Neural Comput & Applic (2013) 23:283–292
DOI 10.1007/s00521-012-0891-8

ORIGINAL ARTICLE

# Surface reconstruction based on extreme learning machine

**Zheng Hua Zhou · Jian Wei Zhao · Fei Long Cao**

Received: 14 November 2011 / Accepted: 11 February 2012 / Published online: 28 February 2012


**Abstract** In this paper, extreme learning machine (ELM) is used to reconstruct a surface with a high speed. It is shown that an improved ELM, called polyharmonic extreme learning machine (P-ELM), is proposed to reconstruct a smoother surface with a high accuracy and robust stability. The proposed P-ELM improves ELM in the sense of adding a polynomial in the single-hidden-layer feedforward networks to approximate the unknown function of the surface. The proposed P-ELM can not only retain the advantages of ELM with an extremely high learning speed and a good generalization performance but also reflect the intrinsic properties of the reconstructed surface. The detailed comparisons of the P-ELM, RBF algorithm, and ELM are carried out in the simulation to show the good performances and the effectiveness of the proposed algorithm.

**Keywords** Surface reconstruction · Feedforward neural networks · Extreme learning machine · Polyharmonic extreme learning machine

## 1 Introduction

Surface reconstruction from a set of sample data is still a challenging topic in the areas of computer graphics, mathematics, and practical applications. With the development of technology and widespread usage of three-dimensional (3D) scanners, surface reconstructions have been playing a very important role. It is well known that a scanner samples the data from the surface of an object, called the underlying surface, expressed in the form of the 3D Cartesian coordinates. After these sample data are loaded to the computer, a surface will be reconstructed. Because of high-resolution of modern 3D scanners, it is possible to acquire large amount of data, called point clouds data, which needs a great deal of computation in the process of surface reconstruction. Therefore, it is necessary to explore the algorithms that can reconstruct the smooth surface with a high speed.

Recently, many algorithms have been developed to perform the surface reconstruction from the sampled data. It has been noted that some approaches are based on combinatorial structures, such as Delaunay triangulations [1–2], alpha shapes [3–5], Voronoi diagrams [6], and so on. These methods are capable of creating a triangle mesh that interpolates all or part of the points. However, in the case of noisy data, the restructed surface loses information. Thus, it is essential to do the smoothing processing (see [2, 4]). Some of other approaches directly reconstruct an approximating surface, typically representing the surface in implicit form (see [7–9]).

From the viewpoint of approximation, surface reconstruction is the interpolation or approximation of some unknown functions of the surfaces from sampled data. Since single-hidden-layer feedforward networks (SLFNs) can approximate complex nonlinear functions arbitrarily from the sampled data [10–16], there have been a lot of research focuses on the surface reconstruction with SLFNs [17–23]. In order to design the SLFN for surface reconstruction, backpropagation (BP) have been widely used to train weights of the SLFN. However, BP training is time-consuming and is also easily trapped in a local minimum.

Z. H. Zhou · J. W. Zhao · F. L. Cao (✉)
Department of Mathematics, China Jiliang University, Hangzhou 310018, Zhejiang Province, People's Republic of China
e-mail: feilongcao@gmail.com

In many cases, the dimension of sampled data vectors from a surface is very large and the large number of hidden nodes of the SLFN is often required in order to perform the accurate approximation, which may results in an increased training time.

In this paper, we will first describe the surface reconstruction from sample data with ELM, proposed by Huang et al. [24], to improve the learning speed, the smoothness of the surface, and the stability of the algorithm. In order to further improve the smoothness and accuracy of the reconstructed surface, a novel learning algorithm, called polyharmonic extreme learning machine (P-ELM), based on ELM, is proposed. It will be shown that the proposed P-ELM improves the smoothness of the reconstructed surface by adding a low-order polynomial to the SLFN. Furthermore, since the hidden weights of the proposed P-ELM are randomly assigned and the output weights are optimized with the least square method, the training time is greatly reduced compared with many other existing learning algorithms, such as BP and RBF algorithm. From the comparisons of performances of P-ELM, ELM, and RBF algorithm for surface reconstruction, the proposed P-ELM has demonstrated the best accuracy, smoothness, and stability.

The rest of this paper is organized as follows. Section 2 describes the surface reconstruction with ELM. Section 3 proposes P-ELM for surface reconstruction to improve the smoothness and accuracy of the reconstructed surface. Performance evaluation of the proposed P-ELM, ELM, and RBF algorithm on computational complexity, accuracy, and stability on the surface reconstruction is given in Sect. 4. Conclusions are given in Sect. 5.

## 2 Surface reconstruction by ELM

Consider the following set of surface data

$$\{(\mathbf{x}_i, t_i)\}_{i=1}^{N} \subseteq D \times \mathbb{R}, \tag{2.1}$$

where $t_i$ is the corresponding observed value of the point $\mathbf{x}_i, i = 1, 2, \ldots, N$ and $D$ is a bounded simply connected subset of the Euclidean space $\mathbb{R}^2$. The goal of surface reconstruction is to find a function $f \in C^{(2)}(D)$ to interpolate or approximate the sample data in (2.1), where $C^{(2)}(D)$ denotes the set of functions whose second-order partial derivatives exist and are continuous on $D$.

In this paper, we will adopt the following set of SLFNs

$$\mathcal{N} = \left\{ f | f(\mathbf{x}) = \sum_{i=1}^{L} \beta_i G(\mathbf{a}_i, b_i, \mathbf{x}), \mathbf{a}_i \in \mathbb{R}^2, b_i, \beta_i \in \mathbb{R}, L \in \mathbb{N} \right\}, \tag{2.2}$$

to interpolate or approximate the surface function, where $\mathbf{a}_i$ and $b_i$ are the parameters of hidden nodes, $\beta_i$ is the weight connecting the $i$th hidden node with the output node, $G(\mathbf{a}_i, b_i, \mathbf{x})$ is the output of the $i$th hidden node with respect to the input vector $\mathbf{x}, i = 1, 2, \ldots, L$ and $L$ is the number of hidden nodes. There are many computational hidden nodes for $G(\mathbf{a}_i, b_i, \mathbf{x})$, such as additive/RBF hidden nodes, multiplicative nodes, high-order nodes, ridge polynomials, wavelets, etc. Additive and RBF hidden nodes are often used in applications for their simple structures.

For the additive hidden nodes, $G(\mathbf{a}_i, b_i, \mathbf{x})$ is expressed as

$$G(\mathbf{a}_i, b_i, \mathbf{x}) = g(\mathbf{a}_i \cdot \mathbf{x} + b_i), \tag{2.3}$$

where $\mathbf{a}_i \cdot \mathbf{x}$ denotes the inner product of the vectors $\mathbf{a}_i$ and $\mathbf{x}$ in the Euclidean space $\mathbb{R}^2$, and $g : \mathbb{R} \to \mathbb{R}$ is an activation function that can be taken to be a bounded nonconstant function in $C^{(2)}(\mathbb{R})$ to guarantee the approximation ability of SLFNs, for example,

$$g(t) = \frac{1}{1 + e^{-t}}, \quad t \in \mathbb{R}. \tag{2.4}$$

It is noted that for RBF hidden nodes, $G(\mathbf{a}_i, b_i, \mathbf{x})$ is expressed as

$$G(\mathbf{a}_i, b_i, \mathbf{x}) = g(b_i \parallel \mathbf{x} - \mathbf{a}_i \parallel), \tag{2.5}$$

where $\mathbf{a}_i$ and $b_i (b_i > 0) (i = 1, 2, \ldots, L)$ are the centers and impact factors of the $i$-th RBF hidden node, respectively. And $g : \mathbb{R} \to \mathbb{R}$ is an activation function that can be chosen to be a function in $C^{(2)}(\mathbb{R})$ to guarantee the approximation ability of SLFNs, for example,

$$g(t) = e^{-t^2}, \quad t \in \mathbb{R}. \tag{2.6}$$

There are a lot of learning algorithms to determine the weights $\mathbf{a}_i, b_i$ and $\beta_i$ of the SLFN from a set of sample data, including perceptron, BP, genetic algorithm, and so on. All these algorithms should adjust all the weights by some iterations, which results in complex computation and costs a lot of training time. Since the number of sample data is huge, it is urgent to choose an effective learning algorithm. Fortunately, the ELM, proposed by Huang et al. [26], randomly chooses the hidden weights and determines the output weight by solving a linear system, so it has a higher learning speed than above-mentioned learning algorithms. What is more, Huang et al. has proved that SLFNs with wide types of random computational hidden nodes have the universal approximation capability as long as SLFNs with these types of adjustable hidden nodes can be universal approximators [27]. Therefore, we will use the ELM to reconstruct surface from a set of sample data.

Given the set of sample data $\{(\mathbf{x}_i,t_i)\}_{i=1}^N$ in (2.1), an SLFN with randomly assigned hidden weights $\mathbf{a}_i^*$ and $b_i^*$ is expressed as follows:

$$f(\mathbf{x}) = \sum_{i=1}^{L} \beta_i G(\mathbf{a}_i^*, b_i^*, \mathbf{x}). \tag{2.7}$$

Then, the SLFN in (2.7) approximating those $N$ sample data best means that there exists $\beta_i$ $(i = 1, 2, \ldots, L)$ and $\epsilon_j (j = 1, 2, \ldots, N)$ such that

$$\sum_{i=1}^{L} \beta_i G(\mathbf{a}_i^*, b_i^*, \mathbf{x}_j) = t_j + \epsilon_j, \quad j = 1, 2, \ldots, N, \tag{2.8}$$

where each $\epsilon_j$ is the observed error with respect to $t_j$. The system of linear equations in (2.8) can be rewritten compactly as

$$\mathbf{H}\beta = \mathbf{T} + \mathbf{E}, \tag{2.9}$$

where

$$\mathbf{H} = \begin{bmatrix} G(\mathbf{a}_1^*, b_1^*, \mathbf{x}_1) & \cdots & G(\mathbf{a}_L^*, b_L^*, \mathbf{x}_1) \\ \vdots & \ddots & \vdots \\ G(\mathbf{a}_1^*, b_1^*, \mathbf{x}_N) & \cdots & G(\mathbf{a}_L^*, b_L^*, \mathbf{x}_N) \end{bmatrix}_{N\times L}, \tag{2.10}$$

$$\beta = \begin{bmatrix} \beta_1 \\ \vdots \\ \beta_L \end{bmatrix}, \quad \mathbf{T} = \begin{bmatrix} t_1 \\ \vdots \\ t_N \end{bmatrix} \quad \text{and} \quad \mathbf{E} = \begin{bmatrix} \epsilon_1 \\ \vdots \\ \epsilon_N \end{bmatrix}. \tag{2.11}$$

As named by Huang et al. in the paper [26], $\mathbf{H}$ in (2.10) is called the hidden-layer output matrix of the SLFN.

With the constraint of minimum norm least square, that is, $\min\|\beta\|$ and $\min\|\mathbf{H}\beta - \mathbf{T}\|$, the approximation solution $\hat{\beta}$ of the system (2.9) can be expressed as

$$\hat{\beta} = \mathbf{H}^{\dagger}\mathbf{T}, \tag{2.12}$$

where $\mathbf{H}^{\dagger}$ is the Moore-Penrose inverse of $\mathbf{H}$ [25]. When $\mathbf{H}^{\top}\mathbf{H}$ is nonsingular,

$$\mathbf{H}^{\dagger} = (\mathbf{H}^{\top}\mathbf{H})^{-1}\mathbf{H}^{\top}; \tag{2.13}$$

otherwise, $\mathbf{H}^{\dagger}$ can be computed by many methods, including orthogonalization method, orthogonal projection, iterative method, and singular value decomposition (SVD) method [28].

Based on above analysis, surface reconstruction with ELM can be summarized as follows.

Surface reconstruction by ELM 2.1

For a given sample data $\{(\mathbf{x}_i, t_i)\}_{i=1}^N \subseteq D \times \mathbb{R}$, choose an SLFN with $L$ hidden nodes as follow

$$f(\mathbf{x}) = \sum_{i=1}^{L} \beta_i G(\mathbf{a}_i, b_i, \mathbf{x}) \tag{2.14}$$

continued

Surface reconstruction by ELM 2.1

to represent the surface to be reconstructed, where the hidden-node output function $G(\mathbf{a}, b, \mathbf{x})$ and the number $L$ of hidden nodes are chosen in prior.

**Step 1**. Randomly assign hidden-layer weights $(\mathbf{a}_i^*, b_i^*), i = 1, 2, \ldots, L$.

**Step 2**. Compute the hidden-layer output matrix

$$\mathbf{H} = \begin{bmatrix} G(\mathbf{a}_1^*, b_1^*, \mathbf{x}_1) & \cdots & G(\mathbf{a}_L^*, b_L^*, \mathbf{x}_1) \\ \vdots & \ddots & \vdots \\ G(\mathbf{a}_1^*, b_1^*, \mathbf{x}_N) & \cdots & G(\mathbf{a}_L^*, b_L^*, \mathbf{x}_N) \end{bmatrix}_{N\times L}. \tag{2.15}$$

**Step 3**. Calculate the output weight vector

$$\hat{\beta} = \mathbf{H}^{\dagger}\mathbf{T}, \tag{2.16}$$

where $\mathbf{T} = [t_1, t_2, \ldots, t_N]^{\top}$.

**Step 4**. The SLFN $f$ in (2.14) for representing the reconstructed surface is determined by

$$f(\mathbf{x}) = \sum_{i=1}^{L} \hat{\beta}_i G(\mathbf{a}_i^*, b_i^*, \mathbf{x}), \tag{2.17}$$

where $\hat{\beta} = [\hat{\beta}_1, \hat{\beta}_2, \ldots, \hat{\beta}_L]^{\top}$.

*Remark 2.2* In surface reconstruction, it is often required that the reconstructed surface is smooth enough. From the structure of SLFNs, the requirement can be easily implemented by choosing an activation function in $C^{(2)}(D)$.

*Remark 2.3* In general, the number $L$ of hidden nodes is chosen initially. If

$$E(f) = \left(\frac{1}{N}\sum_{i=1}^{N}(f(\mathbf{x}_i) - t_i)^2\right)^{\frac{1}{2}} \tag{2.18}$$

is not small enough, we can increase the number of hidden nodes to reduce $E(f)$ according to the incremental learning rule [29].

## 3 Surface reconstruction by the proposed P-ELM

When the surface to be reconstruction contains some flat parts, it often does not work well to reconstruct the surface just with an SLFN. Based on Remark 2.3, we can increase the number of hidden nodes to reduce the training error according to the theory of the incremental learning rule. However, with the increase of hidden nodes, the computational complexity of ELM also increases simultaneously. Furthermore, smoothness of the reconstructed surface is also one of important problems that should be considered in reconstructing geometric modeling. In general, it requires that the function representing the surface is at least $C^{(2)}$ continuous, that is, the second-order partial derivatives are continuous.

In order to characterize the smoothness of the surface to be reconstructed, the concept of surface energy is often employed in surface reconstruction [30]. For a function $f \in C^{(2)}(D)$, the surface energy is measured by the following semi-norm

$$\|f\|_F = \left( \int_D \left[ \left( \frac{\partial^2 f}{\partial x_1{}^2} \right)^2 + \left( \frac{\partial^2 f}{\partial x_2{}^2} \right)^2 + 2 \left( \frac{\partial^2 f}{\partial x_1 \partial x_2} \right)^2 \right] \mathrm{d}x_1 \, \mathrm{d}x_2 \right)^{\frac{1}{2}}, \tag{3.1}$$

where $D$ is a bounded simply connected subset of the Euclidean space $\mathbb{R}^2$. The semi-norm $\|\cdot\|_F$ in (3.1) is the rotation-invariant semi-norm. From the viewpoint of physics, the surface with a small energy is smoother than that with a larger one. That is, the function with a smaller semi-norm is smoother than that with a larger one. Therefore, for the following set $\mathcal{F}$ of interpolators

$$\mathcal{F} = \{ f \in \mathcal{N} : f(\mathbf{x}_i) = t_i + \epsilon_i, \quad i = 1, 2, \ldots, N \}, \tag{3.2}$$

the smoothest interpolator $f^*$ for the sample data $\{(\mathbf{x}_i, t_i)\}_{i=1}^N$ is

$$f^* = arg \min_{f \in \mathcal{F}} \|f\|_F. \tag{3.3}$$

By the theory of minimizing rotation-invariant semi-norms in Sobolev space, $f^*$ usually has the following simple form

$$f^*(\mathbf{x}) = \sum_{i=1}^{N} \beta_i G(\mathbf{a}_i, b_i, \mathbf{x}) + \gamma \mathbf{x} + c, \tag{3.4}$$

where $\gamma \in \mathbb{R}^2$ and $c \in \mathbb{R}$ (see [30]). Therefore, we will realize the surface reconstruction with the following class

$$\mathcal{M} = \left\{ f | f(\mathbf{x}) = \sum_{i=1}^{L} \beta_i G(\mathbf{a}_i, b_i, \mathbf{x}) + \gamma \mathbf{x} + c, \mathbf{a}_i, \gamma \in \mathbb{R}^2, b_i, \beta_i, c \in \mathbb{R}, L \in \mathbb{N} \right\} \tag{3.5}$$

instead of the class $\mathcal{N}$ of SLFNs in (2.2).

For the set of sample data $\{(\mathbf{x}_i, t_i)\}_{i=1}^N$ in (2.1), we will use the idea of ELM to choose the best approximator in $\mathcal{M}$. That is, we randomly assign the hidden weights of the function

$$f(\mathbf{x}) = \sum_{i=1}^{L} \beta_i G(\mathbf{a}_i, b_i, \mathbf{x}) + \gamma \mathbf{x} + c \tag{3.6}$$

with $\mathbf{a}_i^*$ and $b_i^*, i = 1, 2, \ldots, L$. Then, $f$ approximating those $N$ sample data best means that there exists $\beta_i \, (i = 1, 2, \ldots, L), \gamma, c$ and $\epsilon_j \, (j = 1, 2, \ldots, N)$ such that

$$\sum_{i=1}^{L} \beta_i G(\mathbf{a}_i^*, b_i^*, \mathbf{x}_j) + \gamma \mathbf{x}_j + c = t_j + \epsilon_j, \quad j = 1, 2, \ldots, N. \tag{3.7}$$

Let

$$\tilde{\mathbf{H}} = \begin{bmatrix} G(\mathbf{a}_1^*, b_1^*, \mathbf{x}_1) & \cdots & G(\mathbf{a}_L^*, b_L^*, \mathbf{x}_1) & x_{11} & x_{12} & 1 \\ \vdots & \ddots & \vdots & \vdots & \vdots & \vdots \\ G(\mathbf{a}_1^*, b_1^*, \mathbf{x}_N) & \cdots & G(\mathbf{a}_L^*, b_L^*, \mathbf{x}_N) & x_{N1} & x_{N2} & 1 \end{bmatrix}_{N \times (L+3)}, \tag{3.8}$$

$$\tilde{\beta} = \begin{bmatrix} \beta_1 \\ \vdots \\ \beta_L \\ \gamma_1 \\ \gamma_2 \\ c \end{bmatrix}, \mathbf{T} = \begin{bmatrix} t_1 \\ t_2 \\ t_3 \\ \vdots \\ t_{N-1} \\ t_N \end{bmatrix}, \quad \text{and} \quad \mathbf{E} = \begin{bmatrix} \epsilon_1 \\ \epsilon_2 \\ \epsilon_3 \\ \vdots \\ \epsilon_{N-1} \\ \epsilon_N \end{bmatrix}, \tag{3.9}$$

where $\mathbf{x}_i = [x_{i1}, x_{i2}]^\top, i = 1, 2, \ldots, N$, then the system of equations in (3.7) can be rewritten compactly as

$$\tilde{\mathbf{H}} \tilde{\beta} = \mathbf{T} + \mathbf{E}. \tag{3.10}$$

With the Moore-Penrose inverse $\tilde{\mathbf{H}}^\dagger$ of the matrix $\tilde{\mathbf{H}}$, the output weight vector $\tilde{\beta}$ can be computed by

$$\tilde{\beta} = \tilde{\mathbf{H}}^\dagger \mathbf{T}. \tag{3.11}$$

Based on above analysis, the surface reconstruction with a new algorithm, called polyharmonic extreme learning machine (P-ELM), can be summarized as follows:

---

Surface reconstruction by P-ELM 3.1

---

For a given sample data $\{(\mathbf{x}_i, t_i)\}_{i=1}^N \subseteq D \times \mathbb{R}$, choose a form of the approximator $f$ as follows

$$f(\mathbf{x}) = \sum_{i=1}^{L} \beta_i G(\mathbf{a}_i, b_i, \mathbf{x}) + \gamma \mathbf{x} + c \tag{3.12}$$

to represent the surface to be reconstructed, where the hidden-node output function $G(\mathbf{a}, b, \mathbf{x})$ and the hidden-node number $L$ are chosen in prior.

**Step 1**. Randomly assign hidden-layer weights $(\mathbf{a}_i^*, b_i^*), i = 1, 2, \ldots, L$.

**Step 2**. Compute the matrix

$$\mathbf{H} = \begin{bmatrix} G(\mathbf{a}_1^*, b_1^*, \mathbf{x}_1) & \cdots & G(\mathbf{a}_L^*, b_L^*, \mathbf{x}_1) & x_{11} & x_{12} & 1 \\ \vdots & \ddots & \vdots & \vdots & \vdots & \vdots \\ G(\mathbf{a}_1^*, b_1^*, \mathbf{x}_N) & \cdots & G(\mathbf{a}_L^*, b_L^*, \mathbf{x}_N) & x_{N1} & x_{N2} & 1 \end{bmatrix}_{N \times (L+3)}, \tag{3.13}$$

where $\mathbf{x}_i = [x_{i1}, x_{i2}]^\top, i = 1, 2, \ldots, N$.

**Step 3**. Calculate the output weight vector by

$$\tilde{\beta} = \tilde{\mathbf{H}}^\dagger \mathbf{T}, \tag{3.14}$$

where $\tilde{\beta} = [\tilde{\beta}_1, \tilde{\beta}_2, \ldots, \tilde{\beta}_L, \tilde{\gamma}_1, \tilde{\gamma}_2, \tilde{c}]^\top, \tilde{\gamma} = [\tilde{\gamma}_1, \tilde{\gamma}_2]^\top$ and $\mathbf{T} = [t_1, t_2, \ldots, t_N]^\top$.

**Step 4**. The function $f$ in (3.12) representing the reconstructed surface is determined by

$$f(\mathbf{x}) = \sum_{i=1}^{L} \tilde{\beta}_i G(\mathbf{a}_i^*, b_i^*, \mathbf{x}) + \tilde{\gamma} \mathbf{x} + \tilde{c}. \tag{3.15}$$

---

*Remark 3.2* When $\gamma = \mathbf{0}$ and $c = 0$, the surface reconstructed by the proposed P-ELM 3.1 is the one reconstructed by the ELM 2.1.

*Remark 3.3* Note that the coefficients of the polynomial

$$P(\mathbf{x}) = \gamma \mathbf{x} + c \tag{3.16}$$

is determined by the linear system in P-ELM, not randomly assigned. We have once tried to regard it as the $(L + 1)$th hidden node of an SLFN, that is,

$$G_{L+1}(\mathbf{a}_{L+1}, b_{L+1}, \mathbf{x}) := P(\mathbf{x}) \tag{3.17}$$

and randomly assigned parameters $\gamma$, $c$, but the experimental result is not good. So $\gamma$ and $c$ are determined by the linear system in P-ELM, which results in good performance in applications.

## 4 Performance evaluation

In this section, several simulations on surface reconstruction are carried out to show the performances, such as complexity, accuracy, and stability, of the proposed P-ELM and ELM.

All the performance evaluations were carried out in the Matlab 7.0 environment running on a desktop with Intel Pentium E5500 CPU with the speed of 2.8 GHz and 1.96 GB RAM.

The usual root-mean-square error (RMSE) is taken to be the measure of accuracy of a learning algorithm. That is, for $N$ sampling data $\{(\mathbf{x}_i, t_i)\}_{i=1}^{N} \subseteq D \times \mathbb{R}$ and a learned function $f$,

$$\mathrm{RMSE} = \left( \frac{1}{N} \sum_{i=1}^{N} (f(\mathbf{x}_i) - t_i)^2 \right)^{\frac{1}{2}}. \tag{4.1}$$

**Experiment 4.1** In this experiment, we will illustrate the rationality of adding a low-degree polynomial into the SLFNs. The test function in this experiment is taken to be

$$t_1(x, y) = \frac{1}{10}(-1 + 2x - 3y + 4x^2 - xy + 9y^2) \tag{4.2}$$

on the domain $[-8, 8] \times [-8, 8]$.

Choose randomly 200 points $\{((x_i, y_i), z_i)\}_{i=1}^{200}$ as training data and 200 points $\{((u_i, v_i), w_i)\}_{i=1}^{200}$ as testing data, respectively, where $(x_i, y_i)$ and $(u_i, v_i)$ are chosen randomly from the domain $[-8, 8] \times [-8, 8]$ with uniform distribution and $z_i$, $w_i$ are the corresponding observed values of the test function $t_1$ at the point $(x_i, y_i)$ and $(u_i, v_i), i = 1, 2, \ldots, 200$, respectively.

With above uniform distributed training data, we will learn the test function $t_1$ by the P-ELM and ELM. In this simulation, we will use the ELM to train the SLFN with $L$ ($L = 5$, 10, 15, 20, 30, 40) hidden nodes as

$$f_1(x, y) = \sum_{i=1}^{L} \beta_i g(a_{i1}x + a_{i2}y + b_i) \tag{4.3}$$

and use the P-ELM to train the function in $\mathcal{M}$ as

$$f_2(x, y) = \sum_{i=1}^{L} \beta_i g(a_{i1}x + a_{i2}y + b_i) + \gamma_1 x + \gamma_2 y + c, \tag{4.4}$$

where each $\beta_i, b_i, c \in \mathbb{R}, [a_{i1}, a_{i2}]^\top, [\gamma_1, \gamma_2]^\top \in \mathbb{R}^2$, and the activation function $g$ is given to be

$$g(t) = \frac{1}{1 + e^{-0.5t}}, \quad t \in \mathbb{R}. \tag{4.5}$$

Then, the training/testing RMSE of ELM and P-ELM with different numbers of hidden nodes are shown in Table 1.

As observed from Table 1, the training RMSE and testing RMSE with the P-ELM are obviously lower than those by the ELM for training the networks with the same numbers of hidden nodes. Furthermore, with the increase of hidden nodes, the training/testing RMSEs of the ELM and P-ELM both decrease. The experimental result shows that if the test function has some plane parts, the P-ELM has a better accuracy than the ELM.

**Experiment 4.2** In this experiment, we will give some comparisons of training/testing RMSE between ELM and P-ELM with different numbers of hidden nodes for a general test function. The test function in this experiment is given as follows:

**Table 1** Comparisons of training/testing RMSE by ELM and P-ELM with different numbers of hidden nodes for the test function $t_1$

| $L$ | ELM | | P-ELM | |
|---|---|---|---|---|
| | Training RMSE | Testing RMSE | Training RMSE | Testing RMSE |
| 5 | 0.1875 | 0.2101 | 0.0535 | 0.0651 |
| 10 | 0.0171 | 0.0185 | 0.0058 | 0.0072 |
| 15 | 0.0028 | 0.0031 | 0.0010 | 0.0021 |
| 20 | 0.0011 | 0.0014 | 1.3712e−4 | 2.3185e−4 |
| 30 | 4.3568e−5 | 6.2218e−5 | 1.1070e−6 | 2.1382e−6 |
| 40 | 2.5316e−6 | 2.2327e−5 | 5.2047e−7 | 7.6218e−7 |

**Table 2** Comparisons of training/testing RMSE by ELM and P-ELM with different numbers of hidden nodes for the test function $t_2$

| $L$ | ELM | | P-ELM | |
|---|---|---|---|---|
| | Training RMSE | Testing RMSE | Training RMSE | Testing RMSE |
| 50 | 0.0425 | 0.0476 | 0.0331 | 0.0358 |
| 100 | 0.0033 | 0.0038 | 0.0027 | 0.0032 |
| 150 | 3.7354e−4 | 9.9552e−4 | 2.9083e−4 | 7.3122e−4 |
| 200 | 5.4393e−5 | 3.9030e−4 | 3.9529e−5 | 3.5142e−4 |
| 250 | 8.6052e−6 | 4.8267e−5 | 7.7077e−6 | 4.5523e−5 |
| 300 | 3.9119e−6 | 4.7114e−5 | 3.5654e−6 | 3.2575e−5 |
| 500 | 2.8272e−7 | 7.9060e−6 | 1.9010e−7 | 6.3535e−6 |

$$t_2(x,y) = \begin{cases} \sin(\sqrt{x^2+y^2})/\sqrt{x^2+y^2}, & (x,y) \neq (0,0) \\ 1, & (x,y) = (0,0) \end{cases} \tag{4.6}$$

on the domain $[-8, 8] \times [-8, 8]$.

Choose randomly 1,000 points $\{((\tilde{x}_i, \tilde{y}_i), \tilde{z}_i)\}_{i=1}^{1000}$ as training data and 1,000 points $\{((\tilde{u}_i, \tilde{v}_i), \tilde{w}_i)\}_{i=1}^{1000}$ as testing data, respectively, where $(\tilde{x}_i, \tilde{y}_i)$ and $(\tilde{u}_i, \tilde{v}_i)$ are chosen randomly from the domain $[-8, 8] \times [-8, 8]$ with uniform distribution and $\tilde{z}_i, \tilde{w}_i$ are the corresponding observed values of the test function $t_2$ at the point $(\tilde{x}_i, \tilde{y}_i)$ and $(\tilde{u}_i, \tilde{v}_i), i = 1, 2, \ldots, 1000$, respectively.

With above uniform distributed training data, we will learn the test function $t_2$ in (4.6) by the P-ELM and ELM. In this experiment, we will use the ELM to train the SLFN with $L$ $(L = 50, 100, 150, 200, 250, 300, 500)$ hidden nodes as in (4.3) and the P-ELM to train the function as in (4.4) with the same activation function $g$ in (4.5). Then, the training/testing RMSEs of ELM and P-ELM with different numbers of hidden nodes are shown in Table 2, and the relations of training/testing RMSEs by ELM and P-ELM with different numbers of hidden nodes are shown in Fig. 1.

As observed from Table 2, the training/testing RMSEs by the P-ELM for the test function $t_2$ in (4.6) are still lower than those by the ELM for training the networks with the same numbers of hidden nodes. Furthermore, with the increase of hidden nodes, the training/testing RMSEs of the ELM and P-ELM become lower. The experimental result shows that even for a general test function, the P-ELM still has a better accuracy than the ELM.

As observed from Fig. 1, the training RMSE by the P-ELM tends to same when the number of hidden nodes is larger than 150, while it is for ELM when the number of hidden nodes is larger than 250. Furthermore, the testing RMSE by the P-ELM tends to same when the number of hidden nodes is larger than 200, while the testing RMSE by the ELM is still a little fluctuant. The experimental results show that for 1000 training/testing data, 200 hidden nodes are enough for the P-ELM to reconstruct the surface of the test function $t_2$.

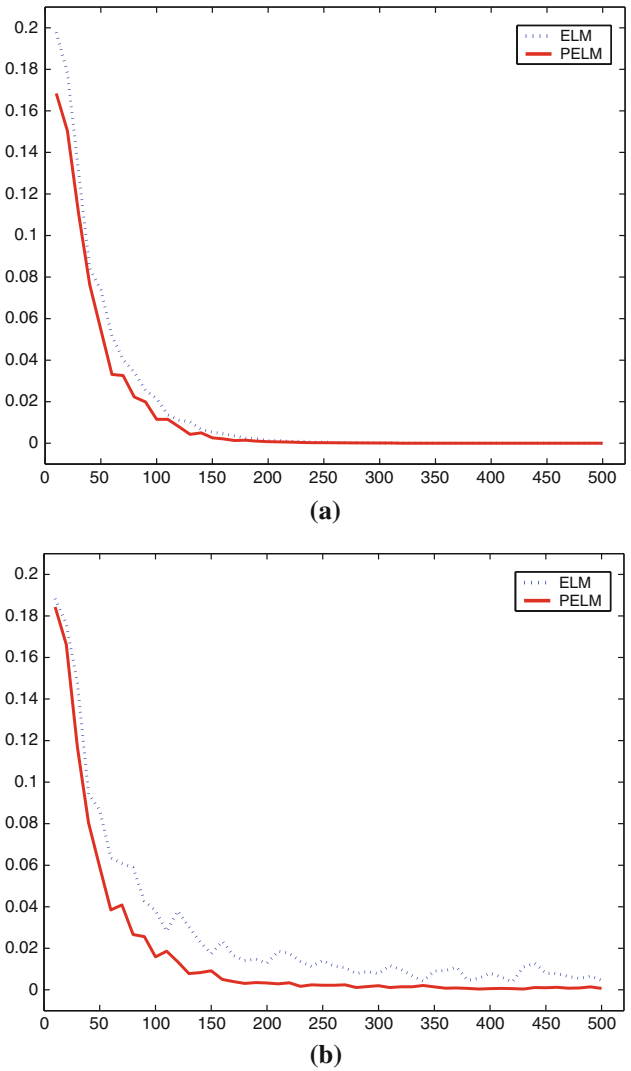



**Fig. 1** Relations of training/testing RMSE by ELM and P-ELM with the numbers of hidden nodes. **a** Relations of training RMSE by ELM and P-ELM with the numbers of hidden nodes. **b** Relations of testing RMSE by ELM and P-ELM with the numbers of hidden nodes

Now we give surface reconstruction for another test function

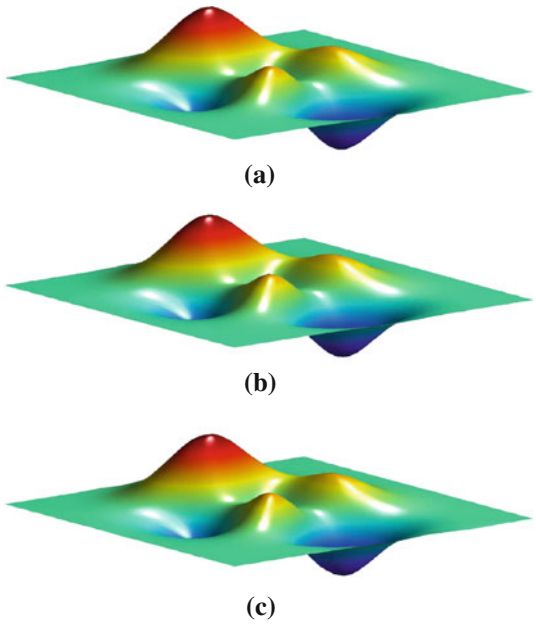



**Fig. 2** Reconstructed surfaces with ELM and P-ELM with 500 hidden nodes for the test function $t_3$. **a** Original surface. **b** Reconstructed surface with ELM. **c** Reconstructed surface with P-ELM

$$t_3(x,y) = 3(1-x)^2 e^{-x^2-(y+1)^2} - \frac{1}{3}e^{-(x+1)^2-y^2} - 10\left(\frac{x}{5} - x^3 - y^5\right)e^{-x^2+y^2} \tag{4.7}$$

on the domain $[-3, 3] \times [-3, 3]$ with ELM and P-ELM.

Choose randomly 1,000 points $\{((\tilde{x}_i, \tilde{y}_i), \tilde{z}_i)\}_{i=1}^{1000}$ as training data and 1,000 points $\{((\tilde{u}_i, \tilde{v}_i), \tilde{w}_i)\}_{i=1}^{1000}$ as testing data, respectively, where $(\tilde{x}_i, \tilde{y}_i)$ and $(\tilde{u}_i, \tilde{v}_i)$ are chosen randomly from the domain $[-3, 3] \times [-3, 3]$ with uniform distribution and $\tilde{z}_i, \tilde{w}_i$ are the corresponding observed values of the test function $t_3$ at the point $(\tilde{x}_i, \tilde{y}_i)$ and $(\tilde{u}_i, \tilde{v}_i), i = 1, 2, \ldots, 1000$, respectively.

With above uniform distributed training data, we will learn the test function $t_3$ in (4.7) by the P-ELM and ELM. We will use the ELM to train the SLFN with 500 hidden nodes as in (4.3) and the P-ELM to train the function as in (4.4) with $L = 500$ and the activation function $g$ as in (4.5). Then, their corresponding reconstructed surfaces are shown in Fig. 2.

Figure 2 shows that the reconstructed surface by P-ELM has the better smoothness than that by ELM since the P-ELM reflects the smoothness of the testing function by means of adding a polynomial in the SLFNs.

*Example 4.3* In this experiment, we will present some comparisons of complexity and accuracy among RBF, ELM, and P-ELM. We choose the same test function $t_2$ in (4.6), 1,000 training/testing data as in Example 4.2, and the same SLFNs with 1,000 hidden nodes for the ELM and P-ELM as in (4.3) and (4.4), respectively. For the RBF algorithm, we will use the following RBF network

$$f_3(x,y) = \sum_{i=1}^{1000} \beta_i h\left(\frac{1}{2}[(x-\tilde{x}_i)^2 + (y-\tilde{y}_i)^2]^{\frac{1}{2}}\right) \tag{4.8}$$

to learn the sample data, where the activation function $h$ is taken to be

$$h(t) = e^{-t^2}, \quad t \in \mathbb{R}. \tag{4.9}$$

Then, the experimental results on training/testing RMSEs and time (s) by RBF algorithm, ELM, and P-ELM with 1000 hidden nodes are shown in Table 3, and their corresponding reconstructed surfaces are shown in Fig. 3.

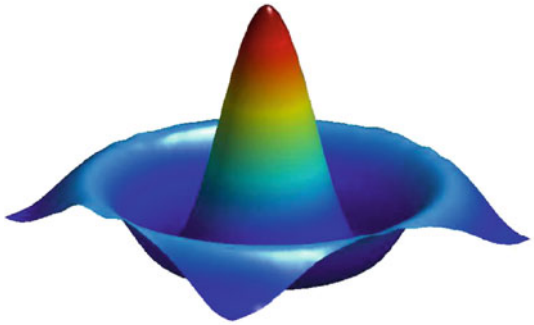
As observed from Table 3, the ELM and P-ELM have the similar training/testing time that are greatly fewer than those for the RBF algorithm. Furthermore, because the RBF algorithm is almost an interpolation for the training data, the training RMSE is lower than those of ELM and P-ELM. However, the testing RMSE of RBF algorithm is greatly larger than those of ELM and P-ELM, which implies that the ELM and P-ELM have the better generalization than that of RBF algorithm.

Figure 3 shows that the reconstructed surface by P-ELM has the best smoothness than that by RBF algorithm and ELM since the P-ELM reflects the smoothness of the testing function by means of adding a polynomial in the SLFNs.

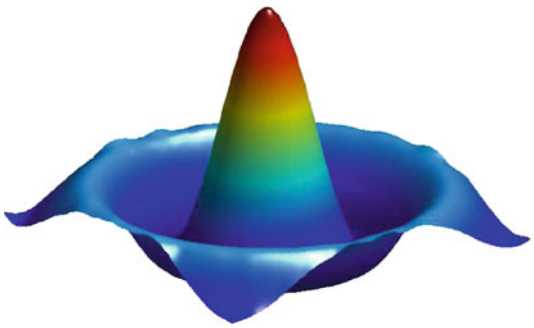
**Experiment 4.4** Since the hidden weights in the P-ELM and ELM are randomly assigned, it is necessary to consider the stability of these learning algorithms. In this experiment, the standard deviation is taken to be a tool for measuring the stability of a learning algorithm. That is, for a set of functional data $\{(\mathbf{x}_i, t_i)\}_{i=1}^{N} \subseteq D \times \mathbb{R}$ and $q$ runs with randomly assigned hidden weights, the standard deviation $\sigma$ of a learning algorithm is given by

**Table 3** Comparisons of training/testing RMSE and time (s) by ELM, P-ELM, and RBF algorithm with 1,000 hidden nodes for the test function $t_2$

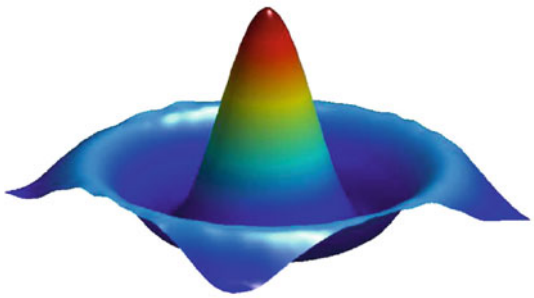
| Algorithms | Time (s) | | RMSE | |
|---|---|---|---|---|
| | Training | Testing | Training | Testing |
| RBF | 5.8131 | 0.2344 | 7.0719e−14 | 0.0022 |
| ELM | 0.8231 | 0.1032 | 2.6873e−6 | 2.5661e−5 |
| P-ELM | 0.8753 | 0.1047 | 6.5413e−8 | 2.1951e−6 |



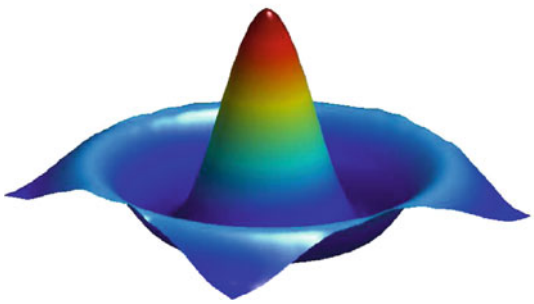
(a)



(b)



(c)



(d)

**Fig. 3** Reconstructed surfaces by RBF, ELM, and P-ELM with 1,000 hidden nodes for the test function $t_2$. **a** Original surface. **b** Reconstructed surface by RBF algorithm. **c** Reconstructed surface by ELM. **d** Surface reconstruction by P-ELM

$$\sigma = \left( \frac{1}{q} \sum_{k=1}^{q} \sum_{i=1}^{N} (\varphi_k(\mathbf{x}_i) - \bar{u}_i)^2 \right)^{\frac{1}{2}}, \tag{4.10}$$

where $\varphi_k$ is the $k$-th learned functional by the $k$th experiment and

$$\bar{u}_i = \frac{1}{q} \sum_{k=1}^{q} \varphi_k(\mathbf{x}_i), \quad i = 1, 2, \ldots, N. \tag{4.11}$$

We choose the same test function $t_2$ in (4.6), same 1,000 training/testing data in Experiment 4.2, and the same SLFNs for ELM and P-ELM in (4.3) and (4.4), respectively. Then, the comparisons of standard deviations of training/testing RMSEs with different hidden nodes are shown in Table 4, and the relations of average training/testing RMSEs with the numbers of trials by ELM and P-ELM under $L = 200$ are shown in Fig. 4.

As observed from Table 4, the standard deviations of training/testing RMSEs by the P-ELM are almost lower than those by the ELM, and the ranges of the standard deviations of training/testing RMSEs by the P-ELM are narrower than that by the ELM, which implies that the P-ELM has a better stability than the ELM.

As observed from Fig. 4, the average training/testing RMSEs by the P-ELM are almost lower than those by the ELM, and the ranges of average training/testing RMSEs by the P-ELM are smaller than those by the ELM.

**Experiment 4.5** In this experiment, we will compare the sensitivity between ELM and P-ELM. We choose the same test function $t_2$ on domain $[-8, 8] \times [-8, 8]$ in (4.6). In order to exhibit the robust of ELM and P-ELM to the noise, we chose randomly 5,000 training data and 1,000 testing data from domain $[-8, 8] \times [-8, 8]$ with uniform distribution, where the output values of training data are contaminated by noise that satisfies uniform distribution on $[-0.1, 0.1]$.

We choose the same SLFNs for ELM in (4.3) and P-ELM in (4.4) with $L = 100$, respectively. Then, Fig. 5 shows the reconstructed surfaces with ELM and P-ELM from noise data for the test function $t_2$, and Table 5 exhibits the testing/training RMSE of ELM and P-ELM with different hidden nodes.

As observed from Fig. 5 and Table 5, for the same noise data, P-ELM has lower training/testing RMSEs than ELM, which implies that P-ELM has a weaker sensitivity to noise than ELM.

## 5 Conclusions

This paper has firstly applied extreme learning machine (ELM) to reconstruct surface from some sample data. Then, it proposed another novel learning algorithm, called polyharmonic extreme learning machine (P-ELM), based on ELM to improve the accuracy, smoothness, and stability

**Table 4** Comparisons of standard deviation by ELM and P-ELM for test function $t_2$

| $L$ | Standard deviation by ELM | | Standard deviation by P-ELM | |
|---|---|---|---|---|
| | Training | Testing | Training | Testing |
| 5 | 0.1384e−3 | 0.1803e−4 | 0.2339e−3 | 0.1151e−4 |
| 10 | 0.1095e−3 | 0.0655e−4 | 0.2016e−3 | 0.1173e−4 |
| 15 | 0.2346e−3 | 0.1268e−4 | 0.2078e−3 | 0.1173e−4 |
| 20 | 0.2077e−3 | 0.0929e−4 | 0.1911e−3 | 0.1083e−4 |
| 25 | 0.2605e−3 | 0.1198e−4 | 0.2173e−3 | 0.1119e−4 |
| 30 | 0.2440e−3 | 0.1465e−4 | 0.1939e−3 | 0.1089e−4 |
| 35 | 0.2086e−3 | 0.1959e−4 | 0.2049e−3 | 0.0984e−4 |
| 40 | 0.2351e−3 | 0.1269e−4 | 0.1932e−3 | 0.1127e−4 |
| 45 | 0.3100e−3 | 0.1651e−4 | 0.1437e−3 | 0.1305e−4 |
| 50 | 0.1996e−3 | 0.1222e−4 | 0.1186e−3 | 0.0989e−4 |

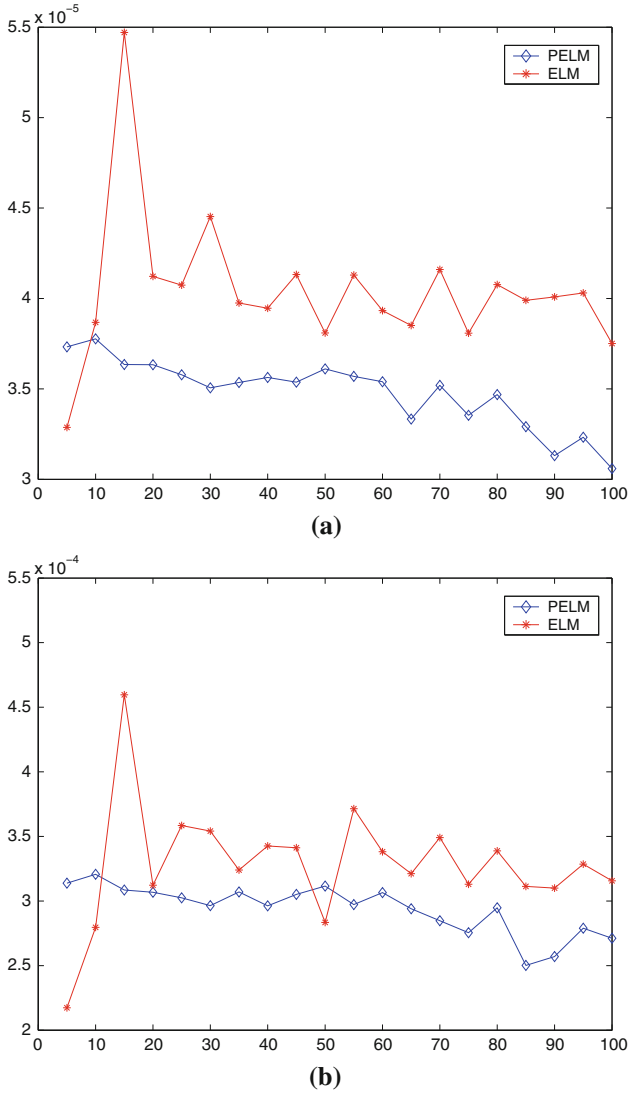



**Fig. 4** Relations of average training/testing RMSE by ELM and P-ELM with different numbers of trials. **a** Relations of the average training RMSE by ELM and P-ELM with different numbers of trials. **b** Relations of the average testing RMSE by ELM and P-ELM with different numbers of trials

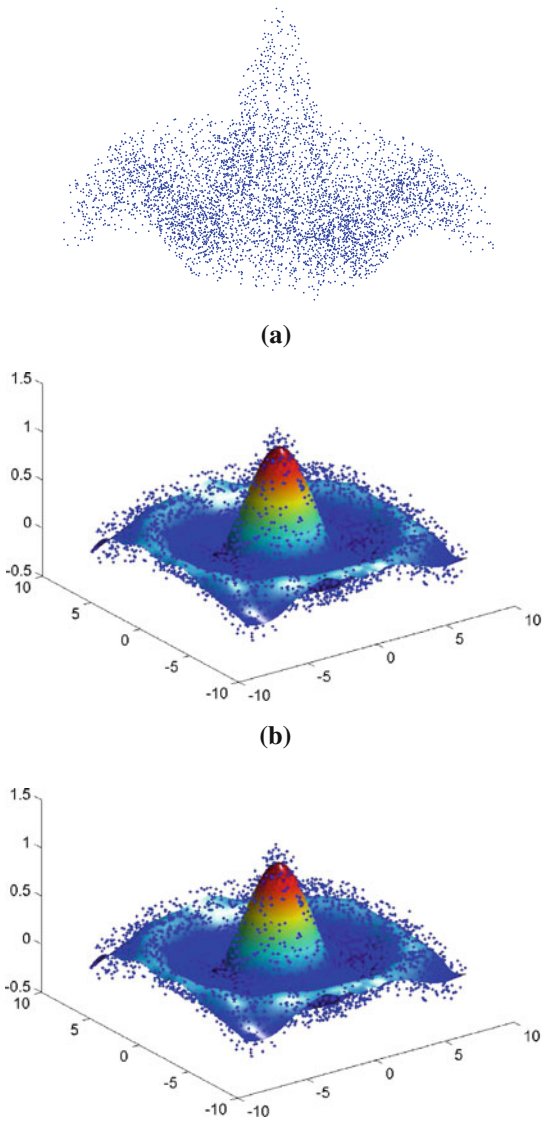



**Fig. 5** Reconstructed surfaces with ELM and P-ELM from noise data for the test function $t_2$. **a** Noise data of a surface. **b** Reconstructed surfaces with ELM from noise data for the test function $t_2$. **c** Reconstructed surfaces with P-ELM from noise data for the test function $t_2$

**Table 5** Comparisons of training/testing RMSE by P-ELM with different hidden nodes for the test function $t_2$

| $L$ | ELM | | P-ELM | |
|---|---|---|---|---|
| | Training RMSE | Testing RMSE | Training RMSE | Testing RMSE |
| 50 | 0.078 | 0.048 | 0.067 | 0.035 |
| 60 | 0.069 | 0.034 | 0.060 | 0.020 |
| 70 | 0.065 | 0.028 | 0.059 | 0.017 |
| 80 | 0.061 | 0.023 | 0.057 | 0.013 |
| 90 | 0.057 | 0.016 | 0.056 | 0.009 |
| 100 | 0.051 | 0.013 | 0.053 | 0.008 |

of the reconstructed surface. The proposed P-ELM has improved the surface reconstruction by adding a low-degree polynomial in an single-hidden-layer feedforward networks (SLF Ns). The detailed comparisons of the P-ELM, RBF algorithm, and ELM are carried out in the simulations, and the experimental results showed that the proposed P-ELM retains the advantages of ELM with an extremely high learning speed, and it has better generalization, stronger stability, and weaker sensitivity.

**Acknowledgments** The research was supported by the National Natural Science Foundation of China(No. 61101240), the Zhejiang Provincial Natural Science Foundation of China (No. Y6110117), and the Science Foundation of Zhejiang Education Office (No. Y201122002).

## References

1. Boissonnat JD (1984) Geometric structures for three-dimensional shape representation. ACM Trans Graph 3(4):266–286
2. Kolluri R, Shewchuk JR, O'Brien JF (2004) Spectral surface reconstruction from noisy point clouds. In: Proceedings of the 2004 Eurographics/ACM SIGGRAPH symposium on geometry processing, pp 11–21
3. Edelsbrunner H, Mücke EP (1994) Three-dimensional alpha shapes. In: Proceedings of the 1992 workshop on volume visualization, pp 43–72
4. Bajaj CL, Bernardini F, Xu GL (1995) Automatic reconstruction of surfaces and scalar fields from 3D scans. In: Proceedings of the 22nd annual conference on computer graphics and interactive techniques, pp 10-18
5. Bernardini F, Mittleman J, Rushmeier H, Silva C, Taubin G (1999) The ball-pivoting algorithm for surface reconstruction. IEEE Trans Visual Comput Graph 5(4):349–359
6. Amenta N, Bern M, Kamvysselis M (1998) A new Voronoi-based surface reconstruction algorithm. In: Proceedings of the 25th annual conference on computer graphics and interactive techniques, pp 415–421
7. Walder C, Schölkopf B, Chapelle O (2006) Implicit surface modelling with a globally regularised basis of compact support. Comput Graph Forum 25(3):635–644
8. Yoon M, Lee YJ, Lee S, Ivrissimtzis I, Seidel HP (2007) Surface and normal ensembles for surface reconstruction. CAD 39(5):408–420
9. Pan RJ, Meng XX, Whangbo TK (2009) Hermite variational implicit surface reconstruction. Sci China Ser F 52(2):308–315
10. Cybenko G (1989) Approximation by superposition of sigmoidal functions. Math Control Signal Syst 2(4):303–314
11. Funahashi KI (1989) On the approximate realization of continuous mappings by neural networks. Neural Netw 2:183–192
12. Hornik K (1993) Some new results on neural network approximation. Neural Netw 6:1069–1072
13. Chen TP, Chen H, Liu RW (1995) Approximation capability in by multiplayer feedforward networks and related problems. IEEE Trans Neural Netw 6:25–30
14. Chen TP, Chen H (1995) Universal approximation to nonlinear operators by neural networks with arbitrary activation functions and its application to dynamical systems. IEEE Trans Neural Netw 6:911–917
15. Cao FL, Xie TF, Xu ZB (2008) The estimate of approximation error for neural networks: a constructive approach. Neurocomputing 71:626–630
16. Cao FL, Zhang YQ, He ZR (2009) Interpolation and rate of convergence for a class of neural networks. Appl Math Model 33(3):1441–1456
17. Muraki S (1991) Volumetric shape description of range data using "blobby model". Comput Graph 25:227–235
18. Floater MS, Iske A (1996) Multistep scattered data interpolation using compactly supported radial basis functions. J Comput Appl Math 73(1):65–78
19. Carr J, Beatson R, Cherrie H, Mitchel T, Fright W, Mccallum B, Evans T (2001) Reconstruction and representation of 3D objects with radial basis functions. SIGGRAPH, pp 67–76
20. Turk G, O'Brien J (2002) Modelling with implicit surfaces that interpolate. ACM Trans Graphics 21(4):855–873
21. Medeiros AD, Doria AD, Dantas JDM, Goncalves LMG (2008) An adaptive learning approach for 3-D surface reconstruction from point clouds. IEEE Trans Neural Netw 19(6):1130–1140
22. Yoon M, Ivrissimtzis I, Lee S (2008) Self-organising maps for implicit surface reconstruction. EG UK Theory and Practice of Computer Graphics
23. Rego RLME, Araujo, Neto FBDL (2010) Growing self-reconstruction maps. IEEE Trans Neural Netw 21(2):211–223
24. Huang GB, Zhu QY, Siew CK (2004) Extreme learning machine: a new learning scheme of feedforward neural networks. In: Proceedings of international joint conference on neural networks (IJCNN2004), Budapest, Hungary, vol 2(25–29):985–990
25. Rao CR, Mitra SK (1971) Generalized inverse of matrices and its applications. Wiley, New York
26. Huang GB, Zhu QY, Siew CK (2006) Extreme learning machine: theory and applications. Neurocomputing 70:489–501
27. Huang GB, Zhu QY, Siew CK (2006) Universal approximation using incremental constructive feedforward networks with random hidden nodes. IEEE Trans Neural Netw 17(4):879–892
28. Ortega JM (1987) Matrix theory. Plenum Press, New York
29. Feng GR, Huang GB, Lin QP, Gay R (2009) Error minimized extreme learning machine with growth of hidden nodes and incremental learning. IEEE Trans Neural Netw 20(8):1352–1357
30. Duchon J (1977) Splines minimizing rotation-invariant seminorms in Sobolev spaces. In: Schempp W, Zeller K (eds) Constructive theory of functions of several variables, No. 571 in Lecture Notes in Mathematics. Springer, Berlin, pp 85–100