,0}$ once as $\inf X_i$ and once as $\sup X_i$ during one iteration.

The additional steps presented here are added to the branch-and-bound algorithm and the complete algorithm is given in Table II.

Next we will look at TMs that are used to determine an inclusion function consisting of a crisp polynomial with an interval remainder. The shape of the crisp polynomial will give more information regarding the actual shape of the function within that interval. This information can be used for more efficient box evaluations and contractions.

III. TAYLOR MODEL METHOD

TM methods use TM expansions of order n about the point $\mathbf{x}_0$ within interval $\mathbf{X}$ with bounds on the remainder. The inclusion function obtained in this way is a polynomial inclusion function formed by a polynomial of order n and an interval which encloses the remainder. Taylor's formula with remainder is given by

$$f(x) = \sum_{v=0}^{n} \frac{f^{(v)}(x_0)}{v!}(x - x_0)^v + \frac{g(x) - g(x_0)}{g'(\xi)} \cdot \frac{f^{(n+1)}(\xi)}{n!}(x - \xi)^n \tag{20}$$

where g is an arbitrary function with nonvanishing derivative strictly between x_0 and x and ξ lies strictly between x_0 and x. The form of the remainder depends on the choice of the function g. When choosing

$$g(\bar{x}) = (x - \bar{x})^{n+1},\quad g'(\bar{x}) = -(n+1)(x - \bar{x})^n \tag{21}$$

we obtain the Lagrange's remainder

$$L_n = \frac{f^{(n+1)}(\xi)}{(n+1)!} \cdot (x - x_0)^{n+1} \tag{22}$$

where again ξ lies strictly between x_0 and x. For the multivariate case, we can use the following notation for the nth-order Taylor polynomial and corresponding remainder:

$$T_{f,\mathbf{x}}^{(n)}(\mathbf{x}) = \sum_{0 \le i_1 + \cdots + i_d \le n} \left\{ \frac{\partial^{i_1 + \cdots + i_d} f(\mathbf{x})}{\partial x_1^{i_1} \ldots \partial x_1^{i_d}} \cdot \frac{(x_1 - x_{0,1})^{i_1} \times \cdots \times (x_d - x_{0,d})^{i_d}}{i_1! \times \cdots \times i_d!} \right\}$$

$$L_n(\mathbf{x}) = \sum_{i_1 + \cdots + i_d = n+1} \left\{ \frac{\partial^{n+1} f(\xi)}{\partial x_1^{i_1} \ldots \partial x_1^{i_d}} \cdot \frac{(x_1 - x_{0,1})^{i_1} \times \cdots \times (x_d - x_{0,d})^{i_1}}{i_1! \times \cdots \times i_d!} \right\}. \tag{23}$$

By evaluating the remainder with IA, we can obtain a bound on the remainder, which, together with the crisp polynomial of order n, forms the inclusion function called a TM. First we will give the formal definition of a TM. Then, we will derive the TM for the FFNN and the RBFNN.

A. Taylor Model

The formal definition of a TM is taken from [34]:

Let f be $C^{(n+1)}$ on $D_f \subset R^v$, and $\mathbf{B} = [a_1, b_1] \times \cdots \times [a_v, b_v] \subset D_f$ be an interval box containing the point $\mathbf{x}_0$. Let T be the Taylor polynomial of f around the point $\mathbf{x}_0$.

We call the interval I an n-th order *Remainder Bound* of f on $\mathbf{B}$ if

$$f(\mathbf{x}) - T(\mathbf{x}) \in I, \qquad \text{for all } \mathbf{x} \in \mathbf{B} \tag{24}$$

We call the pair (T, I) an n-th order TM of f. The set of all remainder bounds is called the *Remainder* Family. Since all partial derivatives of f are continuous on the compact set $\mathbf{B}$, they are bounded there, and so is the $(n+1)$-th application of the directional derivative $(\mathbf{x}-\mathbf{x}_0)\bigtriangledown$. This entails that for all $\mathbf{x} \in \mathbf{B}$, the Lagrange remainder is bounded, and hence a finite remainder bound exists. Furthermore, since $f(\mathbf{x}) - T(\mathbf{x})$ is continuous, it assumes extrema at $\mathbf{x}_u, \mathbf{x}_l$. So $I = [f(\mathbf{x}_l) - T(\mathbf{x}_l), f(\mathbf{x}_u) - T(\mathbf{x}_u)]$ is a remainder bound; all other remainder bounds must contain I, and thus I is often called the *Optimal Remainder Bound*.

A nice property of TMs is that the remainder decreases with order $n+1$ as $x \to x_0$. To obtain the bounds on the remainder for the interval $X : x \in X$, we can use standard IA and insert X for x and ξ (since ξ lies somewhere between x and x_0 thus somewhere in the interval X). We obtain

$$\begin{aligned} I &= L_n(\mathbf{X}) \\ &= \sum_{i_1+\cdots+i_d=n+1} \left\{ \frac{\partial^{n+1} f(\mathbf{X})}{\partial x_1^{i_1} \ldots \partial x_1^{i_d}} \right. \\ &\qquad \left. \cdot \frac{(X_1 - x_{0,1})^{i_1} \times \cdots \times (X_d - x_{0,d})^{i_1}}{i_1! \times \cdots \times i_d!} \right\}. \end{aligned} \tag{25}$$

The width of I depends on the width of $\mathbf{X}$ and on the function $f(\mathbf{x})$. If the interval $\mathbf{X}$ is large then the multiplication $((X_1 - x_{0,1})^{i_1} \times \cdots \times (X_d - x_{0,d})^{i_1})/(i_1! \times \cdots \times i_d!)$ will cause a blowup of the remainder if the $(n+1)$th order derivative of f is nonzero. Another difficulty is the number of occurrences of $\mathbf{X}$ in the $(n+1)$th order derivative of f. As for any function, multiple occurrences of $\mathbf{X}$ can cause overestimation, which yields large remainder bounds. We will show that for the case of NNs the remainder can become extremely large due to the previously described effect.

B. RBFNN

First we will derive the TM for the RBFNN. We restrict ourselves to first- and second-order TMs because for these models we can easily determine location and value of the maximum–minimum value, i.e., $\max_{\mathbf{x} \in \mathbf{X}} \inf \mathrm{NN}(\mathbf{X}, \mathbf{W})$. For orders up to 2, we can also easily determine the interval $\tilde{\mathbf{X}}$ for which $\sup \mathrm{NN}(\mathbf{X}, \mathbf{W}) > \rho$, which we can use to contract $\mathbf{X}$: $\mathbf{X} = \mathbf{X} \cap \tilde{\mathbf{X}}$. The first-order TM is defined as (notation has been simplified)

$$\begin{aligned} \mathrm{NN}(\mathbf{X},\mathbf{W}) &= \left(T^{(1)}_{\mathrm{NN},\mathbf{x}}(\mathbf{x}), L_1(\mathbf{X}) \right) \\ T^{(1)}_{\mathrm{NN},\mathbf{x}}(\mathbf{x}) &= \mathrm{NN}(\mathbf{x}_0, \mathbf{W}) + \sum_i \left\{ \frac{\partial \mathrm{NN}(\mathbf{x}_0, \mathbf{W})}{\partial x_i} \cdot (x_i - x_{0,i}) \right\} \\ I_1(\mathbf{X}) &= \sum_i \sum_q \left\{ \frac{\partial^2 \mathrm{NN}(\mathbf{X}, \mathbf{W})}{\partial x_i \partial x_q} \cdot \frac{(X_i - x_{0,i})(X_q - x_{0,q})}{i!q!} \right\} \end{aligned} \tag{26}$$

and the second-order model is defined as

$$\begin{aligned} &\mathrm{NN}(\mathbf{X}, \mathbf{W}) \\ &\quad = \left(T^{(2)}_{\mathrm{NN},\mathbf{x}}(\mathbf{x}), L_2(\mathbf{X}) \right) \\ &T_{\mathrm{NN},\mathbf{x}^{(2)}(\mathbf{x})} \\ &\quad = \mathrm{NN}(\mathbf{x}_0, \mathbf{W}) \\ &\qquad + \sum_i \left\{ \frac{\partial \mathrm{NN}(\mathbf{x}_0, \mathbf{W})}{\partial x_i} \cdot (x_i - x_{0,i}) \right\} \\ &\qquad + \sum_i \sum_q \left\{ \frac{\partial^2 \mathrm{NN}(\mathbf{x}_0, \mathbf{W})}{\partial x_i \partial x_q} \cdot \frac{(x_i - x_{0,i})(x_q - x_{0,q})}{i!q!} \right\} \\ &I_2(\mathbf{X}) \\ &\quad = \sum_i \sum_q \sum_p \left\{ \frac{\partial^3 \mathrm{NN}(X, \mathbf{W})}{\partial x_i \partial x_q \partial x_p} \right. \\ &\qquad \left. \cdot \frac{(X_i - x_{0,i})(X_q - x_{0,q})(X_p - x_{0,p})}{i!q!p!} \right\}. \end{aligned} \tag{27}$$

To construct the first- and second-order TM, we need the derivatives up to order 2 and 3, respectively

$$\begin{aligned} &\mathrm{NN}(\mathbf{x}, \mathbf{W}) \\ &\quad = \sum_j w_{jk} \exp\left(-v_j^2\right) = \sum_j w_{jk} \phi_j \\ &\frac{\partial \mathrm{NN}(\mathbf{x}, \mathbf{W})}{\partial x_i} \\ &\quad = \sum_j \left\{ w_{jk} \phi_j \left[-2 w_{ij}^2 (x_i - c_{ij}) \right] \right\} \\ &\frac{\partial^2 \mathrm{NN}(\mathbf{x}, \mathbf{W})}{\partial x_q \partial x_i} \\ &\quad = \sum_j \left\{ w_{jk} \phi_j \left\{ \left[-2 w_{ij}^2 (x_i - c_{ij}) \right] \cdot \left[-2 w_{qj}^2 (x_q - c_{qj}) \right] \right. \right. \\ &\qquad \left. \left. + \gamma \left[-2 w_{qj}^2 \right] \right\} \right\} \\ &\frac{\partial^3 \mathrm{NN}(\mathbf{x}, \mathbf{W})}{\partial x_p \partial x_q \partial x_i} \\ &\quad = \sum_j \left\{ w_{jk} \phi_j \left\{ [[-2 w_{ij}^2 (x_i - c_{ij})] \cdot [-2 w_{qj}^2 (x_q - c_{qj})] \right. \right. \\ &\qquad + \gamma [-2 w_{ij}^2]] \cdot [-2 w_{pj}^2 (x_p - c_{pj})] \\ &\qquad + \delta [-2 w_{pj}^2 [-2 w_{qj}^2 (x_q - c_{qj})]] \\ &\qquad \left. \left. + \kappa [-2 w_{pj}^2 [-2 w_{ij}^2 (x_i - c_{ij})]] \right\} \right\} \end{aligned} \tag{28}$$

where

$$\begin{cases} \gamma = 1 & q = i \\ \gamma = 0 & q \neq i \end{cases} \quad \begin{cases} \delta = 1 & p = i \\ \delta = 0 & p \neq i \end{cases}$$
$$\begin{cases} \kappa = 1 & p = q \\ \kappa = 0 & p \neq q. \end{cases}$$

We can see from the derivatives that for this problem the number of occurrences of $\mathbf{X}$ increases with increasing order of derivatives. This results in overestimation of the remainder bounds

for interval set $\mathbf{X}$ in which the $(n+1)$th-order derivative of one or more neurons is nonzero.[4] To show the effect of dependency, in the remainder, we will look at a single-hidden-layer neuron output and the corresponding first-order TM [see Fig. 3(a)]. Note that the TM for the complete network is simply the summation of the TMs of each hidden-layer neuron, therefore any blowup for a single-hidden-layer neuron will also be in the total TM. For the single-hidden-layer neuron, we have evaluated three intervals with increasing diameter. We can clearly see that the remainder blowup is considerable for the larger intervals. For the smallest interval X_1, we have a fairly accurate TM inclusion function, which contains more information regarding the shape of $\mathrm{NN}(x)$ and has tighter bounds on part of the domain X compared to the inclusion provided by standard IA. The higher the derivative order is, there are more occurrences of $\mathbf{x}$, and the dependency effect is more severe [see Fig. 3(b)]. This means that if the $(n+1)$th-order derivative is nonzero on the domain $\mathbf{X}$ and $\mathbf{X}$ is not sufficiently small, the obtained inclusion function will become worse if the order of the TM is increased. This is a highly undesirable property and can have severe consequences for the efficiency of the optimization algorithm. In Section III-D, we will discuss a method to reduce the overestimation of the remainder. First, we will discuss the implementation of TMs for FFNNs.

C. FFNN

As for the RBFNN, we will restrict our discussion to first- and second-order TMs. In Section III-D, we will show that for these models, we can easily derive contraction rules that improve the algorithm efficiency. The first- and second-order TMs are given by (26) and (27), respectively. To derive the models, we need the multidimensional derivatives up to order 3

$$\begin{aligned}
\mathrm{NN}(\mathbf{x},\mathbf{W}) &= \sum_j w_{jk}\left(\frac{2}{1+\exp(-2v_j)}-1\right)=\sum_j w_{jk}\phi_j \\
\frac{\partial \mathrm{NN}}{\partial x_i} &= \sum_j w_{jk}w_{ij}\left[1-\phi_j^2\right] \\
\frac{\partial^2 \mathrm{NN}}{\partial x_i \partial x_q} &= \sum_j w_{jk}w_{ij}w_{qj}\left[(-2\phi_j)\left(1-\phi_j^2\right)\right] \\
&= \sum_j w_{jk}w_{ij}w_{qj}\left[2\phi_j^3-2\phi_j\right] \\
\frac{\partial^3 \mathrm{NN}}{\partial x_i \partial x_q \partial x_p} &= \sum_j w_{jk}w_{ij}w_{qj}w_{pj}\left[\left(6\phi_j^2-2\right)\left(1-\phi_j^2\right)\right] \\
&= \sum_j w_{jk}w_{ij}w_{qj}w_{pj}\left[-6\phi_j^4+4\phi_j^2-2\right] \qquad (29)
\end{aligned}$$

Again, we can see that the parameters $\mathbf{x}$ occur multiple times in the derivatives since ϕ_j is a function of $\mathbf{x}$. Therefore, we expect, as for the RBFNNs, that the remainder will expand in case of larger interval sets $\mathbf{X}$ in which the second and third derivatives are nonzero. We will demonstrate the remainder blowup of a first-order TM for a single-hidden-layer neuron FF network [see Fig. 4(a)]. As expected, the remainder indeed grows with increasing diameter of $\mathbf{X}$. However, the effect of increasing $|\mathbf{X}|$ is far lower for the FFNNs than for the RBFNNs, because only the terms ϕ_j contain $\mathbf{x}$ and the output of this function is bounded between -1 and 1. Therefore, higher orders of ϕ_j are also bounded between -1 and 1. This means that the cause of the blowup of the remainder is determined by the width of $(\mathbf{X}-\mathbf{x}_0)$ and if $\mathbf{X}$ is sufficiently small then an increase in order n would provide a TM with smaller remainder [see Fig. 4(b)]. A conclusion is that TMs are better suited for FFNNs than for RBFNNs. In Section III-D, we will discuss our implementation of the branch-and-bound algorithm using TMs. A combination of standard IA and TMs is used to prevent remainder blowup and make the algorithm more efficient.

[4]When using NNs, one usually wants approximation capacity for the entire range $\mathbf{X}$ for which we require that at least one neuron has a first-order derivative unequal to zero. Therefore, we can conclude that we always have overestimation for $\mathbf{X}$.

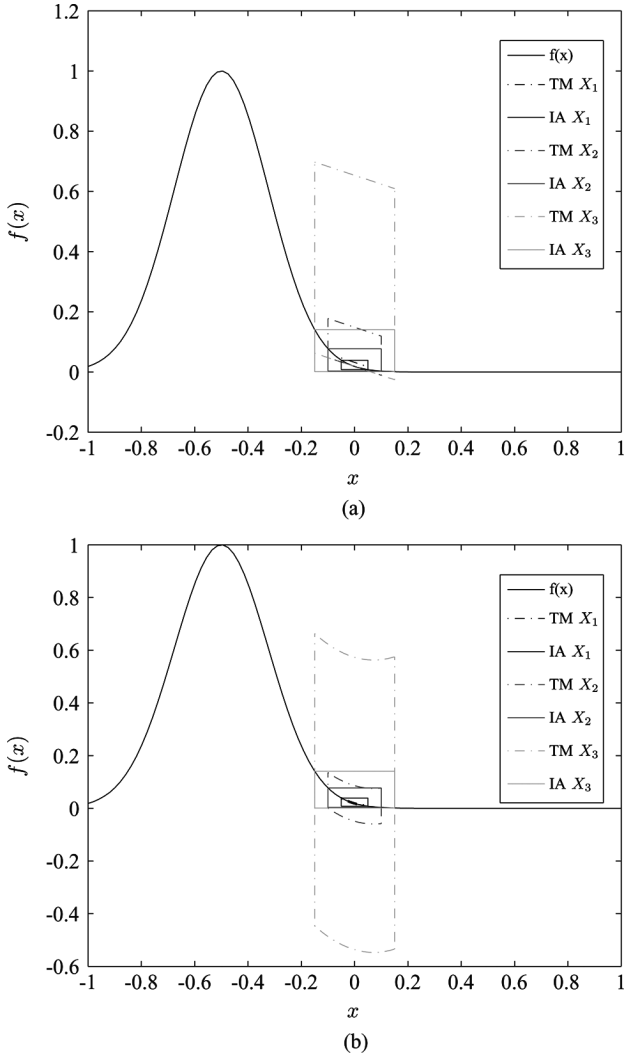

Fig. 3. (a) First- and (b) second-order TM inclusion of a 1-1-1 RBFNN.

D. TM Optimization Algorithm

The optimization algorithm based on TMs is an adapted version of the branch-and-bound algorithm and it is given in Table IV. In step 2, we perform function evaluation over domain $\mathbf{X}$ using a TM $(\mathrm{NN}^{\mathrm{TM}})$ and standard interval analysis $(\mathrm{NN}^{\mathrm{IA}})$. During the computation of the TM, we also computed

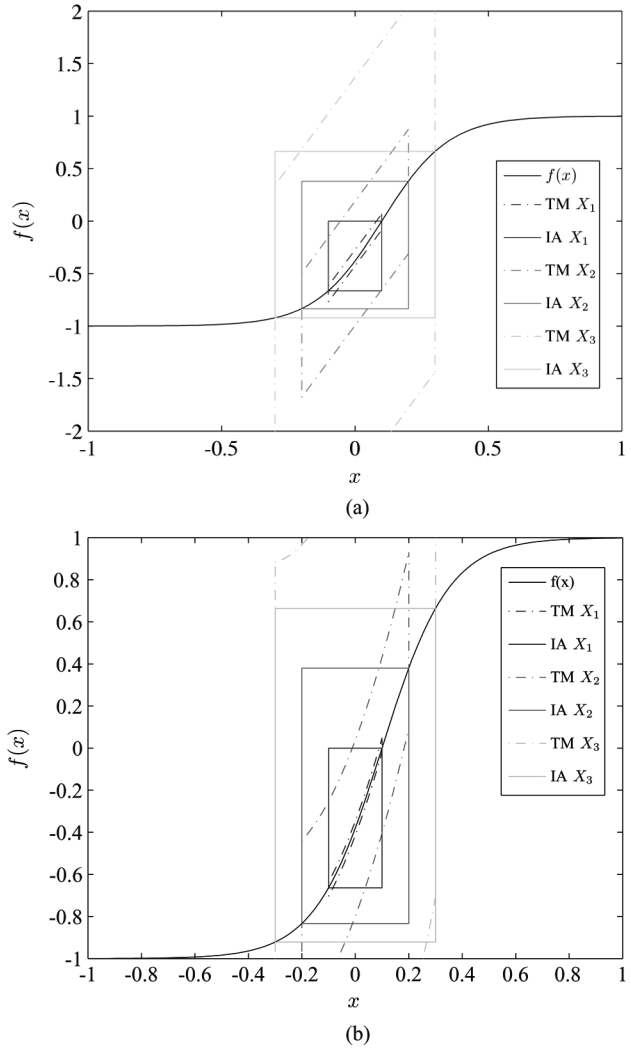

Fig. 4. (a) First- and (b) second-order TM inclusion of a 1-1-1 FFNN.

the bounds provided by standard IA. To prevent the expansion of the function evaluation due to TM remainder blowup (see Sections III-B and III-C), we take the lowest upper bound and the highest lower bound

$$\begin{aligned}\sup \mathrm{NN} &= \min[\sup \mathrm{NN}^{\mathrm{IA}}, \max \sup \mathrm{NN}^{\mathrm{TM}}] \\ &= \min\left[\sup \mathrm{NN}^{\mathrm{IA}}, \max T^n_{\mathrm{NN},\mathbf{x}} + \sup I_n\right] \\ \inf \mathrm{NN} &= \max[\inf \mathrm{NN}^{\mathrm{IA}}, \max \inf \mathrm{NN}^{\mathrm{TM}}] \\ &= \max\left[\inf \mathrm{NN}^{\mathrm{IA}}, \max T^n_{\mathrm{NN},\mathbf{x}} + \inf I_n\right]. \quad (30)\end{aligned}$$

The computation of $\max T^n_{\mathrm{NN},\mathbf{x}}$ and $\min T^n_{\mathrm{NN},\mathbf{x}}$ depends on the order n. To make these computations easy, we restrict ourselves to orders up to 2.

An important step in the optimization algorithm is the contraction step. If the upper bound of $\mathrm{NN}^2(x_i, \hat{\mathbf{X}}, \mathbf{W})$, i.e., the upper bound of the TM with dimension i reduced to a crisp value, is lower than ρ, the current estimate of the maximum value, then contraction of X_i *might* be possible. The problem with the implementation of contractions is that the domain $\hat{X}_i$, where $\mathrm{NN}^2(x_i, \hat{\mathbf{X}}, \mathbf{W}) < \rho$ is usually not aligned with the coordinate axis of $\mathbf{x}$, which could mean that contractions are not possible. If, however, we were able to contract $\mathbf{X}$ for at least one dimension, we must first look if the stopping condition has been reached. If not, we must check if the diameter of the domain is reduced by a factor α or more (α is usually defined in the range of $[0, 0.5]$). If this check is passed, then we decide not to split the box and place it back into the list. This approach reduces the number of bisections and increases the efficiency of the algorithm in terms of total number of required box evaluations. The choice of α is problem dependent and the optimal value can be found through investigation. The rest of the algorithm is the same as the enhanced branch-and-bound algorithm. For orders up to 2, the contraction step is easily performed using simple algebraic relations.

TABLE III
SOLUTION BOXES FOR SIMPLE EXAMPLE BRANCH-AND-BOUND ALGORITHM

$\mathbf{X}$	$NN(\mathbf{X}, \mathbf{W})$
$[0.875, 0.938][0.375, 0.438]$	$[0.855, 0.962]$
$[0.875, 0.938][0.438, 0.500]$	$[0.918, 0.984]$
$[0.938, 1.000][0.375, 0.438]$	$[0.896, 0.977]$
$[0.938, 1.000][0.438, 0.500]$	$[0.962, 1.000]$
$[0.875, 0.938][0.500, 0.563]$	$[0.918, 0.984]$
$[0.875, 0.938][0.563, 0.625]$	$[0.855, 0.962]$
$[0.938, 1.000][0.500, 0.563]$	$[0.962, 1.000]$
$[0.938, 1.000][0.563, 0.625]$	$[0.896, 0.977]$
$[1.000, 1.063][0.375, 0.438]$	$[0.896, 0.977]$
$[1.000, 1.063][0.438, 0.500]$	$[0.962, 1.000]$
$[1.063, 1.125][0.375, 0.438]$	$[0.855, 0.962]$
$[1.063, 1.125][0.438, 0.500]$	$[0.918, 0.984]$
$[1.000, 1.063][0.500, 0.563]$	$[0.962, 1.000]$
$[1.000, 1.063][0.563, 0.625]$	$[0.896, 0.977]$
$[1.063, 1.125][0.500, 0.563]$	$[0.918, 0.984]$
$[1.063, 1.125][0.563, 0.625]$	$[0.855, 0.962]$
$[-1.625, -1.563][-1.063, -1.000]$	$[0.865, 0.973]$
$[-1.563, -1.500][-1.125, -1.063]$	$[0.845, 0.965]$
$[-1.563, -1.500][-1.063, -1.000]$	$[0.939, 1.000]$
$[-1.500, -1.438][-1.125, -1.063]$	$[0.845, 0.965]$
$[-1.500, -1.438][-1.063, -1.000]$	$[0.939, 1.000]$
$[-1.438, -1.375][-1.063, -1.000]$	$[0.865, 0.973]$
$[-1.625, -1.563][-1.000, -0.938]$	$[0.865, 0.973]$
$[-1.563, -1.500][-1.000, -0.938]$	$[0.939, 1.000]$
$[-1.563, -1.500][-0.938, -0.875]$	$[0.845, 0.965]$
$[-1.500, -1.438][-1.000, -0.938]$	$[0.939, 1.000]$
$[-1.500, -1.438][-0.938, -0.875]$	$[0.845, 0.965]$
$[-1.438, -1.375][-1.000, -0.938]$	$[0.865, 0.973]$

TABLE IV
TM OPTIMIZATION ALGORITHM

0.	Initialization: put initial box $\mathbf{X}_0$ into list L, set $\rho = -\inf$
DO WHILE boxes are left in list L	
1.	Take first box $\mathbf{X}$ out of list L and remove it form the list
2.	Perform function evaluation with IA (NN^{IA}) and Taylor Model (NN^{TM}): $\sup NN(\mathbf{X}, \mathbf{W}) = \min[\sup NN^{IA}, \sup NN^{TM}]$ $\inf NN(\mathbf{X}, \mathbf{W}) = \max[\inf NN^{IA}, \inf NN^{TM}]$
3.	Update ρ: $\rho = \max[\rho, \inf NN(\mathbf{X}, \mathbf{W})]$
4.	Perform mid-point evaluation and update ρ: $\rho = \max[\rho, NN(\mathrm{mid}(\mathbf{X}), \mathbf{W})]$
5.	Compute derivatives $\partial NN/\partial x_i$ and apply rules from equations (16)-(18). If box can be deleted goto 1.
6.	Contract box based using polynomial inclusion function.
7.	If $\lvert\mathbf{X}_{new}\rvert < \sigma_x$ or $\lvert NN(\mathbf{X}, \mathbf{W})\rvert < \sigma_y$; store box as possible solution and goto 1; end
8.	If $\lvert\mathbf{X}_{new}\rvert < \alpha\lvert\mathbf{X}\rvert$; Add box to list L and goto 1; end
9.	Split box and add all sub-boxes to list L
END	

IV. Polynomial Set Method

The problem of TMs lies in the determination of the remainder, which leads to an inclusion F that can be larger than the inclusion bounds obtained through standard IA. In this section, we will derive the PS method. There are two key aspects of the new method.

1) The inclusion function is defined as a set of two crisp polynomials $(P^n_{\underline{\mathrm{NN}},\mathbf{x}}, P^n_{\overline{\mathrm{NN}},\mathbf{x}})$, where

$$\begin{aligned} P^n_{\underline{\mathrm{NN}},\mathbf{x}}(\mathbf{x}) &\leq \mathrm{NN}(\mathbf{x},\mathbf{W}) \qquad \forall \mathbf{x}\in\mathbf{X} \\ P^n_{\overline{\mathrm{NN}},\mathbf{x}}(\mathbf{x}) &\geq \mathrm{NN}(\mathbf{x},\mathbf{W}) \qquad \forall \mathbf{x}\in\mathbf{X}. \end{aligned} \tag{31}$$

2) The inclusion function $(P^n_{\underline{\mathrm{NN}},\mathbf{x}}, P^n_{\overline{\mathrm{NN}},\mathbf{x}})$ always lies within the bounds provided by standard IA, i.e.,

$$\inf F \leq P^n_{\underline{\mathrm{NN}},\mathbf{x}}(\mathbf{x}) \leq P^n_{\overline{\mathrm{NN}},\mathbf{x}}(\mathbf{x}) \leq \sup \mathrm{NN} \qquad \forall \mathbf{x}\in\mathbf{X}. \tag{32}$$

The first aspect makes this method different from TMs, where one crisp polynomial is used plus a bounded remainder, and the method of Hansen, where one polynomial with interval parameters is used [38]. The second aspect ensures that the information obtained through a function evaluation is equal to or richer than the information obtained through standard interval analysis. The upper and lower bounds are equal to or tighter than those obtained with standard interval analysis and the shapes of the polynomials give more information about the shape of the NN output. Better estimates of the maximum output value are possible. The method of deriving the two polynomials $P^n_{\underline{\mathrm{NN}},\mathbf{x}}(\mathbf{x}), P^n_{\overline{\mathrm{NN}},\mathbf{x}}(\mathbf{x})$ is problem dependent, but always possible. In the worst case scenario, one can only derive a zeroth-order polynomial, which reduces the method to standard IA. We will first discuss the application to RBFNNs and thereafter the application to FFNNs.

A. RBFNN

The approach to deriving the polynomial model is based on summation of the weighted polynomial models of each hidden-layer neuron

$$P^n_{\mathrm{NN},\mathbf{x}} = \sum_j w_{jk} P^n_{\phi_j,\mathbf{x}}. \tag{33}$$

We will therefore focus on a single RBF for domain $\mathbf{X}$

$$\phi_j(\mathbf{x}) = \exp\left(-\sum_i w^2_{ij}(x_i - c_{ij})^2\right) = \exp\left(-v_j^2\right). \tag{34}$$

For the RBFNNs, the input space $\mathbf{X}$ is mapped into the hidden-layer neuron input space $V_j^2 \subset [0,\infty]$ through the relation

$$v_j^2 = \sum_i w^2_{ij}(x_i - c_{ij})^2. \tag{35}$$

If we determine a first-order polynomial in terms of v_j^2, by substitution of the previous relation, we can obtain a second-order polynomial in $\mathbf{x}$. Higher order polynomials in v_j^2 of order n therefore lead to $2n$-order polynomials in $\mathbf{x}$. The benefit of deriving the polynomial in v_j^2 first is that we simplify the derivation. The drawback on the other hand is that information is lost during the mapping $\mathbf{x} \rightarrow \mathbf{v_j^2} \rightarrow \mathbf{x}$, which can lead to overestimation of the inclusion function bound, i.e., the integrated error $P^n_{\phi_j,\mathbf{x}} - f(\mathbf{x})$ could be made smaller.

Theorem IV.1: Assume the polynomials

$$\begin{aligned} &\left(P^1_{\overline{\phi}_j,v_j^2} = \bar{a}_0 + \bar{a}_1 v_j^2\right) \\ &\left(P^1_{\underline{\phi}_j,v_j^2} = \underline{a}_0 + \underline{a}_1 v_j^2\right) \end{aligned} \tag{36}$$

on the domain $v_j^2 \in V_j^2 \subset R$, where the parameters are defined as

$$\begin{aligned} \bar{a}_1 &= \frac{\phi_j\left(\inf V_j^2\right) - \phi_j\left(\sup V_j^2\right)}{\inf V_j^2 - \sup V_j^2} \\ \bar{a}_0 &= \phi_j\left(\inf V_j^2\right) - \bar{a}_1 \inf V_j^2 \\ \underline{a}_1 &= \left.\frac{\partial \phi_j}{\partial x}\right|_{\left(\sup V_j^2\right)} \\ \underline{a}_0 &= \phi_j\left(\sup V_j^2\right) - \underline{a}_1 \sup V_j^2 \end{aligned} \tag{37}$$

with $\phi_j(v_j^2) = \exp(-v_j^2)$, then

$$\begin{aligned} \phi_j\left(v_j^2\right) &\leq P^1_{\overline{\phi}_j,v_j^2} \leq \sup \Phi_j\left(V_j^2\right) \\ \inf \Phi_j\left(V_j^2\right) &\leq P^1_{\underline{\phi}_j,v_j^2} \leq \phi_j\left(v_j^2\right) \end{aligned} \tag{38}$$

on the domain $v_j^2 \in V_j^2$ and the integrated errors

$$\int_{\inf V_j^2}^{\sup V_j^2} \left(P^1_{\overline{\phi}_j,v_j^2} - \phi_j\right) dv_j^2$$

$$\int_{\inf V_j^2}^{\sup V_j^2} \left(\phi_j - P^1_{\underline{\phi}_j,v_j^2}\right) dv_j^2$$

are minimal.

Proof: See the Appendix. ∎

Once the polynomials have been determined in v_j^2, we can determine the polynomials in $\mathbf{x}$ by substituting equation (35). The final polynomials are of the form

$$\begin{aligned} P^2_{\overline{\phi}_j,\mathbf{x}} &= \bar{b}_0 + \sum_i \bar{b}_{1i} x_i + \bar{b}_{2i} x_i^2 \\ P^2_{\underline{\phi}_j,\mathbf{x}} &= \underline{b}_0 + \sum_i \underline{b}_{1i} x_i + \underline{b}_{2i} x_i^2 \end{aligned} \tag{39}$$

with parameters

$$\begin{aligned} \bar{b}_0 &= \bar{a}_0 + \bar{a}_1 \sum_i w_{ij}^2 c_{ij}^2 \\ \bar{b}_{1i} &= -2\bar{a}_1 w_{ij}^2 c_{ij} \\ \bar{b}_{2i} &= \bar{a}_1 w_{ij}^2 \end{aligned} \tag{40}$$

$$\begin{aligned} \underline{b}_0 &= \underline{a}_0 + \underline{a}_1 \sum_i w_{ij}^2 c_{ij}^2 \\ \underline{b}_{1i} &= -2\underline{a}_1 w_{ij}^2 c_{ij} \\ \underline{b}_{2i} &= \underline{a}_1 w_{ij}^2. \end{aligned} \tag{41}$$

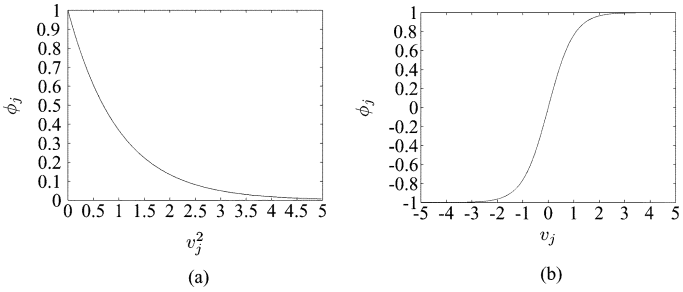

Fig. 5. Hidden-layer neuron activation functions. (a) Exponential function. (b) Tangent sigmoidal function.

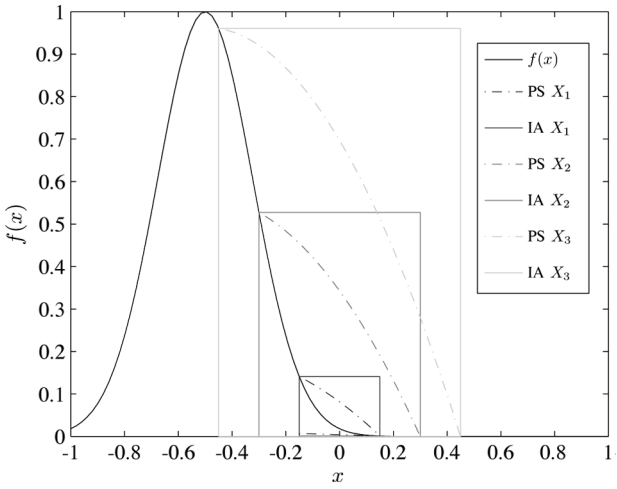

Fig. 6. PS inclusion of a 1-1-1 RBFNN.

The inclusion function for the complete NN is given by

$$P^2_{\overline{\mathrm{NN}},\mathbf{x}} = \sum_j w_{jk}\left[\alpha P^2_{\bar{\phi}_j,\mathbf{x}} + \beta P^2_{\underline{\phi}_j,\mathbf{x}}\right]$$
$$P^2_{\underline{\mathrm{NN}},\mathbf{x}} = \sum_j w_{jk}\left[\beta P^2_{\bar{\phi}_j,\mathbf{x}} + \alpha P^2_{\underline{\phi}_j,\mathbf{x}}\right]$$
$$\begin{cases} \alpha = 1 & \beta = 0, \quad w_{jk} \geq 0 \\ \alpha = 0 & \beta = 1, \quad w_{jk} < 0. \end{cases} \tag{42}$$

Since the output is simply the summation of hidden-layer neuron outputs, the guarantee of tighter bounds on the hidden-layer neuron outputs implies that we have equal or tighter bounds on the output of the complete network as compared to standard IA (see Fig. 6).

An advantage is that the obtained crisp polynomial for the upper bound and lower bound in $\mathbf{x}$ can be easily analyzed. The value of the maximum minimum value can be computed analytically and used to update the estimate of the maximum output value, i.e.,

$$\rho = \max\left(\rho, \max_{\mathbf{x}\in\mathbf{X}} P^2_{\underline{\mathrm{NN}},\mathbf{x}}\right). \tag{43}$$

Then, we can compute the region where the inclusion function upper bound is lower than the current estimate ρ such that we can possibly contract the domain $\mathbf{X}$. This process is the same as for the TMs (see Section III-D).

B. FFNN

The approach taken here is similar to the one for RBFNNs. Since the output is the weighted summation of the hidden-layer neuron output [see (33)], we will look at a single-hidden-layer neuron

$$\phi_j = \frac{2}{1+\exp\left(-2\left[\sum_i \{w_{ij}x_i\} + b_j\right]\right)} - 1$$
$$= \frac{2}{1+\exp(-2v_j)} - 1. \tag{44}$$

The input space $\mathbf{x}$ is mapped into the hidden-layer neuron input space $v_j \in [-\infty, \infty]$ through the relation

$$v_j = \sum_i \{w_{ij}x_i\} + b_j. \tag{45}$$

If we determine nth-order polynomial in terms of v_j, by substituting the previous relation, we can obtain an nth-order polynomial in $\mathbf{x}$. For $n < 2$, we can easily derive bounds on the maximum and minimum values of $P^2_{\underline{\mathrm{NN}},\mathbf{x}}, P^2_{\overline{\mathrm{NN}},\mathbf{x}}$ and contractions can be performed using simple algebraic rules.

Theorem IV.2: Assume the polynomials

$$(P^1_{\bar{\phi}_j,v_j} = \bar{a}_0 + \bar{a}_1 v_j)$$
$$(P^1_{\underline{\phi}_j,v_j} = \underline{a}_0 + \underline{a}_1 v_j) \tag{46}$$

on the domain $v_j \in V_j \subset R$, where the parameters are defined as

$$\bar{a}_1 = \min\left(\left.\frac{\partial\phi_j}{\partial v_j}\right|_{\sup V_j}, \frac{\phi_j(\inf V_j) - \phi_j(\sup V_j)}{(\inf V_j - \sup V_j)}\right)$$
$$\bar{a}_0 = \phi_j(\sup V_j) - \bar{a}_1 \sup V_j$$
$$\underline{a}_1 = \min\left(\left.\frac{\partial\phi_j}{\partial v_j}\right|_{\inf V_j}, \frac{\phi_j(\inf V_j) - \phi_j(\sup V_j)}{(\inf V_j - \sup V_j)}\right)$$
$$\underline{a}_0 = \phi_j(\inf V_j) - \underline{a}_1 \inf V_j \tag{47}$$

with $\phi_j(v_j) = 2/(1+\exp(-2v_j)) - 1)$, then

$$\phi_j(v_j) \leq P^1_{\bar{\phi}_j,v_j} \leq \sup \Phi_j(V_j)$$
$$\inf \Phi_j(V_j) \leq P^1_{\underline{\phi}_j,v_j} \leq \phi_j(v_j) \tag{48}$$

on the domain $v_j \in V_j$ and the integrated errors

$$\int_{\inf V_j}^{\sup V_j} \left(P^1_{\bar{\phi}_j,v_j} - \phi_j\right) dv_j$$
$$\int_{\inf V_j}^{\sup V_j} \left(\phi_j - P^1_{\underline{\phi}_j,v_j}\right) dv_j$$

are minimal.

Proof: See the Appendix. ■

Once the polynomials have been determined in v_j, we can determine the polynomials in $\mathbf{x}$ by substituting equation (45). The final polynomials are of the form

$$P^1_{\bar{\phi}_j,\mathbf{x}} = \bar{b}_0 + \sum_i \bar{b}_{1i}x_i$$
$$P^1_{\underline{\phi}_j,\mathbf{x}} = \underline{b}_0 + \sum_i \underline{b}_{1i}x_i \tag{49}$$

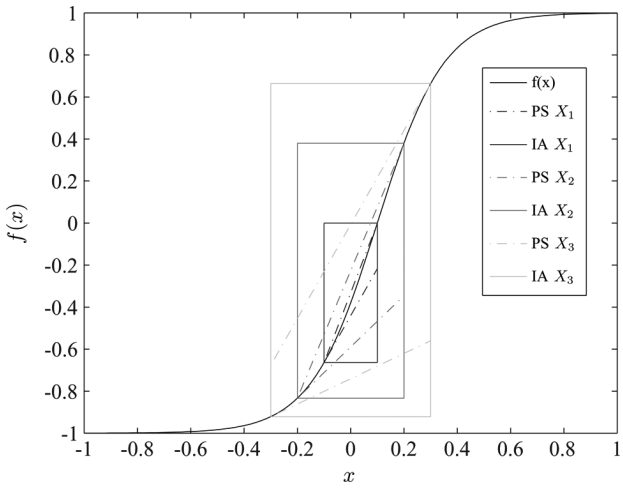

Fig. 7. PS inclusion of a 1-1-1 FFNN.

with parameters

$$\bar{b}_0 = \bar{a}_0 + \bar{a}_1 b_j$$
$$\bar{b}_{1i} = \bar{a}_1 w_{ij} \quad (50)$$
$$\underline{b}_0 = \underline{a}_0 + \underline{a}_1 b_j$$
$$\underline{b}_{1i} = \underline{a}_1 w_{ij}. \quad (51)$$

The inclusion function for the complete NN is given by (42). Since the output is simply the summation of hidden-layer neuron outputs, the guarantee of tighter bounds on the hidden-layer neuron outputs implies that we have equal or tighter bounds on the output of the complete network as compared to standard IA (see Fig. 7).

As for the RBFNNs, we can use the obtained crisp polynomial for the upper bound to derive the value of the maximum–minimum used to update the estimate of the maximum output value. Moreover, we can apply contractions based on the current estimate of the maximum output value and the inclusion polynomials.

C. Polynomial Optimization Algorithm

The optimization algorithm using polynomial inclusion functions is equal to that of the TM optimization algorithm (see Table IV) except for step 2. Step 2 becomes as follows.

Perform function evaluation with polynomial inclusion function

$$\sup F = P^n_{\overline{\mathrm{NN}},\mathbf{x}}$$
$$\inf F = P^n_{\underline{\mathrm{NN}},\mathbf{x}}.$$

Evaluation using standard IAs is not required since we have guaranteed tighter bounds when using polynomial inclusion functions. In Section V, we will present the results for all the methods discussed in this paper.

V. Simulation Results

The methods discussed in this paper differ in the setup of the function evaluation. To test the efficiency of different methods, a simulation is performed. The setup of the experiments for the RBF networks is as follows.

- A vector containing the number of hidden-layer neurons is selected: $N = [2, 4, 8, 16, 32, 64]$. σ_x is set to 1% of the input range for each input dimension and α is set to 0.25.
- For each entry in this vector, a batch of 250 randomly initialized RBFNNs is created for the case of a 1-D, 2-D, and 3-D input.
- For each RBFNNs in the batch, we perform the optimization task with the following optimization algorithms:
  1) enhanced branch-and-bound (BB) algorithm, without contractions (step 6) (see Section II);
  2) enhanced BB algorithm, with contractions (WC) based on Newton step (see Section II);
  3) TM optimization algorithm, second-order polynomial (see Section III);
  4) TM optimization algorithm with contractions (WC), second-order polynomial (see Section III);
  5) PS optimization algorithm, second-order polynomial (see Section IV);
  6) PS optimization algorithm with contractions (WC), second-order polynomial (see Section IV).
- For each algorithm, the computation time will be recorded.
- The experiments are performed on a single 2.4-GHz Intel(R) Core(TM) 2Quad core. During the evaluation of the results, we will look only at relative results since the absolute value of the computational load is central processing unit (CPU) dependent.
- The experiments are performed using MATLAB© and interval toolbox INTLAB© developed by S. M. Rump at Hamburg University of Technology, Hamburg, Germany.

We have chosen to use the second-order TM for the RBFNNs to make a fair comparison with the PS method. Due to the setup of the polynomial derivations for the PS inclusion function, we obtain a second-order polynomial for this method (see Section IV-A). The experiments for the FFNN are the same as for RBFNN although now we choose $N = [2, 4, 8, 16, 32, 64, 128]$, and first-order polynomials are selected for both the TM and the PS methods.

The results of the RBF experiments are shown in Fig. 8. Based on the results, we can note the following.

- The computational load increases approximately linearly with the number of hidden-layer neurons for each method. More neurons means more computation per function evaluation and also more dependency effect due to more occurrences of $\mathbf{X}$ in the network output function. This result does depend on the required accuracy σ_x.
- The PS method with contractions is most efficient for all cases. When considering >7 number of neurons, the PS method is followed by the TM method with contractions and thereafter the BB algorithm. This is purely due to the level of information content obtained from a function evaluation $\mathrm{NN}(\mathbf{X}, \mathbf{W})$. BB uses standard IA, which yields the minimum required information content. TMs provide more information of the function shape *if* the remainder bound is small enough (see Section III-B). The remainder bound is the main problem for the TM method. When the interval V_j^2 is small enough for each hidden-layer neuron then the

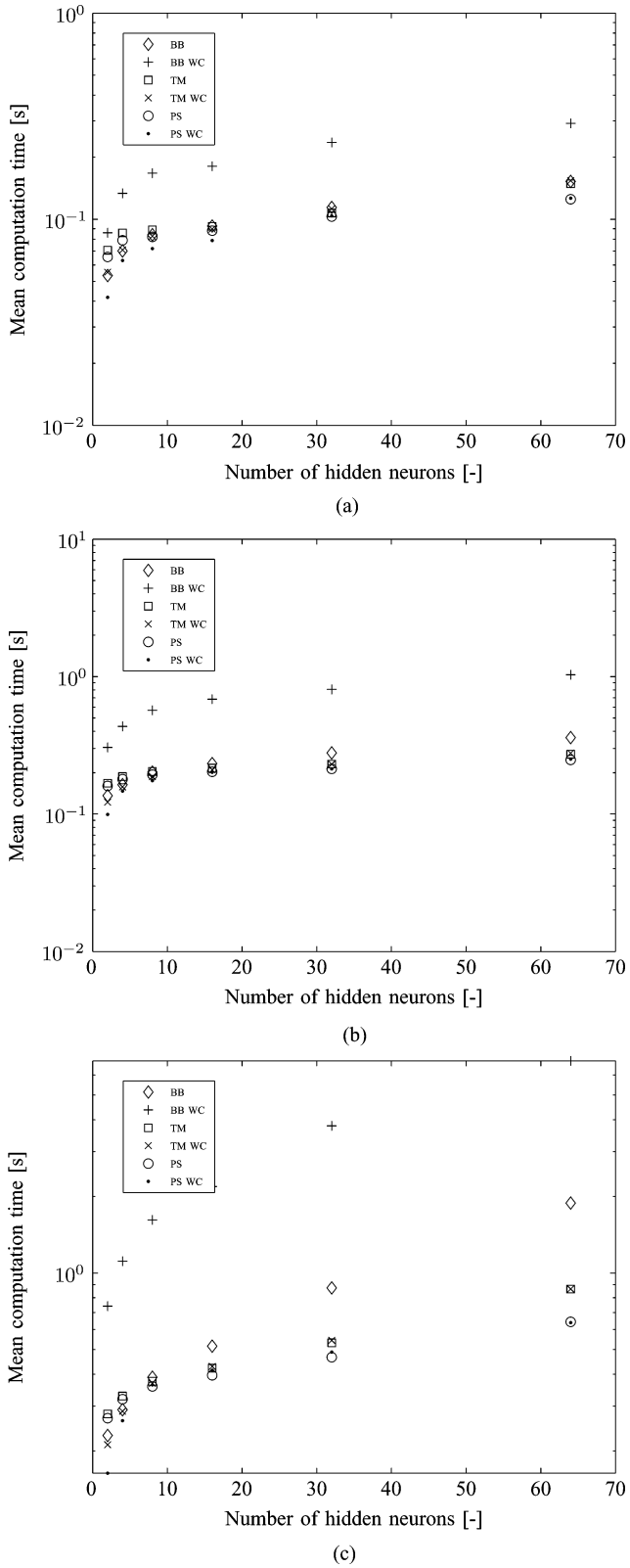

Fig. 8. Optimization algorithm comparison results for RBFNNs. (a) 1-N-1 RBFNNs. (b) 2-N-1 RBFNNs. (c) 3-N-1 RBFNNs.

method provides more information than the standard interval analysis method. This means that during the optimization the initial phase will mostly consist of function evaluations based on standard interval analysis whereas the final phase (when V_j^2 has become small enough) will consist of function evaluations based on TMs. This is the difference with the PS method for which the function evaluations are always performed using the PS and not with standard interval analysis. This means that more information can be used from the very beginning of the optimization therefore making the algorithm more efficient.

- For lower amount of hidden-layer neurons, the BB algorithm performs better than the PS and TM method without contractions. The reason is that the determination of the inclusion function for PS and TM methods demands more computational load, while the additional information about the network shape and output bounds does not improve the efficiency (in terms of computational load) much, since little dependency is present. The methods with contractions are performing equally well or better than the BB algorithm since they make better use of the information regarding the network shape (via contractions).
- The application of contractions decreases efficiency in terms of computational load for the BB algorithm. The additional computations required for the contractions and the larger dependency effect in the derivatives cause little reductions of the boxes. The increased computational load outweighs the increased algorithm efficiency, which causes the overall computation time to increase.
- The application of contractions enhances efficiency of the PS and TM algorithms for the lower number of hidden-layer neurons. This result is depends on the required accuracy σ_x, σ_y. In the experiments, we kept this parameter constant, which implies that for a higher amount of hidden-layer neurons, the minimum V_j^2 interval width increases. The information content using the TM method and the PS method increases with decreasing V_j^2 intervals and contractions are only possible when enough information is present. Therefore, the percentage of actual contractions decreases with increasing number of hidden-layer neurons. If we decreased σ_x or σ_y with increasing number of hidden-layer neurons such that the same widths of V_j^2 were attained, then we predict that the contractions would speed up the optimization process for higher number of hidden-layer neurons.

The results for the FFNN experiments are shown in Fig. 9. The conclusions drawn from these results are similar to the case of RBFNNs.

- The computational load increases approximately linearly with the number of hidden-layer neurons for each method.
- The PS method with contractions is most efficient for the 2-D and 3-D cases. When considering >15 number of neurons, the PS method is followed by the TM method and thereafter the BB algorithm. For the 1-D case, the BB algorithm without contractions outperforms the other methods for low number of neurons. This is due to the increased computational load for computing the inclusion functions for the PS and TM methods. It appears that the increase in algorithm efficiency does not outweigh the increase in computational load.
- For lower amount of hidden-layer neurons, the BB algorithm performs better than the PS and TM methods without

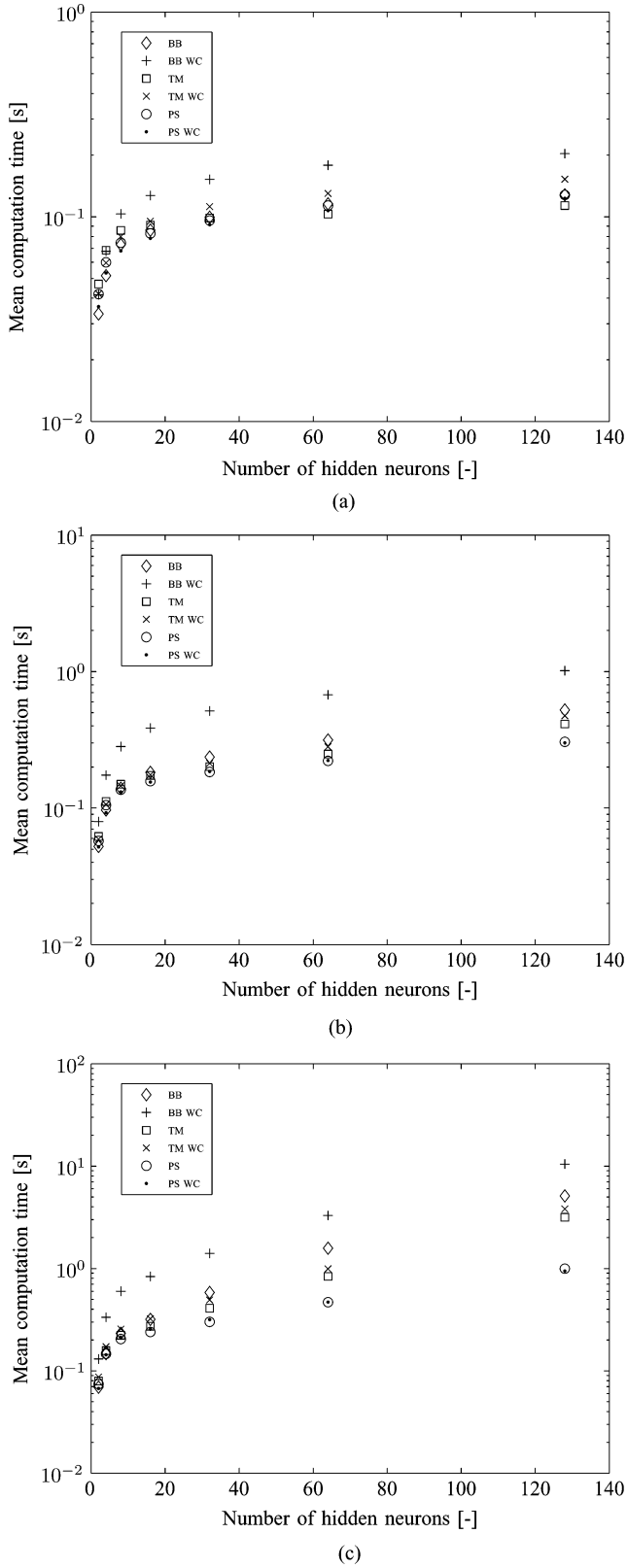

Fig. 9. Optimization algorithm comparison results for FFNNs. (a) 1-N-1 FFNNs. (b) 2-N-1 FFNNs. (c) 3-N-1 FFNNs.

contractions. The reason for this is the more efficient algorithm in terms of computational load, *not* in terms of evaluation efficiency as explained before.

- The application of contractions enhances efficiency of the optimization algorithms for the lower number of hidden-layer neurons due to the same reason as for the RBFNNs.

Looking at the results for both RBF and FF networks, we can state that the computational load increases considerably when increasing the input dimension. Finally, we note that the superior performance of the PS method becomes more pronounced with respect to other methods for higher dimensional input spaces.

VI. Conclusion

NN output optimization via interval analysis guarantees that *all* solutions are obtained. Standard interval analysis is slow due to dependency and minimal information content of a function evaluation. Introducing TMs makes the algorithm more efficient for larger amounts of neurons and higher dimensional input spaces, although it still suffers from dependency, which can lead to severe remainder blowup. Therefore, using TMs is only beneficial for smaller input space intervals. The PS method guarantees bounds tighter than those of standard interval analysis irrespectively of input space interval widths. Therefore, more useful information is introduced from the beginning of the optimization. This information can be used to increase efficiency considerably, and therefore, the PS method has proved to be the fastest algorithm for both the FFNNs and the RBFNNs.

Appendix
Proof of Theorems IV.1 and IV.2

In this Appendix, we will present the proof of Theorems IV.1 and IV.2.

Proof of Theorem IV.1: The derivative of $\phi_j(v_j^2)$ is

$$\frac{\partial \phi_j}{\partial v_j^2} = -\exp\left(-v_j^2\right) = -\phi_j \tag{52}$$

which is negative on the domain V_j^2. This means that $\phi_j(v_j^2)$ is monotonically decreasing on domain V_j^2 and that the maximal value of ϕ_j is attained at $v_j^2 = \inf V_j^2$ and the minimal value at $v_j^2 = \sup V_j^2$. By Theorem II.1, we have that

$$\begin{aligned} \phi_j\left(\inf V_j^2\right) &= \sup \Phi_j\left(V_j^2\right) \\ \phi_j\left(\sup V_j^2\right) &= \inf \Phi_j\left(V_j^2\right). \end{aligned} \tag{53}$$

Since we must have $\sup \Phi_j(V_j^2) \geq P^1_{\phi_j, v_j^2} \geq \phi(v_j^2)\, \forall v_j^2 \in V_j^2$, we know that

$$\begin{aligned} \bar{a}_1 \inf V_j^2 + \bar{a}_0 &= \phi_j\left(\inf V_j^2\right) \\ \underline{a}_1 \sup V_j^2 + \underline{a}_0 &= \phi_j\left(\sup V_j^2\right) \\ &\Rightarrow \begin{array}{l} \bar{a}_0 = \phi_j\left(\inf V_j^2\right) - \bar{a}_1 \inf V_j^2 \\ \underline{a}_0 = \phi_j\left(\sup V_j^2\right) - \underline{a}_1 \sup V_j^2 \end{array}. \end{aligned} \tag{54}$$

Now we only need to specify the value of $\bar{a}_1$ and $\underline{a}_1$. Let us first look at the value of $\bar{a}_1$. Since we require $P^1_{\phi_j, v_j^2} > \phi_j(v_j^2) \forall v_j^2 \in$

V_j^2, we can find the optimal value of $\bar{a}_1$ by minimizing the integrated error between the polynomial and the activation function

$$\begin{aligned}
&\min_{\bar{a}_1} \int_{\inf V_j^2}^{\sup V_j^2} \left(P^1_{\phi_j,v_j^2} - \phi_j\right) dv_j^2 \\
&\quad \rightarrow \min_{\bar{a}_1} \int_{\inf V_j^2}^{\sup V_j^2} P^1_{\phi_j,v_j^2} dv_j^2 \\
&\quad = \min_{\bar{a}_1} \int_{\inf V_j^2}^{\sup V_j^2} \left(\bar{a}_1 v_j^2 - \bar{a}_1 \inf V_j^2\right) dv_j^2 \\
&\quad = \min_{\bar{a}_1} \left[\frac{\bar{a}_1}{2}\left(\sup V_j^2 - \inf V_j^2\right)^2\right]. \qquad (55)
\end{aligned}$$

The term $(\sup V_j^2 - \inf V_j^2)$ is always positive thus we know that the minimization of the integral can be seen as the minimization of $\bar{a}_1$. The lowest possible value of $\bar{a}_1$ is determined by the constraint

$$P^1_{\phi_j,v_j^2} > \phi_j\left(v_j^2\right) \qquad \forall v_j^2 \in V_j^2. \qquad (56)$$

We can rewrite this in the form

$$\begin{aligned}
&\bar{a}_1 v_j^2 - \bar{a}_1 \inf V_j^2 + \phi_j\left(\inf V_j^2\right) > \phi_j\left(v_j^2\right) \\
&\bar{a}_1\left(v_j^2 - \inf V_j^2\right) > \phi_j\left(v_j^2\right) - \phi_j\left(\inf V_j^2\right) \\
&\bar{a}_1 > \frac{\phi_j\left(v_j^2\right) - \phi_j\left(\inf V_j^2\right)}{\left(v_j^2 - \inf V_j^2\right)}. \qquad (57)
\end{aligned}$$

We can take the derivative of the latter function with respect to v_j^2 to find the value of $\bar{a}_1$ such that the constraint holds $\forall v_j^2 \in V_j^2$

$$\frac{\partial \bar{a}_1}{\partial v_j^2} = \frac{-\phi_j\left(v_j^2\right)\left(v_j^2 - \inf V_j^2\right) - \phi_j\left(v_j^2\right) + \phi_j\left(\inf V_j^2\right)}{\left(v_j^2 - \inf V_j^2\right)^2}. \qquad (58)$$

Since we have that

$$\begin{aligned}
&-\phi_j\left(v_j^2\right)\left(v_j^2 - \inf V_j^2\right) < 0 \\
&-\phi_j\left(v_j^2\right) + \phi_j\left(\inf V_j^2\right) > 0 \qquad (59)
\end{aligned}$$

we know that the right-hand side of (58) is always positive if

$$\phi_j\left(v_j^2\right)\left(v_j^2 - \inf V_j^2\right) < \phi_j\left(\inf V_j^2\right) - \phi_j\left(v_j^2\right). \qquad (60)$$

We can rewrite this relation to

$$\left(v_j^2 - \inf V_j^2\right) < \frac{\phi_j\left(\inf V_j^2\right)}{\phi_j\left(v_j^2\right)} - 1 = \phi_j\left(\inf V_j^2 - v_j^2\right) - 1 \qquad (61)$$

which holds $\forall v_j^2 > \inf V_j^2$ since

$$v_j^2 = \inf V_j^2 \rightarrow 0 \leq 1 - 1 = 0 \qquad (62)$$

and the derivative with respect to v_j^2 of the right-hand side of (58) is always larger than that of the left-hand side

$$1 < \phi_j\left(\inf V_j^2 - v_j^2\right). \qquad (63)$$

Since we know that the function $\partial \bar{a}_1 / \partial v_j^2$ is always positive, we can determine the maximal value of the right-hand side of (58)

$$\bar{a}_1 > \frac{\phi_j\left(\sup V_j^2\right) - \phi_j\left(\inf V_j^2\right)}{\left(\sup V_j^2 - \inf V_j^2\right)}. \qquad (64)$$

We need to minimize the value of $\bar{a}_1$, which means that the optimal value of $\bar{a}_1$ is

$$\bar{a}_1 = \frac{\phi_j\left(\sup V_j^2\right) - \phi_j\left(\inf V_j^2\right)}{\left(\sup V_j^2 - \inf V_j^2\right)}. \qquad (65)$$

In a similar way, we can prove that we have for the $\underline{a}_1$ value that the minimization problem of

$$\begin{aligned}
&\min_{a_1} \int_{\inf V_j^2}^{\sup V_j^2} \left(\phi_j - P^1_{\phi_j,v_j^2}\right) dv_j^2 \rightarrow \\
&\min_{\bar{a}_1} \left[\frac{\bar{a}_1}{2}\left(\sup V_j^2 - \inf V_j^2\right)^2\right] \qquad (66)
\end{aligned}$$

is solved by taking the lowest possible value of $\underline{a}_1$. This value, however, is a constraint by the following relation:

$$\underline{a}_1 > \frac{\phi_j\left(v_j^2\right) - \phi_j\left(\sup V_j^2\right)}{\left(v_j^2 - \sup V_j^2\right)}. \qquad (67)$$

It is stated without proof that the relation on the right-hand side is monotonically increasing [proof is the same as for relation (57)] and that the maximal value is found when taking the following limit:

$$\lim_{v_j^2 \rightarrow \sup V_j^2} \left(\frac{\phi_j\left(v_j^2\right) - \phi_j\left(\sup V_j^2\right)}{\left(v_j^2 - \sup V_j^2\right)}\right) = \left.\frac{\partial \phi_j}{\partial v_j^2}\right|_{\sup V_j^2} \qquad (68)$$

Since we want to minimize the integrated error, we need to minimize $\underline{a}_1$, which means we have the optimial value

$$\underline{a}_1 = \left.\frac{\partial \phi_j}{\partial v_j^2}\right|_{\sup V_j^2}. \qquad (69)$$

This completes the proof. ■

Proof of Theorem IV.2: The derivative of $\phi_j(v_j)$ is

$$\frac{\partial \phi_j}{\partial v_j} = 1 - \phi_j^2 \qquad (70)$$

which is positive on the domain V_j since $\phi_j \in [-1, 1] \forall v_j \in V_j$. This means that $\phi_j(v_j)$ is monotonically increasing on domain V_j and that the minimal value of ϕ_j is attained at $v_j = \inf V_j$

and the maximal value at $v_j = \sup V_j$. By Theorem II.1, we have that

$$\begin{aligned}\phi_j(\inf V_j) &= \inf \Phi_j(V_j)\\ \phi_j(\sup V_j) &= \sup \Phi_j(V_j).\end{aligned} \tag{71}$$

Since we must have $\sup \Phi_j(V_j) \geq P^1_{\phi_j,v_j} \geq \phi(v_j)\, \forall v_j \in V_j$, we known that

$$\begin{aligned}\bar{a}_1 \sup V_j + \bar{a}_0 &= \phi_j(\sup V_j)\\ \underline{a}_1 \inf V_j + \underline{a}_0 &= \phi_j(\inf V_j)\\ \Rightarrow\ \bar{a}_0 &= \phi_j(\sup V_j) - \bar{a}_1 \sup V_j\\ \underline{a}_0 &= \phi_j(\inf V_j) - \underline{a}_1 \inf V_j.\end{aligned} \tag{72}$$

Now we only need to specify the value of $\bar{a}_1$ and $\underline{a}_1$. Let us first look at the value of $\bar{a}_1$. Since we require $P^1_{\phi_j,v_j} > \phi_j(v_j) \forall v_j \in V_j$, we can find the optimal value of $\bar{a}_1$ by minimizing the integrated error between the polynomial and the activation function

$$\begin{aligned}&\min_{\bar{a}_1} \int_{\inf V_j}^{\sup V_j} \left(P^1_{\phi_j,v_j} - \phi_j\right) dv_j \rightarrow \min_{\bar{a}_1} \int_{\inf V_j}^{\sup V_j} P^1_{\phi_j,v_j} dv_j\\ &= \min_{\bar{a}_1} \int_{\inf V_j}^{\sup V_j} (\bar{a}_1 v_j - \bar{a}_1 \sup V_j) dv_j\\ &= \min_{\bar{a}_1} \left[-\frac{\bar{a}_1}{2} (\sup V_j - \inf V_j)^2\right].\end{aligned} \tag{73}$$

The term $(\sup V_j - \inf V_j)^2$ is always positive thus we know that the minimization of the integral can be seen as a maximization of $\bar{a}_1$. The largest possible value of $\bar{a}_1$ is determined by the constraint

$$P^1_{\phi_j,v_j} > \phi_j(v_j) \qquad \forall v_j \in V_j. \tag{74}$$

This relation can be rewritten into

$$\begin{aligned}&\bar{a}_1 v_j - \bar{a}_1 \sup V_j + \phi_j(\sup V_j) > \phi_j(v_j)\\ &\bar{a}_1 (v_j - \sup V_j) > \phi_j(v_j) - \phi_j(\sup V_j)\\ &\bar{a}_1 < \frac{\phi_j(v_j) - \phi_j(\sup V_j)}{(v_j - \sup V_j)}.\end{aligned} \tag{75}$$

The right-hand side is always positive and, depending of the value $\sup V_j$, has one or no point in domain V_j where the derivative is zero. If the derivative does become zero within domain V_j, then the second derivative will be negative at that location. This means that the minimal value of the right-hand side of (75) will be obtained on the border of domain V_j

$$\begin{aligned}&\min_{v_j \in V_j} \frac{\phi_j(v_j) - \phi_j(\sup V_j)}{(v_j - \sup V_j)}\\ &= \min \begin{cases} \dfrac{\phi_j(\inf V_j) - \phi_j(\sup V_j)}{(\inf V_j - \sup V_j)} \\ \lim_{v_j \rightarrow \sup V_j} \dfrac{\phi_j(v_j) - \phi_j(\sup V_j)}{(v_j - \sup V_j)} = \left.\dfrac{\partial \phi_j}{\partial v_j}\right|_{\sup V_j} \end{cases}.\end{aligned} \tag{76}$$

Since we need to fulfill the constraint and maximize $\bar{a}_1$, the optimal value of $\bar{a}_1$ will be

$$\bar{a}_1 = \min \left(\left.\frac{\partial \phi_j}{\partial v_j}\right|_{\sup V_j}, \frac{\phi_j(\inf V_j) - \phi_j(\sup V_j)}{(\inf V_j - \sup V_j)} \right). \tag{77}$$

In a similar way, we can prove that we have for the $\underline{a}_1$ value the minimization problem of

$$\min_{\underline{a}_1} \int_{\inf V_j}^{\sup V_j} \left(\phi_j - P^1_{\phi_j,v_j}\right) dv_j \rightarrow \min_{\underline{a}_1} \left[-\frac{\underline{a}_1}{2} (\sup V_j - \inf V_j)^2\right] \tag{78}$$

which is solved by taking the highest possible value of $\underline{a}_1$. This value is, however, a constraint by the following relation:

$$\underline{a}_1 < \frac{\phi_j(\inf V_j) - \phi_j(v_j)}{(\inf V_j - v_j)}. \tag{79}$$

It is stated without proof [which is similar as for (75)] that the minimal value of the right-hand side is obtained at the border of domain V_j. We need to maximize the value of $\underline{a}_1$ to minimize the integrated error, which means that the optimal value of $\underline{a}_1$ is

$$\underline{a}_1 = \min \left(\left.\frac{\partial \phi_j}{\partial v_j}\right|_{\inf V_j}, \frac{\phi_j(\inf V_j) - \phi_j(\sup V_j)}{(\inf V_j - \sup V_j)} \right). \tag{80}$$

This completes the proof.

REFERENCES

[1] G. Zhang, *Neural Networks in Business Forecasting*. Hershey, PA: Information Science Publishing, 2003.

[2] K. Smith and J. Gupta, *Neural Networks in Business: Techniques and Applications*. Hershey, PA: IGI Global, 2002.

[3] P. Bhagat, *Pattern Recognition in Industry*. New York: Elsevier, 2005.

[4] C. Bishop, *Neural Networks for Pattern Recognition*. Oxford, U.K.: Oxford Univ. Press, 1995.

[5] B. Ripley, *Pattern Recognition and Neural Networks*. Cambridge, U.K.: Cambridge Univ. Press, 1996.

[6] R. Duda, P. Hart, and D. Stork, *Pattern Classification*, 2nd ed. New York: Wiley, 2001.

[7] M. Smith, *Neural Networks for Statistical Modeling*. New York: Van Nostrand, 1993.

[8] S. Haykin, *Neural Networks a Comprehensive Foundation*. Upper Saddle River, NJ: Prentice-Hall, 1994.

[9] W. McCulloch and W. Pitts, "A logical calculus of the ideas immanent in nervous activity," *Bull. Math. Biophys.*, vol. 5, pp. 115–133, 1943.

[10] P. Werbos, *The Roots of Backpropagation*. New York: Wiley, 1994.

[11] Y. Shang and B. Wah, "Global optimization for neural network training," *IEEE Comput. Sci. Eng.*, vol. 29, no. 3, pp. 45–54, Mar. 1996.

[12] B. Cetin, J. Burdick, and J. Barhen, "Global descent replaces gradient descent to avoid local minima problem in learning with artificial neural networks," in *Proc. IEEE Int. Conf. Neural Netw.*, 1993, pp. 836–842.

[13] N. Baba, Y. Mogami, M. Kohzaki, Y. Shiraishi, and Y. Yoshida, "A hybrid algorithm for finding the global minimum of error function of neural networks and its application," *Neural Netw.*, vol. 7, no. 8, pp. 1253–1265, 1994.

[14] L. Hamm, B. Brorsen, and M. Hagan, "Comparison of stochastic global optimization methods to estimate neural network weights," *Neural Process. Lett.*, vol. 26, pp. 145–158, 2007.

[15] K. Toh, "Deterministic global optimization for FNN training," *IEEE Trans. Syst. Man Cybern. B, Cybern.*, vol. 33, no. 6, pp. 977–983, Dec. 2003.

[16] Z. Tang and G. Koehler, "Deterministic global optimal FNN training algorithms," *Neural Netw.*, vol. 7, pp. 301–311, 1994.

[17] H. Li, H. Li, and Y. Du, "A global optimization algorithm based on novel interval analysis for training neural networks," in *Advances in Computation and Intelligence*. Cambridge, MA: MIT Press, 2007, vol. 4683, pp. 286–295.
[18] E. de Weerdt, Q. Chu, and J. Mulder, "Continuous state and action advantage-learning using interval analysis and neural networks," in *Proc. AIAA Guid. Navigat. Control Conf. Exhibit*, 2007, AIAA-2007-6522.
[19] R. S. Sutton and A. G. Barto, , T. Dietterich, Ed., *Reinforcement Learning, An Introduction*, 3rd ed. Cambridge, MA: MIT Press, 1998, printing (2000).
[20] L. Baird, "Reinforcement learning in continuous time: Advantage updating," in *Proc. IEEE Int. Conf. Neural Netw.*, Jun.–Jul. 1994, vol. 4, pp. 2448–2453.
[21] D. Bertsekas and J. Tsitsiklis, *Neuro-Dynamic Programming*. Belmont, MA: Athena Scientific, 1996, vol. 1.
[22] G. Cybenko, "Approximations by superpositions of a sigmoidal function," *Math. Control Signals Syst.*, vol. 2, pp. 303–314, 1989.
[23] R. Moore, *Interval Analysis*. Englewood Cliffs, NJ: Prentice-Hall, 1966.
[24] E. van Kampen, Q. Chu, J. Mulder, and M. van Emden, "Nonlinear aircraft trim using interval analysis," in *Proc. AIAA Guid. Navigat. Control Conf. Exhibit*, Aug. 2007, AIAA-2007-6766.
[25] S. Juliana, Q. Chu, J. Mulder, and T. van Baten, "Flight envelope clearance of atmospheric re-entry module with flight control," in *Proc. AIAA Guid. Navigat. Control Conf. Exhibit*, 2004, AIAA-2004-5170.
[26] E. de Weerdt, E. van Kampen, Q. Chu, and J. Mulder, "Integer ambiguity resolution using interval analysis," *J. Navigation*, 2008, to be published, .
[27] C. Hu and L. T. He, "An application of interval methods to stock market forecasting," *Reliable Comput.*, vol. 13, no. 5, pp. 423–434, Oct. 2007.
[28] J. Garloff, I. Idriss, and A. Smith, "Guaranteed parameter set estimation for exponential sums: The three-terms case," *Reliable Comput.*, vol. 13, no. 4, pp. 351–359, Aug. 2007.
[29] A. Neumaier, "Taylor forms—Use and limits," *Reliable Comput.*, vol. 9, pp. 43–79, 2002.
[30] K. Makino and M. Berz, "Taylor models and other validated functional inclusion functions," *Int. J. Pure Appl. Math.*, vol. 4, no. 4, pp. 379–356, 2003.
[31] K. Makino and M. Berz, "Higher order multivariate automatic differentiation and validated computation of remainder bounds," *Trans. Comput.*, vol. 4, pp. 1611–1618, Nov. 2005.
[32] K. Makino and M. Berz, "New applications of Taylor model methods," *Automatic Differentiation of Algorithms: From Simulation to Optimization*, pp. 359–364, 2000, ISBN: 0-387-95305-1.
[33] K. Makino and M. Berz, "Higher order verified inclusions of multidimensional systems by Taylor models," *Nonlinear Anal.*, vol. 47, pp. 3503–3514, 2001.
[34] M. Berz and G. Hoffstatter, "Computation and application of Taylor polynomials with interval remainder bounds," *Reliable Comput.*, vol. 4, pp. 83–97, 1998.
[35] E. Hansen and G. Walster, *Global Optimization Using Interval Analysis*, 2nd ed. New York: Marcel Dekker/Sum Microsystems, 2004.
[36] R. Moore, *Methods and Applications of Interval Analysis*. Philadelphia, PA: SIAM, 1979.
[37] L. Jaulin, M. Kieffer, O. Didrit, and E. Walter, *Applied Interval Analysis*. New York: Springer-Verlag, 2001.
[38] E. Hansen, *A Generalized Interval Arithmetic*. Berlin, Germany: Springer-Verlag, 1975, vol. 29.

E. de Weerdt received the M.Sc. degree in aerospace engineering with honors from the Faculty of Aerospace Engineering, Delft University of Technology, Delft, The Netherlands, in 2005, where he is currently working towards the Ph.D. degree at the Division of Control and Simulation.

His research field ranges from neural networks to flight dynamics and control, satellite formation flying, nonlinear control, adaptive control, and global optimization.

Q. P. Chu received the Ph.D. degree in aerospace engineering from the Faculty of Aerospace Engineering, Delft University of Technology, Delft, The Netherlands, in 1987.

Currently, he is an Associate Professor at the Division of Control and Simulation, Faculty of Aerospace Engineering, Delft University of Technology, responsible for aerospace guidance, navigation and control education, and research activities. His research field ranges from adaptive control, nonlinear control, robust control, and intelligent control for aerospace vehicles to precise navigation, system identification, and nonlinear optimization. He is also the responsible teacher for courses on aerospace GNC.

Dr. Chu was the designer of the attitude control system for the third Dutch satellite Sloshsat launched in Feb. 2005. He is reviewer for many international journals in aerospace guidance navigation and control and a member of the American Institute of Aeronautics and Astronautics (AIAA).

J. A. Mulder received the Ph.D. degree with honors from Delft University of Technology, Delft, The Netherlands, in 1986.

In 1989 he was appointed Full Professor at Delft University of Technology, where he is the Head of the Control and Simulation Division, Faculty of Aerospace Engineering. He serves as Scientific Director of the Institute for Research in Simulation, Motion and Navigation Technologies (SIMONA) and the Institute for Aerospace Software and Technologies (ASTI) of Delft University of Technology.

Dr. Mulder has served in numerous ministerial advisory committees, as advisor to the board of the National Aerospace Laboratory (NLR) and the Radiobiological Institute of the Netherlands Organisation for Applied Scientific Research (TNO). He has been a member of the AGARD Flight Mechanics panel from 1982 to 1997. Currently, he is a member of the Avionics Committee of the Society of Automotive Engineers (SAE), member of the American Institute of Aeronautics and Astronautics (AIAA) AFM Technical Committee and Associate Editor of the *AIAA Journal of Aerospace Computing, Information, and Communication*.