Hindawi
Mathematical Problems in Engineering
Volume 2019, Article ID 2781437, 21 pages
https://doi.org/10.1155/2019/2781437

*Research Article*

# Prediction of Ship Cabin Noise Based on RBF Neural Network

**Jun Guo**,[1] **Mei-ting Wang**,[1] **You-wei Kang**,[2] **Yin Zhang**,[1] **and Chen-xu Gu**[1]

[1]*Harbin Engineering University, No. 145, Nantong Street, Nangang District, Harbin 150001, China*
[2]*CIMC Offshore Co. LTD, CIMC R&D Center, No. 2, Gangwan Avenue, Shekou Industrial Park, Nanshan District, Shenzhen 518067, China*

Correspondence should be addressed to Jun Guo; guo_jun@hrbeu.edu.cn

Received 18 November 2018; Revised 13 March 2019; Accepted 25 March 2019; Published 14 April 2019

Academic Editor: Roberto G. Citarella



Prediction of cabin noise for new types of ships and offshore platforms, based on measurement or simulation databases, is a common problem that needs a solution at the beginning of the design process. In this paper, we explore the use of a radial basis function (RBF) neural network to study this problem. Within the framework of the RBF network, we implement and compare several algorithms to devise a fast and precise cabin noise prediction model. We select a combination of algorithms after training the RBF with noise measurement samples. The results show that the RBF neural network trained using the DE algorithm has better prediction accuracy, generalization, and robustness than the others. Our work provides a new method for preliminary noise assessment during the schematic design phase and enables rapid analysis of vibration and noise control schemes for ships and offshore platforms.

## 1. Introduction

Increasing interest in environmental protection means that people pay more attention to the effects of vibration and noise from offshore platforms and ships on crew happiness, working environment, and physical health. Delaying the implementation of noise reduction measures needed to meet cabin noise requirements until after construction adds significant cost and affects both interior arrangements and weight [1]. Accurate prediction of cabin noise during the design or early construction phases is an urgent need.

For elaborate structures, such as ships and drilling platforms, the classical approach that requires defining and solving differential equations for each component is too complex. Therefore, most existing noise prediction is done using approximate methods to study sound radiating from dynamic systems. Multiple commercial acoustics software products, using different algorithms and frequency domains, have been available, including VA One and LMS Virtual.Lab Acoustics. The main solution methods of these software products involve Finite Element Method (FEM), Boundary Element Method (BEM), Statistical Energy Method (SEM), and so forth. In the field of ship industry and ocean engineering, Statistical Energy Method can be utilized in the process of compartment noise analysis of high-speed passenger ships, which helps detect the noise transmission route and find the solution for noise reduction. Boundary Element Method might be used for the analysis of normal ventilation systems of the ship, which helps explore the noise frequency characteristics of pipeline and the changing rules within different flow velocity of the pipe. Statistical Energy Method can also be used in the noise influence factor prediction of huge semisubmersible ocean platform, and the noise control methods that meet the requirements of specification are proposed [2, 3]. As for the civilian automobile and aerospace industry, the commercial software and solution methods used are basically the same, but the frequency range and research object might be different. Currently, various calculation methods are used to predict the low, intermediate, and high frequency, respectively, additionally improving the accuracy of the forecast [4]. However, as in the early stage of ship design and ocean engineering, the detailed design drawing has not been mapped and the required differential equation cannot be acquired. Therefore, the purpose of this paper is to find out a new method for the rapid and accurate prediction of ship compartment noise in the initial design stage of the project, under the background of the absence of detailed calculation parameters, that is, to introduce the artificial intelligence or, specifically, artificial nerves network technology into the field of compartment noise control.

## 2. The Working Mechanism of Radial Basis Function Neural Network and Algorithm

Artificial intelligence (AI) is a field devoted to the development of computer systems to simulate and extend human intelligence. Ideally, the goal is the creation of a machine that reacts in a way similar to human intelligence [5]. The study of neural networks is a branch within the field that models the neural network of the human brain using mathematical methods aimed at imitating the brain's function and structure [6].

*2.1. Radial Basis Function.* The greatest advantage of artificial neural networks is their ability to approximate the output of a human brain without specific knowledge of its operation. As long as sufficient training samples are provided, the neural network can provide this output with arbitrary precision [7, 8]. Due to the complex structure and large number of vibrating machines on a ship, vibration and noise have a nonlinear relationship [9, 10]. The self-learning and self-adapting characteristics of a neural network can model the nonlinear system and predict the overall vibration and noise throughout the system. There is no need for geometric modeling and complex boundary and parameter settings. Once the neural network modeling the system is trained, it has all it needs to forecast the noise, which greatly shortens the time needed to make the prediction.

In 1988, Moody and Darken proposed the radial basis function (RBF) neural network for the precise interpolation of functions. The RBF network can approximate any nonlinear function, deal with the intractable regularity in the system, and converge quickly to a solution. It has been successfully applied in many fields, including pattern recognition, nonlinear function approximation, time series analysis, data classification, image processing, and control and fault diagnosis [11–13]. We use the RBF network in our approach to ship noise prediction.

The RBF network is known as a feedforward neural network with three layers of neurons, as shown in Figure 1. The first layer is the input layer, with the number of nodes equal to the dimension of the input vector. The second layer is the hidden layer, with the number of nodes depending on the complexity of the problem and being larger than the number of input layer nodes. The third layer is the output layer, with the number of nodes equal to the dimension of the output vector. The weight parameter $\omega$ represents the link between nodes. The weight vector exists only between the hidden and output layers. The arrow in the figure indicates the direction in which information propagates through the network. The activation function of the RBF network can be selected in various forms as shown in Table 1, and its curve shape is shown in Figure 2.

The three functions in Table 1 are all radially symmetric. The function value decreases when the independent variable deviates from the center, where $\sigma_j$ is the spread or width. The smaller $\sigma_j$ is, the smaller the curve width is and the faster the function falls, and the function is more selective. Because the width of the Gaussian function is the smallest, we generally choose the Gaussian function as the activation function of the

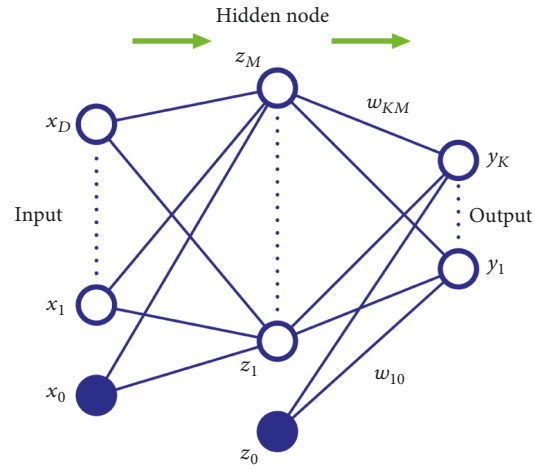


Figure 1: RBF network structure.

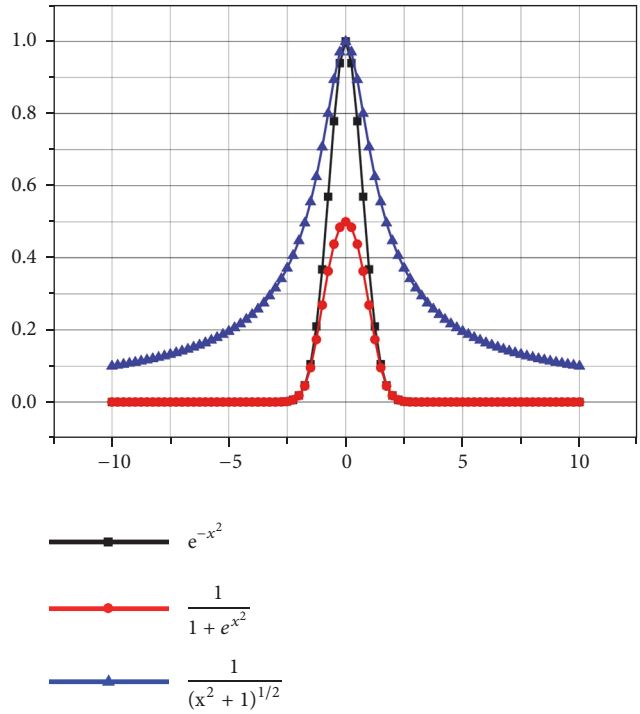


Figure 2: Comparison of three activation functions.

RBF network. As can be seen from Figure 2 , the output of a single hidden node can be expressed as

$$Z_j = e^{-\|x_i - c_j\|/2\sigma_j^2} \tag{1}$$

where $x_j (i = 1, 2, \ldots, N)$ is the ith sample, N is the number of samples, $\sigma_j = 1$ is the width of the center $c_j$ of the jth hidden node, and the dimension of $x_j c_j$ is D. Without affecting universality, we let $\sigma_j = 1$; it is easy to prove that its output can represent infinite multidimension:

Table 1: RBF typical activation function.

| Activation function | Expression |
|---|---|
| Gauss function | $\emptyset_j(x) = e^{-x^2/2\sigma_j^2}$ |
| Reflected Sigmoid function | $\emptyset_j(x) = \frac{1}{1+e^{x^2/\sigma_j^2}}$ |
| Inverse Multiquadric function | $\emptyset_j(x) = \frac{1}{(x^2+\sigma_j^2)^a}$ |

$$
\begin{aligned}
e^{-\|x-c\|^2/2} &= e^{-\|x\|^2/2} e^{-\|c\|^2/2} e^{x^T c} = \sum_{j=0}^{\infty} \frac{\left(X^T c\right)^j}{j!} \\
&\cdot e^{-\|x\|^2/2} e^{-\|c\|^2/2} = \sum_{j=0}^{\infty} \left( \frac{e^{-\|x\|^2/2}}{\sqrt{j!}} \frac{e^{-\|c\|^2/2}}{\sqrt{j!}} X^T c \right)^j \\
&= \sum_{j=0}^{\infty} \sum_{\sum_i n_i = j} \frac{e^{-\|x\|^2/2j}}{\sqrt{j!}^{1/j}} \binom{j}{n_1,,, n_n}^{1/2} x_1^{n_1}, \cdots, x_D^{n_D} \\
&\cdot \frac{e^{-\|c\|^2/2j}}{\sqrt{j!}^{1/j}} \binom{j}{n_1,,, n_n}^{1/2} c_1^{n_1}, \cdots, c_D^{n_D}
\end{aligned}
\tag{2}
$$

It is not difficult to get the feature mapping function from the above proof:

$$
\phi(x) = \left( \frac{e^{-\|x\|^2/2j}}{\sqrt{j!}^{1/j}} \binom{j}{n_1, \ldots, n_n}^{1/2} \cdot x_1^{n_1}, \ldots, x_D^{n_D} \right)_{j=0,\cdots,\infty,\sum_{i=1}^{k} n_i = j}. \tag{3}
$$

There are infinite terms on the right side of the equation, so the feature space corresponding to the radial basis function is infinite. Thus, the kth output can be expressed as

$$
y_k = \sum_{j=1}^{M} w_{kj} \phi\left( \left\| x - c_j \right\| \right). \tag{4}
$$

As can be seen from (2)-(4), each hidden layer neuron responds to input x, and the output of the RBF network is the weighted sum of these responses. If the input x is close to the jth hidden node center $c_j$, the response is large, which is equivalent to the implicit node being activated. If the distance is far away, the response is almost zero, which is equivalent to the implicit node being suppressed. This feature is known as the "local mapping" feature, which makes the RBF network converge faster than the "global mapped" network.

In order to design an RBF network for noise prediction, we must determine the number and coordinates of the hidden nodes, the expansion constants of the activation functions of the hidden nodes, and the weights of the output nodes. We will combine a variety of algorithms to train the network to achieve the quality required.

*2.2. Gradient Descent Algorithm.* The calculation process of the gradient descent method is to solve the minimum value of the error function along the direction of the gradient descent. Nowadays, the numerical iterative method is generally adopted. The specific formula is as follows:

$$
W_j^{(\tau+1)} = W_j^{\tau} - \eta_w \Delta E\left(W_j^{(\tau)}\right) \tag{5}
$$

$$
c_j^{(\tau+1)} = c_j^{\tau} - \eta_c \Delta E\left(c_j^{\tau}\right) \tag{6}
$$

$$
\sigma_j^{(\tau+1)} = \sigma_j^{\tau} - \eta_\sigma \Delta E\left(c_j^{\tau}\right) \tag{7}
$$

In the formula, parameters $\eta_\omega$, $\eta_c$, and $\eta_\sigma$ are the learning rates of the corresponding variables; $c_j$ is the central coordinate of the hidden nodes; $\sigma_j$ is the expansion constant; $W_j$ is the network weight.

The learning rate should be set smaller (increasing accuracy) when the gradient falls faster and larger when the gradient is flat (increasing the convergence speed). Weights, center coordinates, and width are typically initialized to random numbers.

*2.3. Particle Swarm Optimization Algorithm.* The particle swarm algorithm treats the potential solution of the optimization problem as a particle in the search space and searches for the optimal solution of the current space through the particle.

For the first time in a D search space, there are m particles forming a group, where the position of the ith particle is represented by a vector $x_i = (x_{i1}, x_{i2}, \ldots, x_{iD}), i = 1, 2, \cdots m$, and the velocity of the particle is represented by a vector $v_i = (v_{i1}, v_{i2}, \cdots, v_{iD})$. The optimal position searched by the ith particle is $p_i = (p_{i1}, p_{i2}, \cdots, p_{iD})$, and the optimal position searched by the entire particle swarm is $p_g = (p_{g1}, p_{g2}, \cdots p_{gD})$. The particle update formula is as follows:

$$
\begin{aligned}
v_{id}(t+1) &= v_{id}(t) + c_1 r_1 \left[ p_{id} - x_{id}(t) \right] \\
&\quad + c_2 r_2 \left[ p_{gd} - x_{gd}(t) \right]
\end{aligned}
\tag{8}
$$

$$
x_{id}(t+1) = x_{id}(t) + v_{id}(t+1) \tag{9}
$$

When $v_{id} \geq V_{max}$, take $v_{id} = V_{max}$; when $v_{id} < V_{max}$, take $v_{id} = -V_{max}$.

i = 1, 2, ..., m; d = 1, 2, ..., D; nonnegative constants $c_1$ and $c_2$ are called acceleration constants; $r_1$ and $r_2$ are uniformly distributed over the interval [0, 1] random number. $x_{id}(t)$ is the current position of the ith particle, $p_{id}$ is the optimal position searched for the ith particle, $p_{gd}$ is the optimal

position for the entire particle swarm search, $v_{id}$ is the current velocity of the ith particle, and the negative number $v_{max}$ is the maximum speed limit.

The particle velocity update is divided into three parts: the first part is the influence of the current speed, the current state of the particle is connected, in addition to the ability to balance the global and local search; the second part is the influence of the particle's own cognition, that is, the influence of the particle's own memory. Make the particles have global search ability and avoid falling into local minimum values; the third part is the influence of group information, and realize information sharing and cooperation between particles.

*2.4. Differential Evolution Algorithm.* The differential evolution algorithm generates a difference vector by randomly selecting two individuals from the parent population and weights it and adds it to another random individual in the parent class. Then, the generated new individual and the corresponding individual in the parent are performed. The crossover operation generates a new individual and the individual with good fitness is the child individual.

The initial population is generated first, and the specific formula is as follows:

$$\{x_i(0) \mid x_i(0) = [x_{i1}, x_{i2}, x_{i3}, \cdots, x_{iD}],\ i = 1, 2, \cdots, NP\} \tag{10}$$

$$x_{ij} = X_{min} + \text{rand} \times (X_{max} - X_{min}) \tag{11}$$

where $x_i(0)$ represents the ith individual in the initial population, NP represents the population size, $j = 1, 2, \cdots, D$, D represents the dimension of the solution to the problem, and rand is the uniformly distributed random number in the interval $[0, 1]$.

Next, the differential variability is as follows:

$$v_i(g) = x_{r1}(g) + F \cdot [x_{r2}(g) - x_{r3}(g)] \tag{12}$$

$r_1 \neq r_2 \neq r_3$, where g is the gth generation and F is the scaling factor.

Then there is the crossover operation; the specific formula is as follows:

$$u_{ij}(g) = \begin{cases} v_{ij}(g), & \text{rand}(0,1) \leq CR \text{ or } j = j_{\text{rand}} \\ x_{ij}(g), & otherwise \end{cases} \tag{13}$$

$j = 1, 2, \cdots, D$; $j_{\text{rand}}$ is a randomly selected integer within $[1, D]$, and CR is the crossover rate.

Finally, by comparing the parents and performing the selection operation, the specific formula is as follows:

$$x_i(g+1) = \begin{cases} u_i(g), & fitness[u_i(g)] < fitness[x_i(g)] \\ x_i(g), & otherwise \end{cases} \tag{14}$$

$$i = 1, 2, \cdots, NP$$

## 3. Establishment of Ship Cabin Noise Database

Although the intelligent forecasting method does not need to provide a precise formula for calculating vibration noise, it is necessary to identify the independent variable that influences the dependent variable. This process is called feature extraction. Cabin noise samples are used for intelligent forecasting, and the quality of feature set extraction results is a key to accurate noise prediction. In this section, we explore the extraction of the main features affecting the ship cabin noise in detail and express them mathematically to establish a corresponding ship cabin noise database.

### 3.1. Input Parameter

*(1) Noise Source.* The ship belongs to a large offshore structure, and its internal contains a large number of sources of vibration and noise, including the main engine, motor unit, propeller, gear box, pump, and fan. For various excitation sources, the acceleration level can be determined by empirical formulas [14].

The vibration excitation generated by the propeller can be loaded onto the ship's floor directly above the propeller. The specific formula is as follows:

$$L_a = 10\lg(MN) + 40\lg D + 30\lg n_e + 10 \tag{15}$$

M is the number of propellers; N is the number of propeller blades; D is propeller diameter; $n_e$ is rated speed of propeller.

The empirical estimation formula for the diesel engine and motor acoustic radiation power level $L_w$ (reference sound power level $w_0 = 10^{-12}$W) is as follows:

$$L_w = 10\lg P_e + 57 + C_w \tag{16}$$

$p_e$ is rated power of diesel engines and motors; $c_w$ is octave correction of diesel and motor air noise.

The engine foot acceleration (reference acceleration is $a_0 = 1\mu m/s^2$) is determined by the formula

$$L_a = -20\lg m + 20\lg P_e + 30\lg\frac{n}{n_e} + 120 + C_a \tag{17}$$

m is the quality of the diesel engine; $p_e$ is rated power of diesel engine; $n_e$ is rated speed of diesel engine; n is working speed of diesel engine; $c_a$ is octave correction value of diesel engine vibration.

The motor foot acceleration (reference acceleration is $a_0 = 1\mu m/s^2$) is determined by the formula

$$L_a = 10\lg P_e + 7\lg n_e + 42 + C_a \tag{18}$$

$p_e$ is rated power of the motor; $n_e$ is rated speed of the motor; $C_a$ is the octave correction value of the motor vibration.

According to the above formula, the sound radiation power level, and the foot acceleration level of the existing equipment under the rated power, the excitation condition of the diesel engine and the motor at other powers can be roughly estimated by modifying the rated power value.

*(2) Noise Propagation Path.* On-board noise is transmitted primarily through the air and the hull structure. Because it

Table 2: Database input parameters.

| Serial number | Parameter name | Serial number | Parameter name |
|---|---|---|---|
| 1 | Number of decks separated from the previous host | 9 | Number of decks separated from the propeller |
| 2 | Number of transverse bulkheads separated from the front mainframe | 10 | Number of transverse bulkheads separated from the propeller |
| 3 | Number of decks separated from the front motor | 11 | Target cabin surface area |
| 4 | Number of transverse bulkheads separated from the front motor | 12 | Target compartment plate average thickness |
| 5 | Number of decks separated from the rear host | 13 | Front host power |
| 6 | Number of transverse bulkheads separated from the rear host | 14 | Rear host power |
| 7 | Number of decks separated from the rear motor | 15 | Front generator set power |
| 8 | Number of transverse bulkheads separated from the rear motor | 16 | Rear generator set power |

is typically unnecessary to predict cabin noise near the cabin (i.e., in the air), we ignore air noise and address only the hull structure as a sound source parameter. Structural noise is attenuated by the structure of the cabin, deck, transverse bulkheads, and so on during propagation. We quantify these influencing factors as the average shelf thickness and surface area of the cabin, the number of decks between the cabin and the noise source, and the number of transverse bulkheads between the cabin and the noise source. We store these factors for multiple points in the ship and use them as input parameters as shown in Table 2.

*3.2. Output Measurements.* For the output, we adopt the A-weighted sound pressure level (SPL) in dB(A). This measurement is commonly used as the noise evaluation index in current engineering. The A-weighted SPL is close to the human ear's hearing characteristics: sensitive to high frequency noise and insensitive to low frequency noise. Its logarithmic representation is directly measurable with a sound level meter, which simplifies the work and reduces the measurement range.

*3.3. Acquisition of Ship Cabin Noise Database Data.* All data in our database is based on a ship with characteristics as given in Table 3.

Table 3: Ship dimensions and power used by our data.

| | |
|---|---|
| Length overall | 55.00 m |
| Molded depth | 4.20 m |
| Molded breadth | 9.50 m |
| Draft | 2.20 m |
| Displacement | 381.00 t |
| Main engine power | 3×2528 kW |
| Cruising speed | 18.00 kn |

We selected 10 cabins within the ship as training and test samples as shown in Figure 4. These cabins had no noise reduction measures installed. We selected the front and rear host groups, front and rear motor groups, and propellers as noise sources, also called excitation sources, positioned as shown in Figure 5. We treat the two rear main engines as a unit, operating synchronously and driving the propeller consistent with the rear host's output. We selected 45 sets of combined working conditions under different power levels as excitation conditions and expressed the power parameters of the excitation source as percentages between 0 and 100. Based on the different working conditions of the ship, we used the software product VA One to simulate noise levels in the 10 cabins; the specific model SEA subsystem connection diagram is shown in Figure 3. We selected subsystems with a height of 1.5 m~2 m from the deck as research objects and obtained the A-level noise for each cabin under the 45 working conditions.

Because the statistical energy analysis method has a more accurate solution to the medium-high frequency dynamic response problem in complex structural systems, the ship belongs to a relatively complicated and large structural system. Generally, it is considered to belong to the low-frequency region when the frequency is less than 50 Hz, and the intermediate frequency is 50 to 200 Hz. In the area, above 200Hz can be regarded as high-frequency area, while the hearing range of normal human ear is within 20~20kHz. It can be seen that the proportion of low-frequency area is small, so it is inevitable to solve the problem of acoustic prediction of ship cabin and system dynamics problems in the mid-high frequency region. In addition, since the human

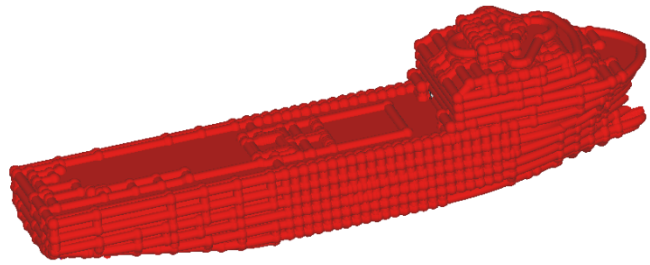

Figure 3: SEA subsystem connection diagram.

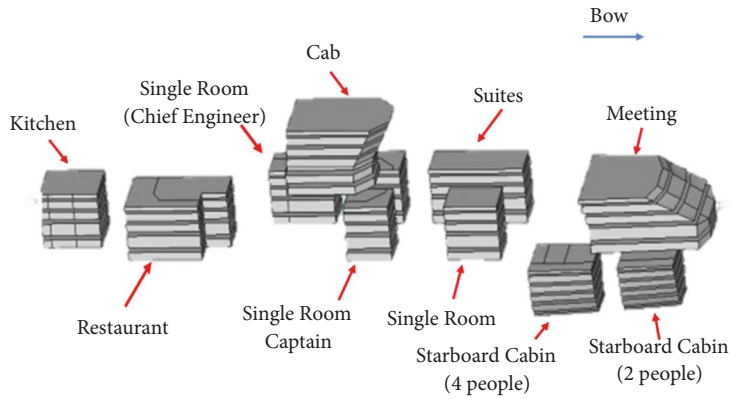


Figure 4: Schematic diagram of the ship cabin layout.

ear is not sensitive to high-frequency noise to low-frequency noise, the proportion of low-frequency noise is weakened in the A sound level. Therefore, it is more suitable to use the statistical energy analysis method to solve the ship cabin noise prediction.

In summary, the 10 cabins and 45 test samples produced 450 sample pairs. The input vector consisted of 16 dimensions, the excitation source of 4 dimensions, and the cabin structure parameters of 12 dimensions.

## 4. RBF Network Design and Noise Prediction Based on Gradient Descent Algorithm

*4.1. Network Process Design.* The training process for the gradient descent algorithm in the RBF network is relatively simple, but there are many controlling parameters. The network training process is shown in Figure 6. Because the root mean square (RMS) error well describes the accuracy of the cabin noise prediction for each training iteration, we use the RMS error as the iteration test. The RMS error formula is

$$\mathrm{RMSE} = \sqrt{\frac{\sum_{i=1}^{n} \left(X_{obs,i} - X_{model,i}\right)^2}{n}}. \tag{19}$$

RMSE denotes root mean square error; $X_{obs,i}$ is network forecast; $X_{model,i}$ is simulation calculated value; n is number of trainings.

In order to facilitate the initialization of the hidden node center coordinates and to avoid computational saturation, we normalize the input data so that the parameters of all dimensions are in the [0, 1] interval.

*4.2. Selection of Parameters and Their Influence on the Algorithm.* Several parameters influence the output quality and learning rate but there are diminishing returns beyond a near-optimal setting. We now present our empirical investigation of these parameters.

*(1) The Influence of the Number of Hidden Nodes.* We start the process with 17 hidden nodes, increasing the number of nodes by 1 each time the prediction test for the four cabins is completed, up to a maximum of 62. We calculate the average relative error of noise prediction for all working conditions in each of the four compartments for each test and obtain the correlation between the number of hidden nodes and the forecast result of the algorithm, as shown in Figure 7. During the experiment, other parameters remain unchanged. The maximum number of iterations is set to 25000, and the target error $E_0$ is set to 5. The hidden node center coordinates, the extended constants of each hidden node, and the output weights are initialized to random numbers in the interval [0, 1]. The learning rate $\lambda$ is set to $10^{-6}$ in all cases.

The figure shows that there is no correlation between the number of hidden nodes and the algorithm prediction results and that the average relative error of the four target cabin predictions fluctuates around a value. As the iterations continue, the extra hidden nodes appear to offset, attenuate,

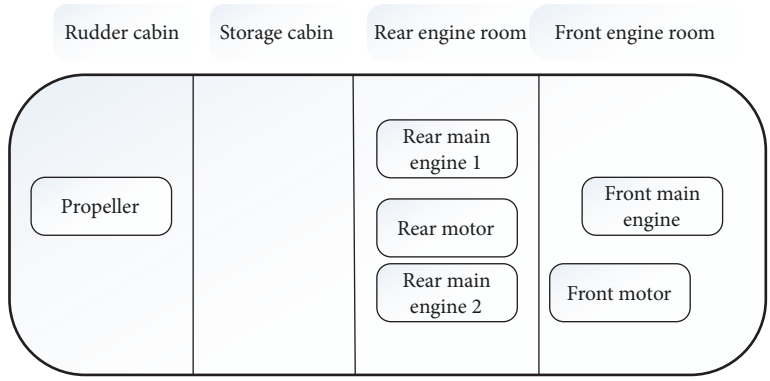


Figure 5: Diagram of noise source distribution.

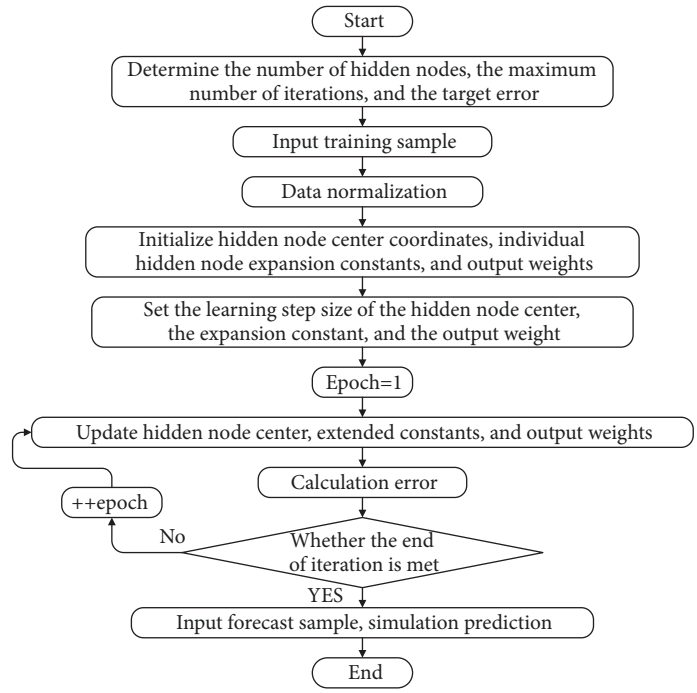


Figure 6: RBF network training flow chart based on a gradient descent algorithm.

shrink, and merge. The curve with relatively few hidden nodes is smoother overall. Therefore, from the perspective of network generalization, we use 40 hidden nodes.

*(2) The Effect of Learning Rate*. We set the learning rate of the center coordinates of the hidden nodes, the expansion constants of the hidden nodes, and output weights to be all the same value, $\lambda$, with values of $10^{-3}$, $10^{-4}$, $10^{-5}$, and $10^{-6}$ in turn. After performing 20 training passes on the cabin data from the upper two decks in the database, we calculate the root mean square error after each iteration and average them. The results are shown in Figure 8. We left other parameters unchanged, with 40 hidden nodes, 25000 maximum iterations, the target error $E_0$ of 5, the hidden node center coordinates, extended constants of each hidden node, and output weights all initialized to a random number in the interval $[0, 1]$ [15].

The figure shows that the larger the learning rate $\lambda$ is, the faster the root mean square error decreases. However, when $\lambda = 10^{-4}$, the root mean square error fluctuates around the minimum value after iterating a certain number of steps and does not continue to fall. This is because a larger learning rate results in a greater parameter update at each iteration, which benefits the global optimization at the beginning, and quickly approaches the global minimum. However, near the global minimum, each movement is too large, crossing the minimum value each time and leading to the oscillation around it.

However, a smaller learning rate does not always mean that it is easier to approximate the error to the global minimum. As shown in the figure, with 25000 iterations, the RMS error for $\lambda = 10^{-6}$ is still greater than the RMS error for $\lambda = 10^{-5}$. This situation occurs for two reasons. First, the smaller $\lambda$ is, the slower the error moves, and more iterations

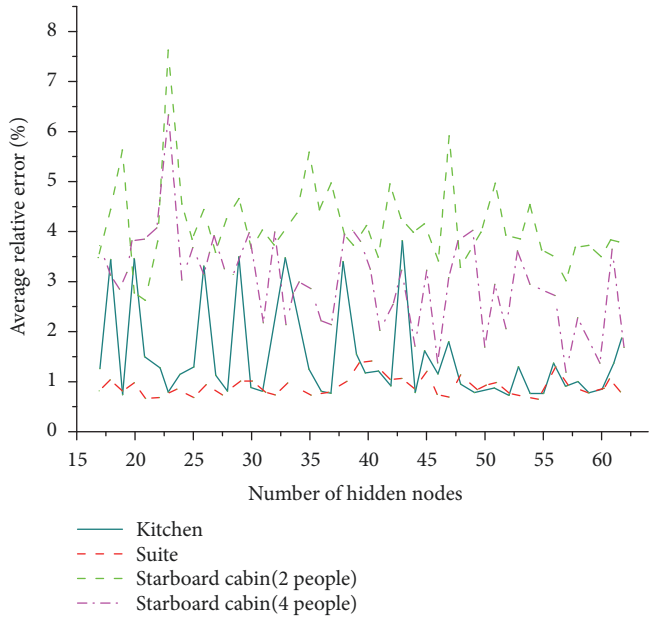


Figure 7: The relationship between the number of hidden nodes and the forecast error.

are needed to fall to the minimum value. Second, the smaller $\lambda$ is, the easier it is to fall into the local minimum and stay there, which is an obvious disadvantage of the gradient descent algorithm.

In summary, we set the learning rate at around $\lambda = 10^{-5}$.

*(3) Effect of Target Error $E_0$.* Although the purpose of the iteration is to reduce the RMS error, the inevitable noise in the training samples coupled with a too-small target error causes the noise data to be accurately fitted, resulting in overfitting [16, 17]. Therefore, we set $E_0$ to an integer between 6 and 10 and leave the other parameters of the network unchanged. Carrying out the prediction method with cross-validation 20 times yields the results shown in Figure 9.

The figure shows that when the target error $E_0$ is 6 and 7, the average relative error of the actual forecast is even larger, and it is obvious that the noisy data is also accurately interpolated. When $E_0$ is 8 or 9, the actual forecast average relative error is the smallest. After comparing the data, we find that the relative error of each working condition is more stable when $E_0$ is taken as 8, and the maximum relative error is smaller than when $E_0$ is 9.

*4.3. Forecast Error and Evaluation.* We evaluate the forecast error by setting the number of hidden nodes to 40, the maximum number of iterations to 25,000, the target error $E_0$ to 8, and the learning rate $\lambda$ to $10^{-5}$ and the hidden node center coordinates, the extended constants of the hidden nodes, and the output weights to random numbers in $[0, 1]$. We make 10 predictions using a cross-validation method and show the average of the forecast results in Figures 10 and 11 and Table 4.

The results show that the RBF network trained by the gradient descent method has a high prediction accuracy, a small prediction error for cabin noise on different decks, with different surface areas and with different numbers of transverse bulkheads from the excitation source, and stable error characteristics.

Although the RBF network trained by the gradient descent algorithm has a good performance on cabin noise prediction, it has equally notable shortcomings: more controlling parameters and sensitivity to those parameters. The target error $E_0$, the learning rate $\lambda$, the initialization of hidden node center coordinates, individual hidden node expansion constants, and output weights all exert great influence on the performance of the algorithm. More relevant experience is needed to ensure selection of appropriate parameters in the actual application of our method to a project in order to avoid falling into the local minimum.

## 5. RBF Network Design and Noise Prediction Using Particle Swarm Optimization

*5.1. Network Process Design.* In this section, we present our use of the particle swarm optimization algorithm to train the hidden node center coordinates, expansion constants, and output weights of the RBF network. With $M$ hidden nodes, the dimension of each particle $pop(i)(i = 1, 2, \ldots m)$ is $16M+M+M=18M$. The first $16M$ dimension represents the coordinates of the center of $M$ hidden nodes, the $M$ dimension represents the expansion constant of each hidden node, and the last M dimension represents the output weight; namely,

$$pop\,(i) = \left[c_{i1}^{1}, c_{i2}^{1}, \cdots, c_{i16}^{1}, c_{i1}^{2}, \cdots, c_{iM}^{16}, \sigma_{i1}, \cdots, \sigma_{iM}, w_{i1}, \cdots, w_{iM}\right]. \quad (20)$$

The design flow of the network is shown in Figure 12. We normalize the data so that the parameters of each dimension

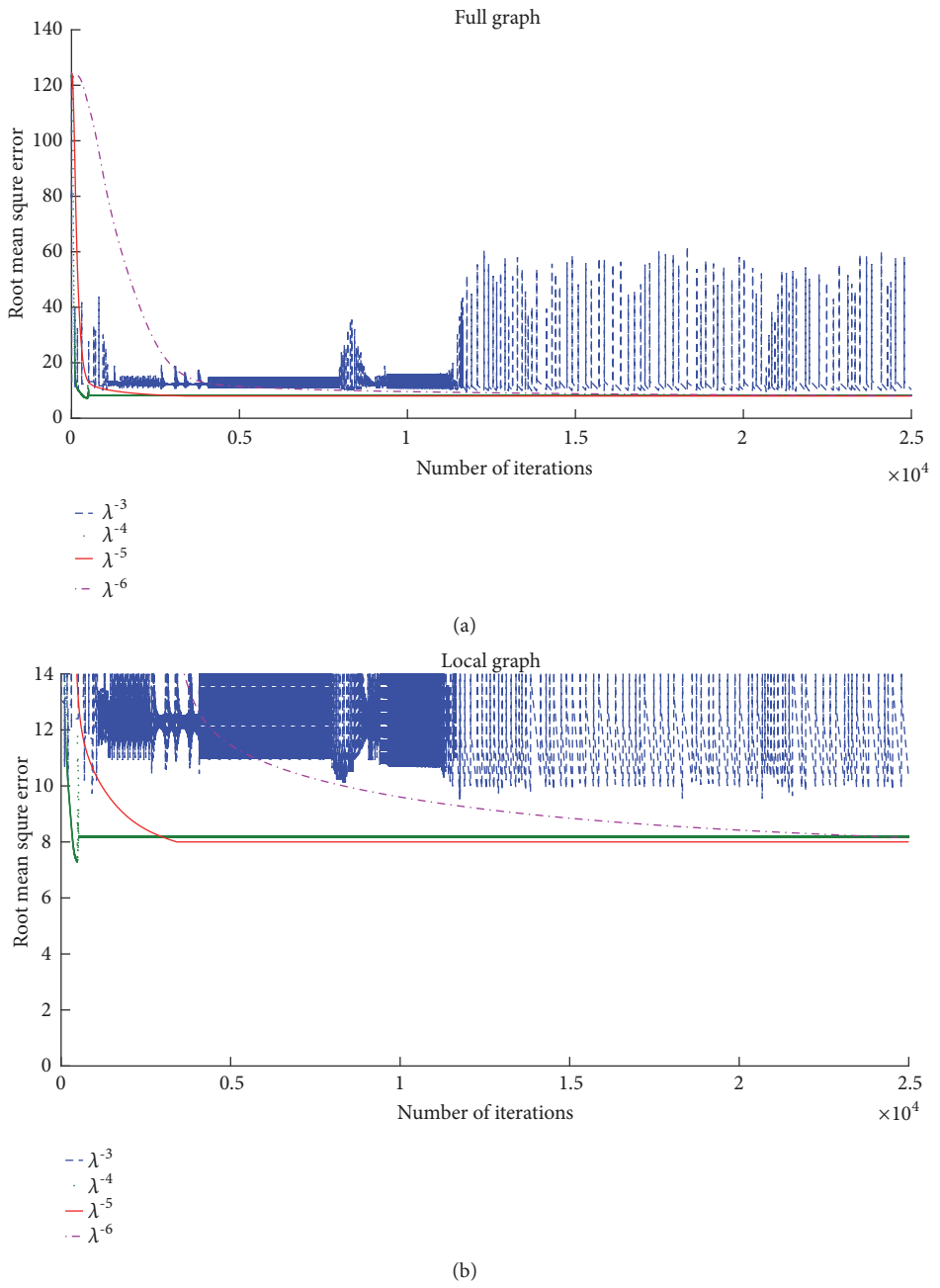


Figure 8: The curve of the RMS error decreases with the number of iterations at different learning rates.

Table 4: Forecast error analysis.

| Forecast cabin | Right chord cabin (2 people) | Right chord cabin (4 people) | Suite | Kitchen |
|---|---|---|---|---|
| Average relative error | 0.80% | 0.82% | 1.2% | 1.4% |
| Maximum relative error | 2.1% | 2.1% | 2.4% | 2.8% |
| Difference/dB(A) | <1 | <1 | <1.5 | <1.5 |

of the input variable are in the interval $[0, 1]$. In order to avoid the particles leaving the search space during training, we also set the boundary of the particle position. The first $16M$ dimension boundary is set to $[-5, 5]$, the next $M$ dimension is set to $[0, 200]$, and the output weight has no boundary. In MATLAB, we initialize the particle position to a uniformly distributed random number in the interval $[0, 1]$ and the velocity to normal random distribution. We calculate the particle fitness value according to the RMS error of the RBF network [18].

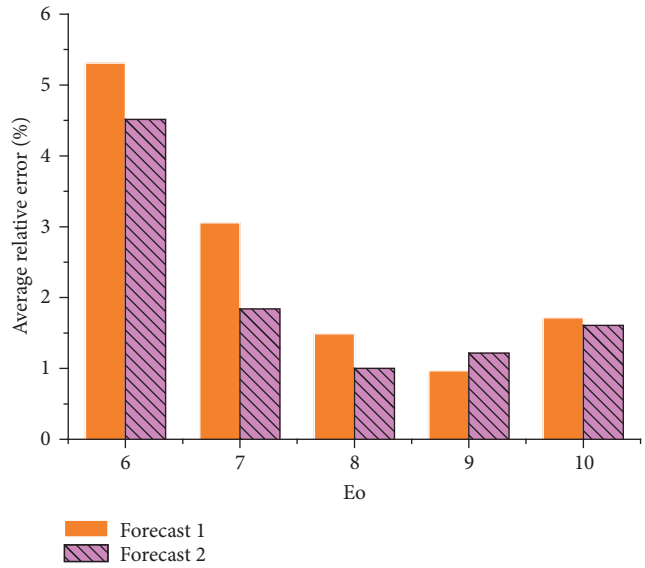


Figure 9: Average relative error corresponding to different $E_0$.

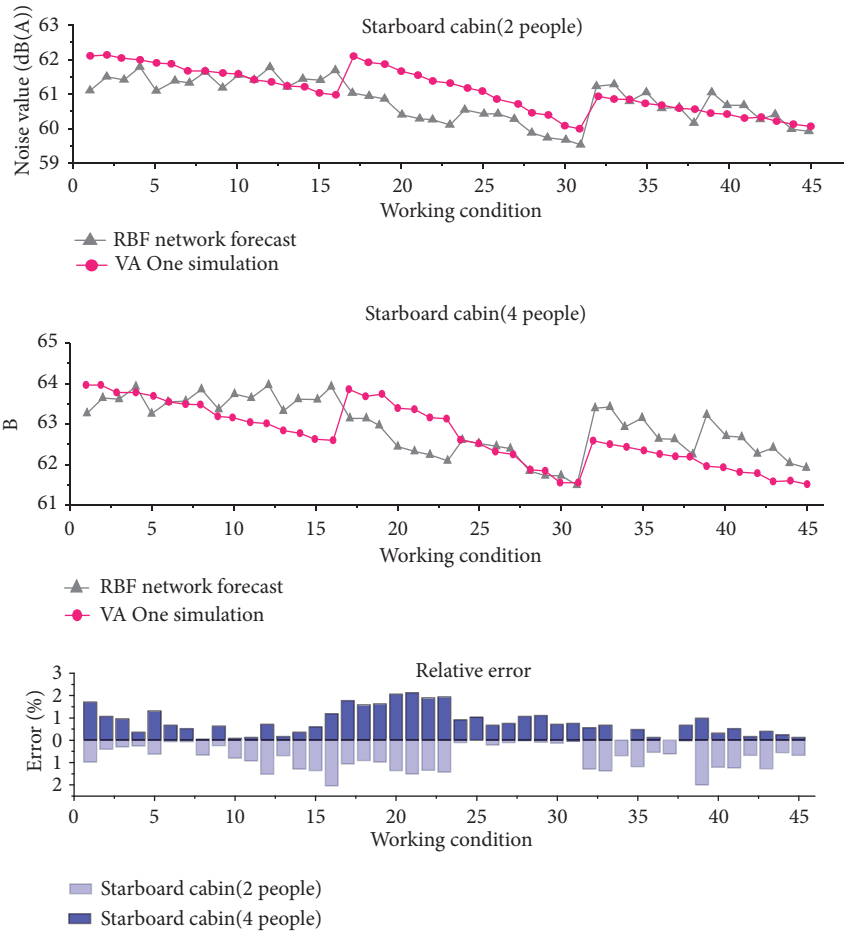


Figure 10: Prediction results and relative errors for noise in two cabins.

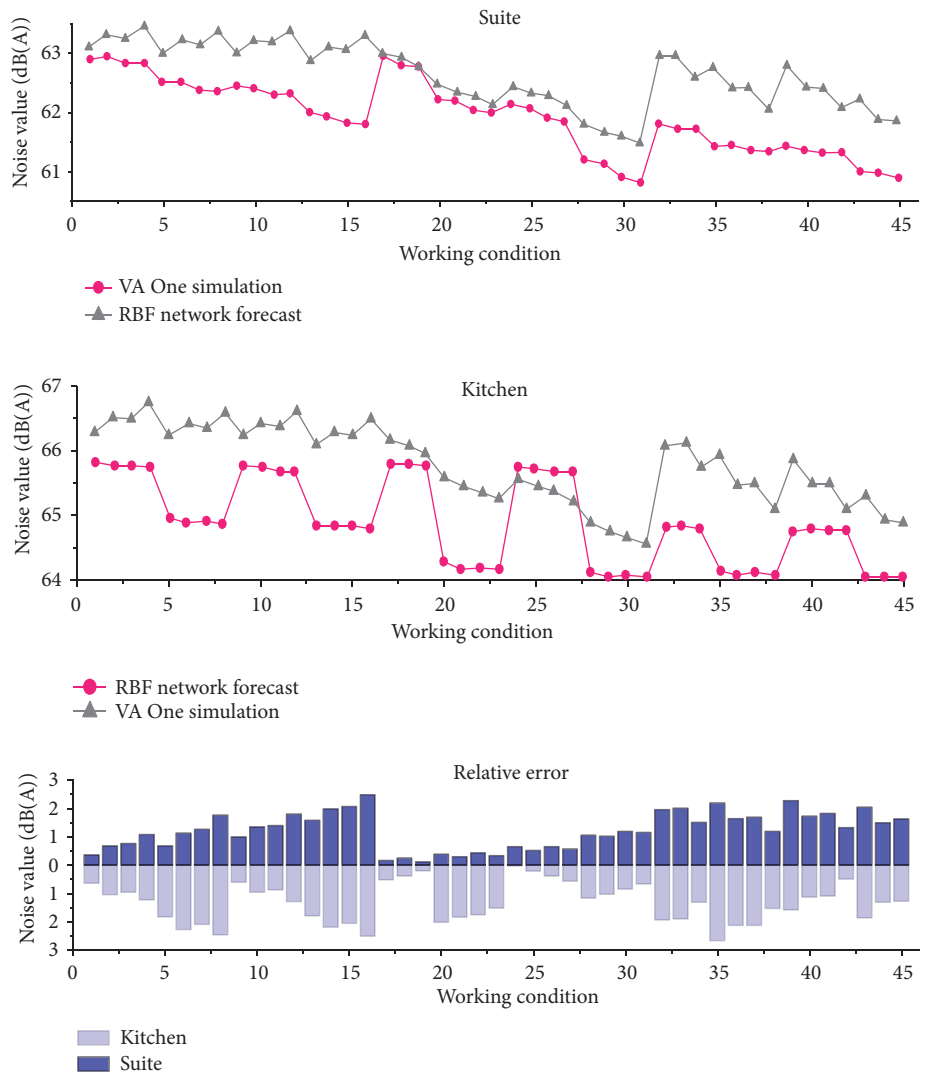


Figure 11: Prediction results and relative errors for noise in a suite and the kitchen.

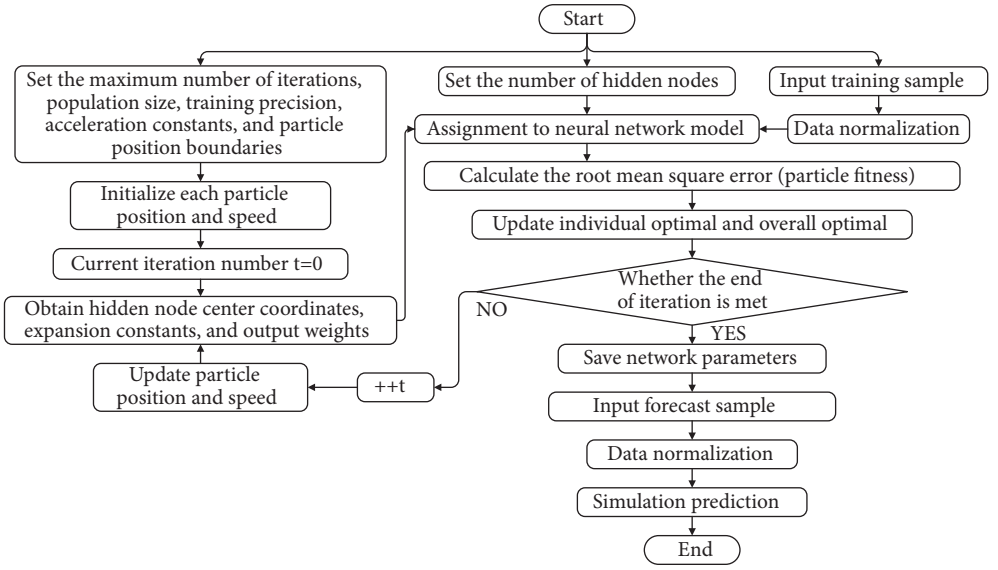


Figure 12: Flow chart of the particle swarm optimization algorithm in the RBF network.

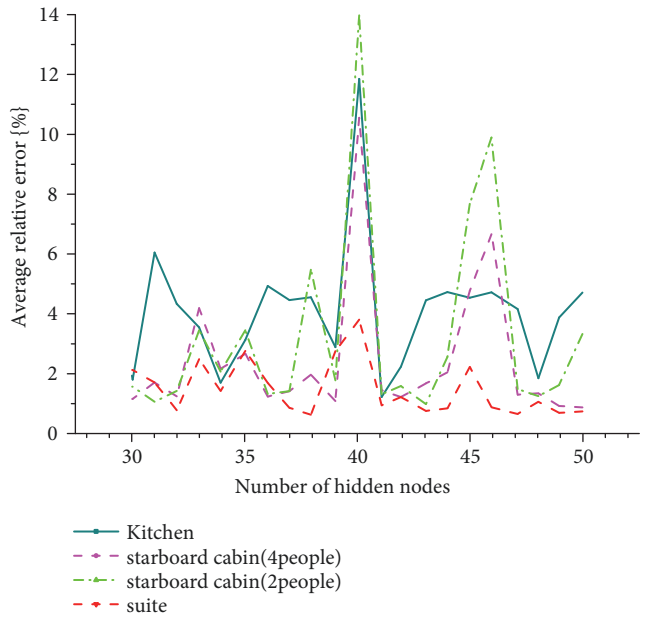


Figure 13: Effect of the number of hidden nodes on the forecast results.

*5.2. Parameter Selection and Effects on the PSO Algorithm.* Leaving other parameters unchanged ($m = 50, T_{\max} = 1200, E_0 = 4, c_1 = c_2 = 2$) [19, 20], we set the number of hidden nodes to an integer between 30 and 50 and used random weight optimization in the particle swarm optimization algorithm for the prediction experiment. The results are shown in Figure 13.

The figure shows that when the number of hidden nodes is 40, 45, or 46, the average relative error for the prediction of the four cabins is large, while the other cases fluctuate within a certain range. Since the particle swarm optimization algorithm is unstable, the algorithm converges to different training precisions even when the number of hidden nodes in the network is the same. Thus, the fluctuation range in the graph is not large. Therefore, we conclude that the number of hidden nodes has little effect on the forecast results. In our test, setting the number of hidden blocks to 41 results in the best prediction results, so we use 41 hidden nodes throughout the process described in this section.

Because the particle swarm optimization algorithm easily falls into the local minimum value, we now address enhancements to the algorithm's convergence behavior. We begin by leaving other parameters unchanged and evaluating three alternatives—shrinkage factor, linear weighting, and random weighting—to alter the convergence behavior. We train the PSO algorithm 10 times with 1000 iterations using each method. Figure 14 plots the RMS error of the convergence results according to the method. The figure shows that the shrinkage factor method has the worst convergence behavior, converging only to a relatively large local minimum value each time. The linear weight and random weight methods are better, with the random weight method being more stable than the linear weight method and converging to a larger local minimum with fewer iterations [21]. Modifying PSO to use random weights leads to far faster convergence speed than

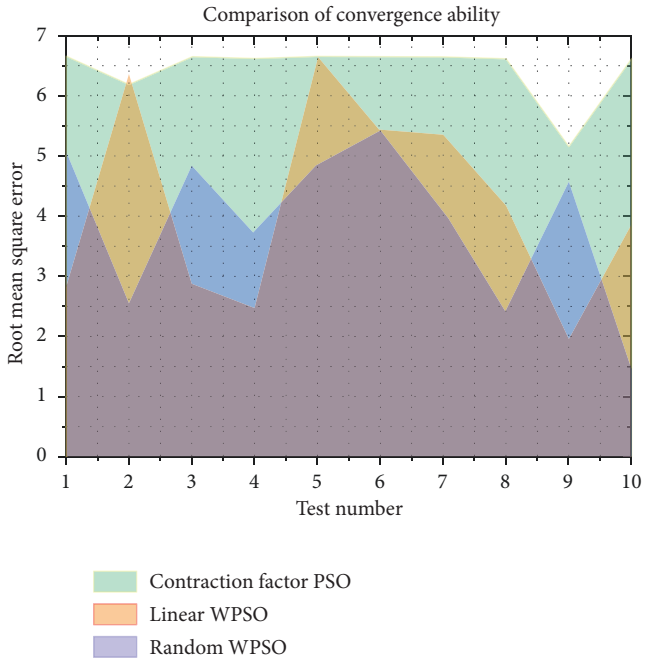


Figure 14: Particle swarm algorithm convergence ability results by method.

the gradient descent algorithm. The RMS error achieved by training 1000 times with random weights is smaller than that achieved by training 25000 times with the gradient descent algorithm [22].

We have determined experimentally that forecast results differ even when the training precision of the PSO algorithm is constant. Using a training accuracy $E_0$ between 2 and 4 increases the probability of producing good results. The use of an $E_0$ value less than 2 leads to overfitting due to noise in

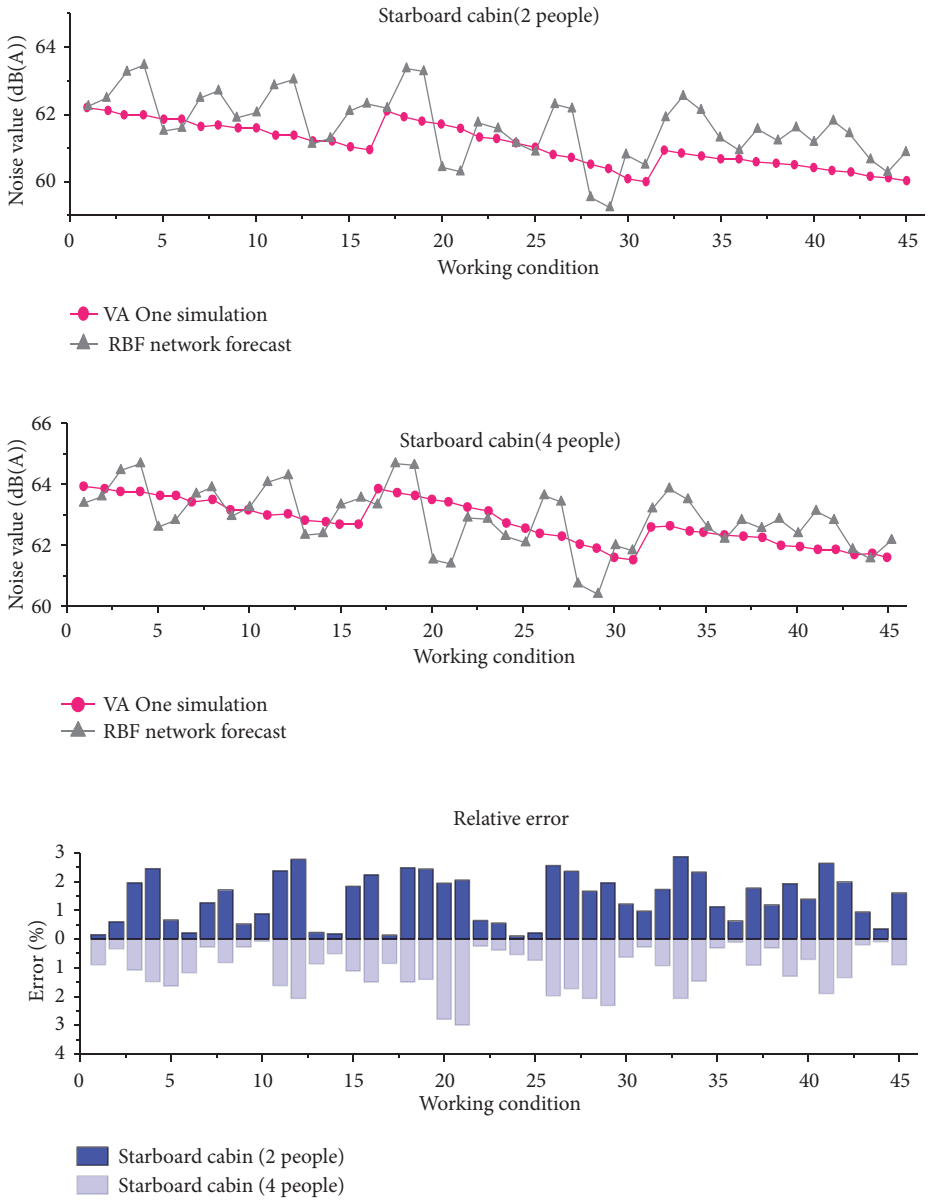


Figure 15: Prediction results and relative errors for noise in two cabins.

the data. Therefore, we set the training accuracy $E_0$ to 2, and if the actual training error of the algorithm is greater than 4, we must retrain the analysis.

*5.3. Prediction Error and Evaluation.* We also consider the quality of the predictions produced by our approach. We begin by setting our core parameters: number of particles $m = 50, T_{max} = 1200, E_0 = 2, c_1 = c_2 = 2$, and number of hidden nodes = 41. After cross-validating the forecasts, we choose the best results from our RBF network and plot them alongside the VA One software simulation in Figures 15 and 16 and Table 5.

The results show that the RBF network trained by our PSO algorithm offers higher accuracy with faster convergence and fewer training iterations compared to the gradient descent algorithm [23]. However, the PSO algorithm is also unstable, easily falls into the local optimal solution, and has many controlling parameters. Evaluating the above advantages and disadvantages, we combine the PSO algorithm with the gradient descent algorithm to train the RBF network for noise prediction to reduce the risk caused by algorithm instability and to improve the reliability of the forecast.

## 6. ADE-MRBF Network Design and Noise Prediction

The Differential Evolution (DE) algorithm is an optimization method proposed by R. Storm and K. Price in 1997 to solve the Chebyshev polynomial fitting problem [24]. The algorithm uses floating point calculations and offers fewer controlled parameters, easy implementation, good robust performance, and high reliability.

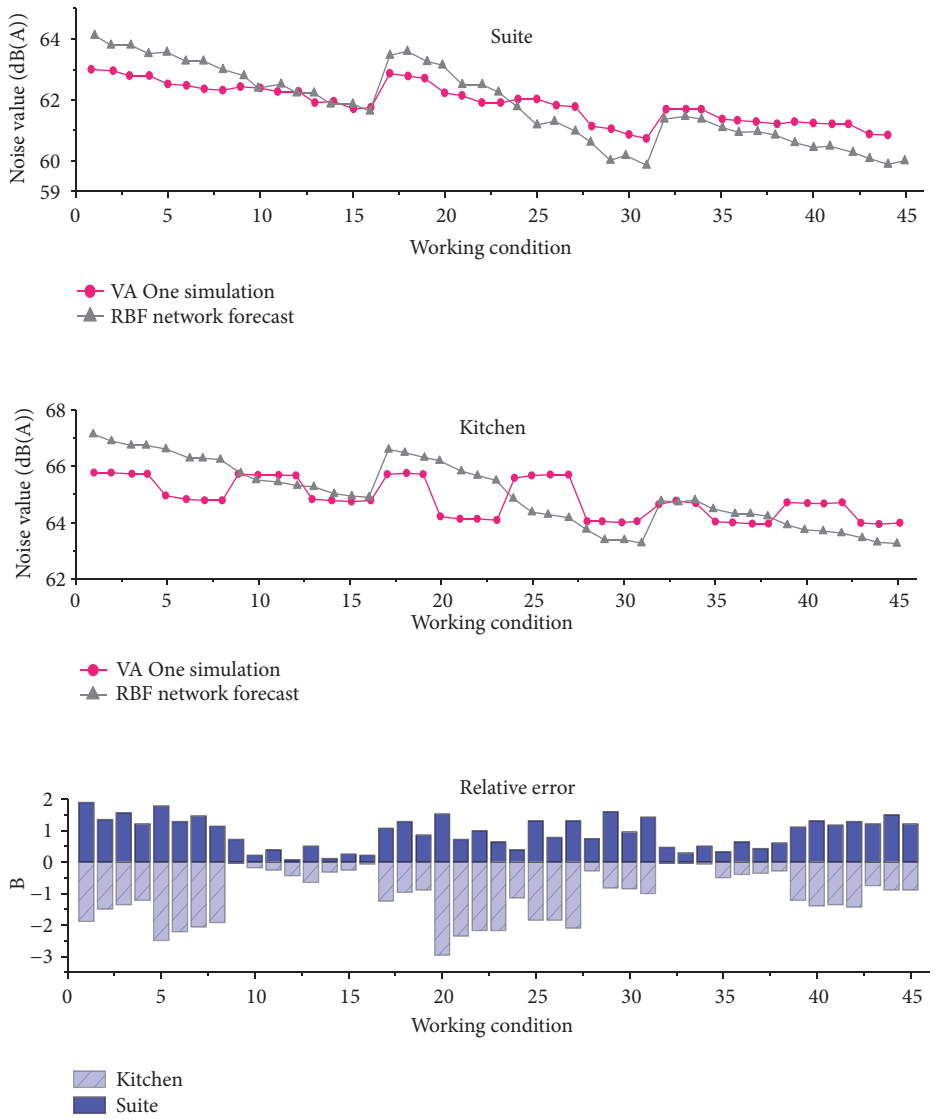


Figure 16: Prediction results and relative errors for noise in a suite and the kitchen.

Table 5: Forecast error analysis.

| Forecast cabin | Right chord cabin (2 people) | Right chord cabin (4 people) | Suite | Kitchen |
|---|---|---|---|---|
| Average relative error | 1.4% | 1.1% | 0.90% | 1.2% |
| Maximum relative error | 2.8% | 2.9% | 1.9% | 3.1% |
| Difference/dB(A) | <2 | <2 | <1 | <2 |

Our noise problem is complex. Even after simplification, 16 components are still extracted as input vector dimensions, and these components correlate with each other. Normalizing the input vector during preprocessing helps, but the method is still coarse. The addition of loss factors or other complex considerations reduces the performance of the algorithm. Therefore, we use the Mahalanobis distance instead of a simple normalization operation to decouple and dimensionless the sample to optimize the RBF network and predict the cabin noise [25]. We refer to our RBF network trained by the DE algorithm with adaptive parameter adjustment and based on Mahalanobis distance optimization as the ADE-MRBF network.

*6.1. Network Process Design.* The network design process is shown in Figure 17. Since the Mahalanobis distance is used instead of the Euclidean distance to measure the similarity between input variables and the hidden node centers, there is no need to normalize the input data. The initialization process is the same as that of the PSO algorithm, with all components initialized to a random number between 0 and 1 [26]. In addition, the center coordinate boundary of the hidden node

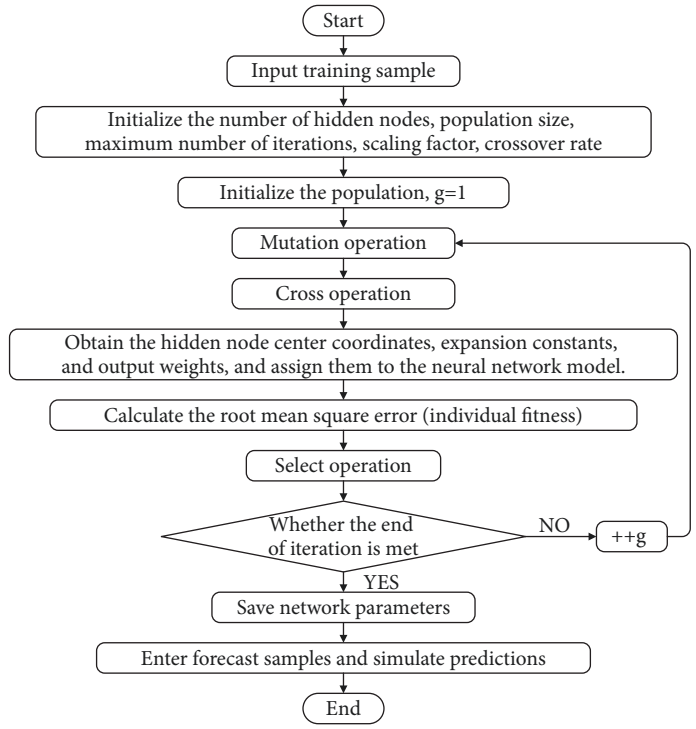


Figure 17: ADE-MRBF Network Design Flow.

in the cross operation is set to $[0, 7]$, the expansion constant is set to a positive number, and the output weight is not set to a boundary.

*6.2. Parameter Selection and Effects.* We note that while the number of hidden nodes has little effect on the performance of the algorithm, we use 40 hidden nodes to eliminate the interference of third-party factors and to facilitate the performance comparison between the DE algorithm and other algorithms [27].

*(1) Effect of Population Size on Convergence Performance.* Changing only the population size, we use the ADE-RBF algorithm when training with the data from the eight cabins on the upper two decks in the database. The training results are shown in Figure 18.

The figure shows that converge behavior is worst when the population size (NP) is 100, with the population diversity lacking in the later iterations, which leads to search stagnation. Increasing the NP causes the search stagnation phenomenon to appear later and strengthens the algorithm's global optimization ability. When the NP is greater than 200, the convergence curves are very close, which indicates that further increases in NP have little effect on the convergence behavior [28–31]. We use a population size of 300.

*(2) The Effect of Different Scaling Factor Strategies on Convergence Ability*. We also evaluate the choice of scaling factor in the reduction strategy. In this test, we change only the scaling factor, using one of four different adaptive reduction strategies as shown in (21), (22), (23), and (24) [32]. Exponential convergence strategy:

$$F(g) = F_{min} \cdot \left(\frac{F_{max}}{F_{min}}\right)^{e^{1-G_m/(1+G_m-g)}} \tag{21}$$

Linear strategy:

$$F(g) = F_{min} + \left(\frac{G_m + 1 - g}{G_m}\right) \cdot (F_{max} - F_{min}) \tag{22}$$

Parabolic strategy:

$$F(g) = F_{max} - (F_{max} - F_{min}) \cdot \left(\frac{g-1}{G_m}\right)^2 \tag{23}$$

Sigmoid strategy:

$$F(g) = F_{min} + (F_{max} - F_{min}) \cdot \frac{1}{1 + e^{(g-(1/2)(1+G_m))/(1/20)G_m}} \tag{24}$$

$F(g)$ is scaling factor; $F_{min}$ is scaling factor minimum; $F_{max}$ is scale factor maximum; $g$ is Generation G; $G_m$ is set maximum optimal algebra.

Training 20 times using data for the eight cabins on the upper two decks in the database gives the results as shown in Figure 19. The algorithm has the highest training precision along with good global convergence and local optimization ability when using the sigmoid strategy. The exponential convergence strategy has too short of a global optimization time and falls easily into the local minimum value in later stages. The parabolic strategy fails to perform local optimization near the global optimum in spite of the

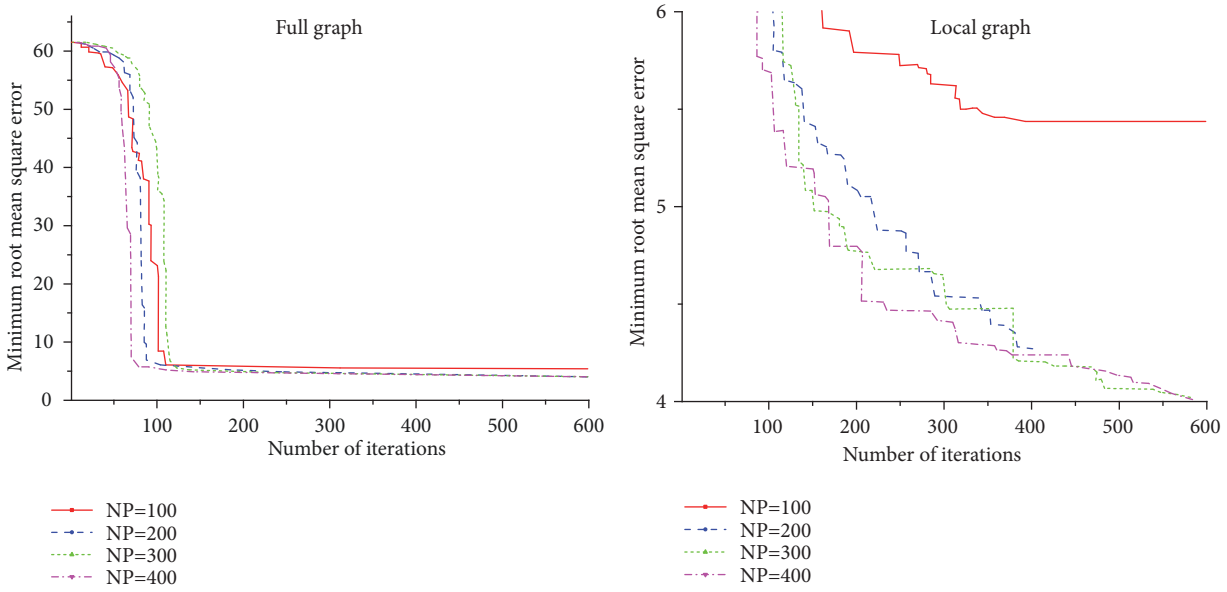


Figure 18: Algorithm convergence for different population sizes.

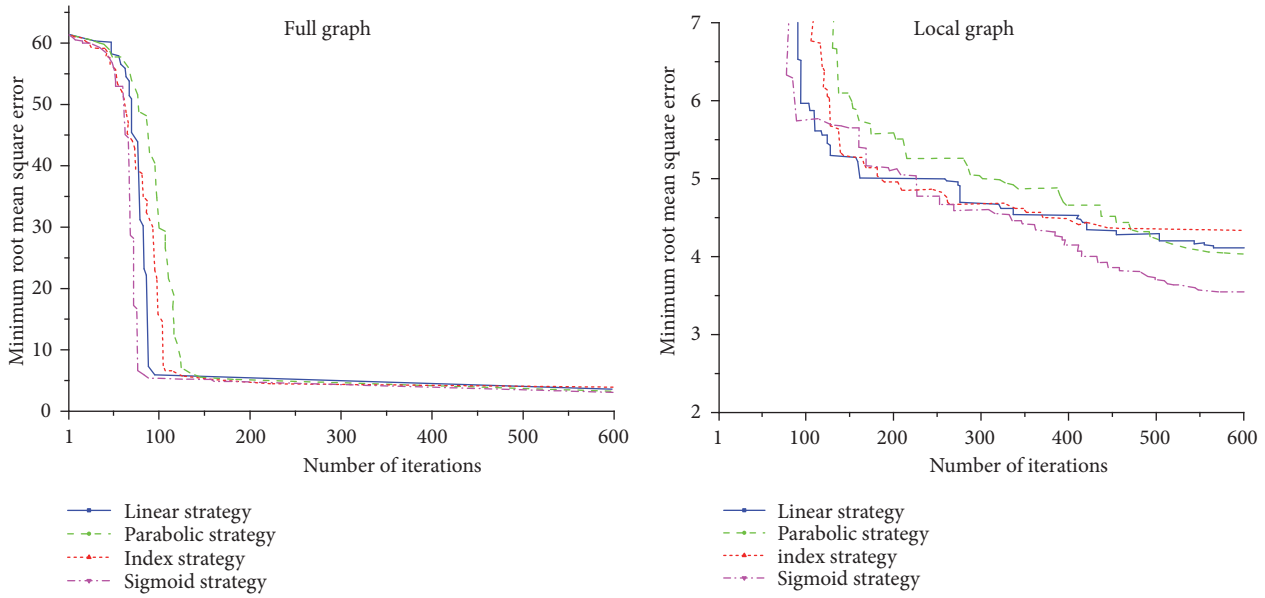


Figure 19: The convergence of the algorithm under different contraction factor strategies.

large number of global optimizations and offers insufficient training accuracy.

*(3) Influence of Different Cross-Rate Strategies on Convergence Ability*. As the iteration progresses, population diversity declines. Increasing the size of the population can solve this problem to a certain extent, but it greatly increases the amount of calculation necessary. We try to optimize the performance of the algorithm by improving the value of the crossover rate. Maintaining a low crossover rate in the early stage of the search is beneficial to the algorithm for global search [33]. Increasing the crossover rate in the later stage can improve the accuracy of the algorithm. Therefore, we adopt an exponential increment strategy for the crossover rate, as shown in the following equation:

$$CR = CR_{min} + \left(CR_{max} - CR_{min}\right) \cdot e^{-30(1-g/G_m)^3}. \quad (25)$$

CR is cross rate; $CR_{min}$ is the maximum value of the set cross rate; $CR_{max}$ is the minimum value of the set cross rate;$g$ is Generation $g$; $G_m$ is the maximum number of optimization algebras set.

The curve of the crossover rate $CR$ under the exponential strategy and the line increment strategy is shown in Figure 20. It can be seen from the figure that the $CR$ is almost unchanged at the beginning of the iteration when using the

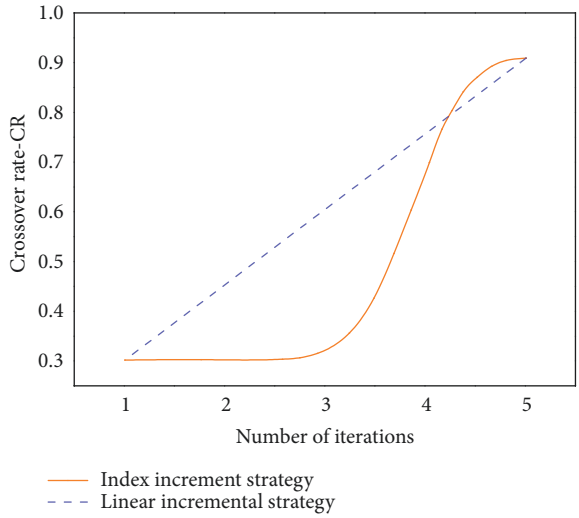


Figure 20: Change curve of crossover rate.

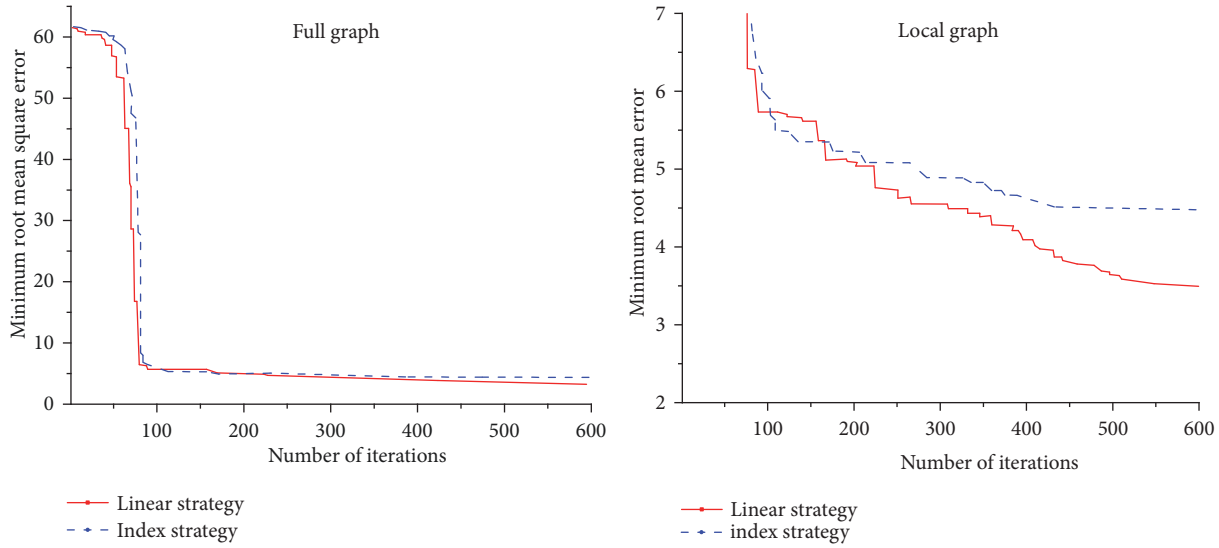


Figure 21: The convergence of the algorithm under different crossover rate strategies.

exponential increment strategy and stays near $CR_{\min}$, making the algorithm spend more time on the global search. The rapid increase of $CR$ in the late period maintains a high value, which is conducive to maintaining the diversity of the population [34]. After experimentation, this section selects $CR_{\min} = 0.3, CR_{\max} = 0.9$.

We want to compare cross-rate strategies also. Again, leaving other parameters unchanged, we trained the algorithm 20 times each with linear and exponential cross-rate increase strategies using the eight-cabin data from the upper two decks in the database. Figure 21 shows the average convergence results.

The results indicate better convergence ability with the exponential strategy. However, the algorithm needs more time for global optimization in the early stage [35, 36].

*(4) Influence of Mahalanobis Distance on Network Performance.* Keeping the other parameters unchanged, based on the differential evolution algorithm using Mahalanobis distance and Euclidean distance, 20 trainings were performed on the eight-cabin data on the upper two decks in the database. Figure 22 shows the plot of the minimum RMS error decreasing as the number of iterations increases.

The RBF network performs better using the Mahalanobis distance. This indicates that the Mahalanobis distance better measures the similarity between multidimensional variables, which compensates for the deficiencies in the data mining process to a certain extent and has specific strengths in complex multidimensional engineering problems [37].

*(5) Effect of Training Accuracy on Forecast Results.* Because noise is present in our data, we must take steps to avoid overfitting. We use the ADE-MRBF network to perform five predictions using cross-validation prediction methods under five different $E_0$ training precisions. The relative errors are averaged and plotted as Figure 23.

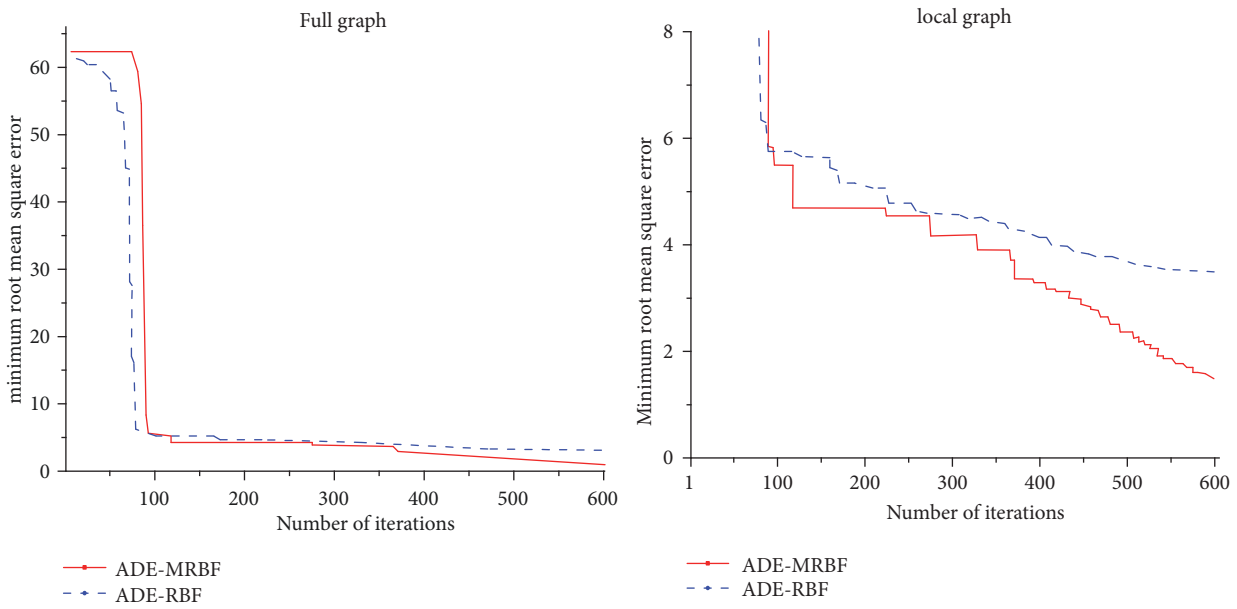


Figure 22: Network performance under different distance metrics.

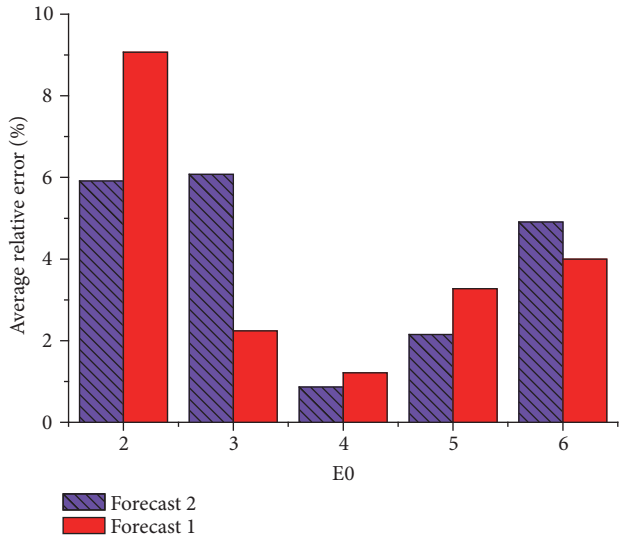


Figure 23: Forecast error under different training precision.

The figure shows that the prediction accuracy is the highest when the training precision $E_0$ is set to 4. When $E_0$ is below 3, the forecast error is increased, which causes overfitting. Therefore, the training accuracy should be set to 4.

*6.3. Prediction Error and Evaluation.* Based on the above test analysis, we combine the best performing parameters and improvement strategies and use the cross-validation forecasting method to carry out 20 predictions. The average forecast results under different working conditions are shown in Figures 24 and 25 and Table 6.

The results show that the RBF network trained by the DE algorithm has high accuracy for cabin noise prediction, low forecast error, fast convergence speed, and few controlling parameters. The scaling factor and crossover rate have little effect on the performance of the algorithm. The optimized RBF network has strong applicability in the field of ship cabin noise prediction.

## 7. Conclusion

Using an artificial intelligence method, we have been able to predict the noise in an unknown cabin of a ship. We reach the following conclusions.

(1) The large numbers of vibrating equipment coupled with an irregular hull shape and complicated structure produce a large number of sound source dimensions and make the forecast model complex and difficult to train. Using the Mahalanobis distance in the RBF network effectively

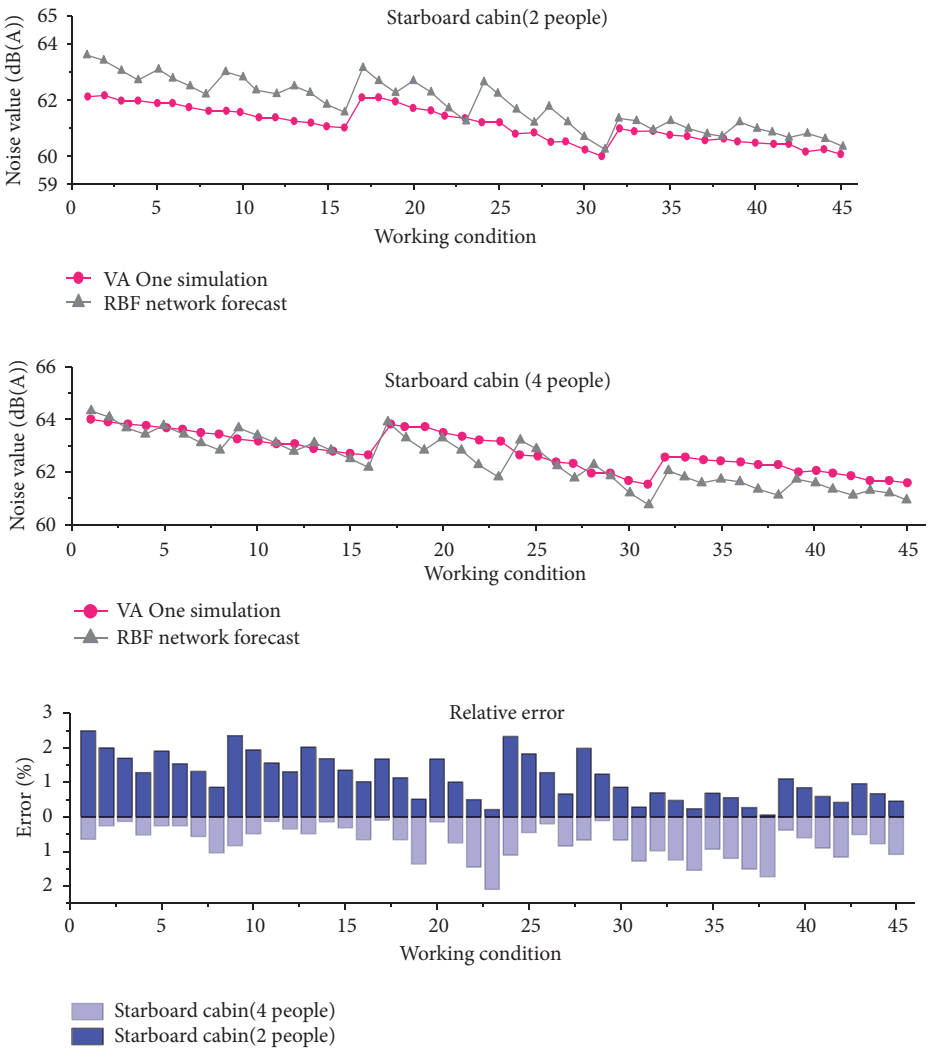


Figure 24: Prediction results and relative errors for noise in two cabins.

Table 6: Forecast error analysis.

| Forecast cabin | Right chord cabin (2 people) | Right chord cabin (4 people) | Suite | Kitchen |
|---|---|---|---|---|
| Average relative error | 1.2% | 0.71% | 0.49% | 0.75% |
| Maximum relative error | 2.5% | 2.1% | 1.4% | 2.6% |
| Difference/dB(A) | <1.5 | <1 | <1 | <1.5 |

decoupled the nondimensional influence compared with the Euclidean distance and had stronger classification ability when measuring the similarity between high-dimensional vectors.

(2) After evaluating the RBF network for predicting noise, we determined that the differential evolution algorithm obtains the best prediction results with fewer controlling parameters and greater flexibility. Further, this approach is less sensitive to changes in the controlling parameters. The combination of particle swarm optimization and gradient descent algorithms yields good performance, stability, and robustness for training. Our work provides a suitable reference for applying an RBF neural network for noise prediction.

(3) Our preliminary exploration of noise prediction for an unknown cabin of a ship under various working conditions shows that the forecast results from our RBF approach offer good accuracy overall. The average relative error is within 1%, and the maximum relative error is within 3%. We expect good noise prediction for other unknown cabins with similar structures.

## Data Availability

The data used to support the findings of this study are included within the article.

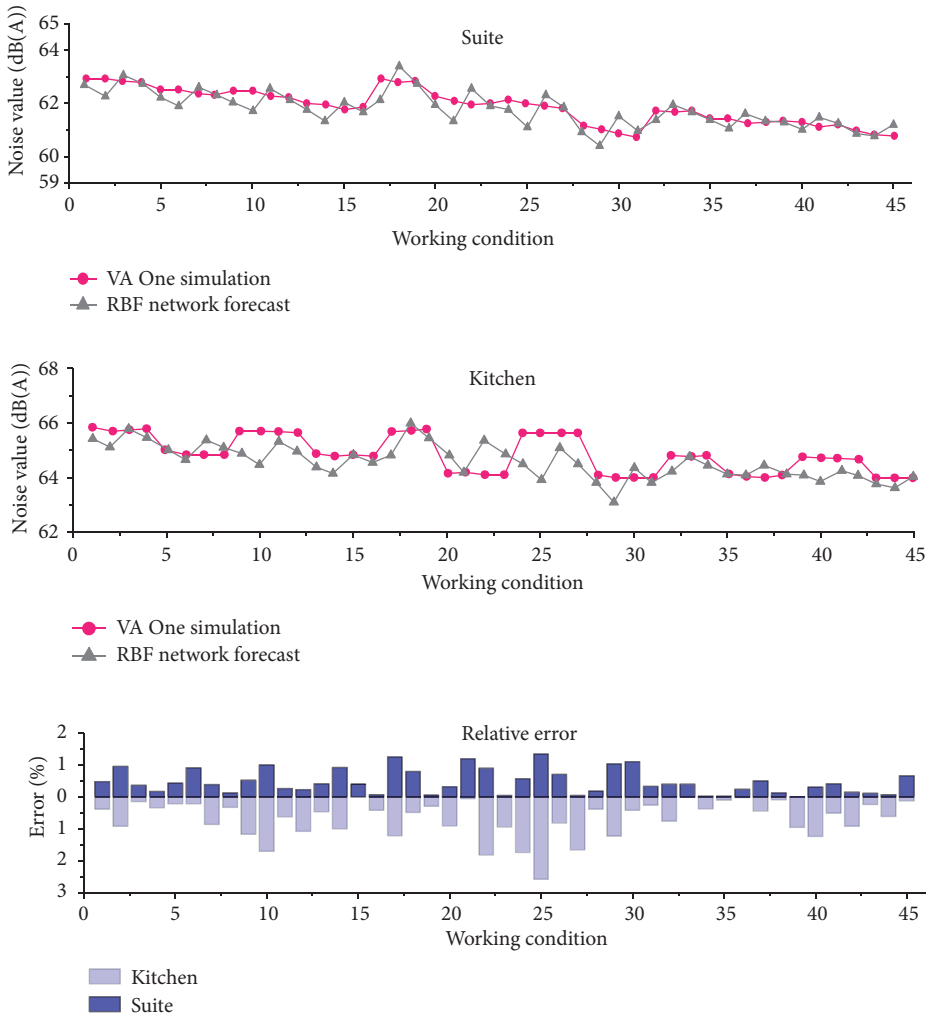


Figure 25: Prediction results and relative errors for noise in a suite and the kitchen.

## Conflicts of Interest

The authors declare that there are no conflicts of interest regarding the publication of this paper.

## Acknowledgments

This study was funded by high performance numerical wind tunnel algorithm and software research (20160131), High Technology Ship Funds of Ministry of Industry and Information of P.R. China, and major innovation projects of High Technology Ship Funds of Ministry of Industry and Information of P.R. China.

## References

[1] Y. Yang, C.-D. Che, and W.-Y. Tang, "Applications of reducing vibration and noises in a polar scientific icebreaker based on green shipbuilding technologies," *Journal of Ship Mechanics*, vol. 18, no. 6, pp. 724–737, 2014.

[2] B.-N. Liang, H.-L. Yu, and Y.-N. Cai, "Research on noise prediction and acoustics design of shipboard cabin," *Journal of Vibroengineering*, vol. 18, no. 3, pp. 1991–2003, 2016.

[3] W.-H. Joo, S.-H. Kim, J.-G. Bae, and S.-Y. Hong, "Control of radiated noise from a ship's cabin floor using a floating floor," *Noise Control Engineering Journal*, vol. 57, no. 5, pp. 507–514, 2009.

[4] R. Citarella and L. Federico, "Advances in vibroacoustics and aeroacustics of aerospace and automotive systems," *Applied Sciences*, vol. 8, no. 3, article no 366, 2018.

[5] A. Sabharwal and B. Selman, "Artificial intelligence: a modern approach," *Artificial Intelligence*, vol. 175, no. 5-6, pp. 935–937, 2011.

[6] D. Zhu, "The research progress and prospects of artificial neural networks," *Journal of Southern Yangtze University*, vol. 2004, no. 01, pp. 103–110, 2004.

[7] E. J. Hartman, J. D. Keeler, and J. M. Kowalski, "Layered neural networks with Gaussian hidden units as universal approximations," *Neural Computation*, vol. 2, no. 2, pp. 210–215, 1990.

[8] F. Girosi and T. Poggio, "Networks and the best approximation property," *Biological Cybernetics*, vol. 63, no. 3, pp. 169–176, 1990.

[9] H. Li, "Application of first-order shear deformation theory for the vibration analysis of functionally graded doubly-curved shells of revolution," *Composite Structures*, vol. 212, pp. 22–42, 2019.

[10] F. Pang, H. Li, H. Chen et al., "Free vibration analysis of combined composite laminated cylindrical and spherical shells with arbitrary boundary conditions," *Mechanics of Advanced Materials and Structures*, pp. 1–18, 2019.

[11] M. Dehghan and V. Mohammadi, "A numerical scheme based on radial basis function finite difference (RBF-FD) technique for solving the high-dimensional nonlinear Schrödinger equations using an explicit time discretization: Runge–Kutta method," *Computer Physics Communications*, vol. 217, pp. 23–34, 2017.

[12] W. Zhu and D. Fu, "PMSM control system based on RBF neural network," *Electronic Science and Technology*, vol. 29, no. 1, pp. 161–164, 2016.

[13] Z. Huang and H. Yuan, "Lonospheric single-station TEC short-term forecast using RBF neural network," *Radio Science*, vol. 49, no. 4, pp. 283–292, 2014.

[14] J. Fu, Y.-S. Wang, K. Ding, and Y.-S. Wei, "Research on vibration and underwater radiated noise of ship by propeller excitations," *Journal of Ship Mechanics*, vol. 19, no. 4, pp. 470–476, 2015.

[15] H. Yang, X. Li, and W. Jiang, "Simulation and analysis of stochastic parallel gradient descent control algorithm for adaptive optics system," *Acta Optica Sinica*, vol. 27, no. 8, pp. 1355–1360, 2007.

[16] P. Zhou, Z. Liu, X. Wang, Y. Ma, and X. Xu, "Theoretical and experimental investigation on coherent beam combining of fiber lasers using SPGD algorithm," *Acta Optica Sinica*, vol. 29, no. 8, pp. 2232–2237, 2009.

[17] R. K. Tyson, *Principles of Adaptive Optics*, Academic Press, Inc, San Diego, 1991.

[18] Y. Shi and R. Eberhart, "Particle swarm optimization," in *Proceedings of the IEEE International Conference on Neural Networks (ICNN '95)*, vol. 4, pp. 1942–1948, Perth, Western Australia, November-December 1995.

[19] L. Zhang, *The Theorem and Practice upon the Particle Swarm Optimization Algorithm*, Zhejiang Uniwersity, 2005.

[20] P. N. Suganthan, "Particle swarm optimiser with neighbourhood operator," in *Proceedings of the 1999 Congress on Evolutionary Computation, CEC 1999*, pp. 1958–1962, USA, July 1999.

[21] C. Guang, *Temprerature Predition Model Based on Improved PSO-RBF Neural Network*, Lanzhou University, 2015.

[22] G. P. Liu and Q. Zeng, "PSO based on multiple target optimization," *Journal of Hangzhou Teachers College: Natural Science Edition*, vol. 4, no. 1, pp. 30–33, 2005.

[23] G. Z. Chen, H. M. Xie, and X. Y. Lu, "Nonlinear optimization based on genetic algorithm toolbox of matlab," *Computer Technology and Development*, vol. 3, pp. 246–248, 2008.

[24] R. Storn and K. Price, "Differential evolution—a simple and efficient heuristic for global optimization over continuous spaces," *Journal of Global Optimization*, vol. 11, no. 4, pp. 341–359, 1997.

[25] X. H. Wu, S. J. Niu, and C. O. Wu, "An improvement on estimation covariance matrix during cluster analysis using mahalanobis distance," *Journal of Applied Statistics and Management*, vol. 30, no. 2, pp. 240–245, 2011.

[26] B. Liu, L. Wang, and Y. H. Jin, "Advances in differential evolutional," *Control and Decision*, vol. 22, no. 7, pp. 721–729, 2007.

[27] K. V. Price, R. M. Storn, and J. A. Lampinen, *Differential Evolution: A Practical Approach to Global Optimization*, Springer, Berlin, Germany, 2005.

[28] R. Mendes and A. S. Mohais, "DynDE: a differential evolution for dynamic optimization problems," *Evolutionary Computation*, vol. 3, pp. 2808–2815, 2005.

[29] T. Blackwell and J. Branke, "Multi-swarm optimization in dynamic environments," *Workshops on Applications of Evolutionary Computation*, vol. 3005, pp. 489–500, 2004.

[30] Y.-C. Lin, F.-S. Wang, and K.-S. Hwang, "A hybrid method of evolutionary algorithms for mixed-integer nonlinear optimization problems," in *Proceedings of the 1999 Congress on Evolutionary Computation, CEC 1999*, pp. 2159–2166, USA, July 1999.

[31] L. Wu, Y. Wang, and X. Yuan, "Differential evolution algorithm with adaptive second mutation," *Control and Decision*, vol. 21, no. 8, p. 898, 2006.

[32] M. Lin, F. Luo, and Y. Xu, "Optimization control of wastewater treatment process based on improved differential evolution algorithm," *Information and Control*, vol. 44, no. 3, pp. 339–345, 2015.

[33] Y. Tan, G. Z. Tan, and L. Tu, "Differential evolution algorithm with local search strategy," *Computer Engineering and Application*, vol. 45, no. 7, pp. 56–58, 2009.

[34] K. Zielinski, P. Weitkemper, R. Laur, and K.-D. Kammeyer, "Parameter study for differential evolution using a power allocation problem including interference cancellation," in *Proceedings of the 2006 IEEE Congress on Evolutionary Computation, (CEC '06)*, pp. 1857–1864, Canada, July 2006.

[35] W. Yang, F. Yao, and M. Zhang, "Differential evolution algorithm based on adaptive crossover probability factor and its application," *Information and Control*, vol. 39, no. 2, pp. 187–193, 2010.

[36] Z. X. Deng and X. J. Liu, "Study on strategy of increasing cross rate in differential evolution algorithm," *Computer Engineering and Applications*, vol. 44, no. 27, pp. 33–36, 2008.

[37] R. De Maesschalck, D. Jouan-Rimbaud, and D. L. Massart, "The Mahalanobis distance," *Chemometrics and Intelligent Laboratory Systems*, vol. 50, no. 1, pp. 1–18, 2000.