

Article

# A Best-Fitting B-Spline Neural Network Approach to the Prediction of Advection–Diffusion Physical Fields with Absorption and Source Terms

Xuedong Zhu <sup>1</sup>, Jianhua Liu <sup>1,2</sup>, Xiaohui Ao <sup>1,2,\*</sup>, Sen He <sup>1</sup>, Lei Tao <sup>1</sup> and Feng Gao <sup>1</sup>

- <sup>1</sup> School of Mechanical Engineering, Beijing Institute of Technology, Beijing 100081, China; 3120205243@bit.edu.cn (X.Z.); jeffliu@bit.edu.cn (J.L.); hesen9712@163.com (S.H.); bit\_taolei@163.com (L.T.); swai1213@163.com (F.G.)
- <sup>2</sup> Hebei Key Laboratory of Intelligent Assembly and Detection Technology, Tangshan Research Institute, Beijing Institute of Technology, Tangshan 063000, China
- \* Correspondence: xhao@bit.edu.cn

**Abstract:** This paper proposed a two-dimensional steady-state field prediction approach that combines B-spline functions and a fully connected neural network. In this approach, field data, which are determined by corresponding control vectors, are fitted by a selected B-spline function set, yielding the corresponding best-fitting weight vectors, and then a fully connected neural network is trained using those weight vectors and control vectors. The trained neural network first predicts a weight vector using a given control vector, and then the corresponding field can be restored via the selected B-spline set. This method was applied to learn and predict two-dimensional steady advection–diffusion physical fields with absorption and source terms, and its accuracy and performance were tested and verified by a series of numerical experiments with different B-spline sets, boundary conditions, field gradients, and field states. The proposed method was finally compared with a generative adversarial network (GAN) and a physics-informed neural network (PINN). The results indicated that the B-spline neural network could predict the tested physical fields well; the overall error can be reduced by expanding the selected B-spline set. Compared with GAN and PINN, the proposed method also presented the advantages of a high prediction accuracy, less demand for training data, and high training efficiency.

**Keywords:** B-spline; best-fitting; physical fields; neural network; field gradient



**Citation:** Zhu, X.; Liu, J.; Ao, X.; He, S.; Tao, L.; Gao, F. A Best-Fitting B-Spline Neural Network Approach to the Prediction of Advection–Diffusion Physical Fields with Absorption and Source Terms. *Entropy* **2024**, *26*, 577. <https://doi.org/10.3390/e26070577>

Academic Editor: Kevin H. Knuth

Received: 6 May 2024

Revised: 20 June 2024

Accepted: 2 July 2024

Published: 4 July 2024



**Copyright:** © 2024 by the authors. Licensee MDPI, Basel, Switzerland. This article is an open access article distributed under the terms and conditions of the Creative Commons Attribution (CC BY) license (<https://creativecommons.org/licenses/by/4.0/>).

## 1. Introduction

At present, industry and scientific research extensively involve various forms of physical fields and their computation and simulation, such as flow field, temperature field, etc. The traditional way, which applies various numerical methods to solve these complex mathematical models, usually encounters problems related to complex methods, poor convergence, and the high cost in terms of time, and presents poor adaptability, especially for systems with strong nonlinearity and whose model parameters are difficult to determine [1–6]. In recent years, neural networks represented by deep learning have been successfully applied to the learning and prediction of many complex systems and scenarios, and applying machine learning to predict fields has become a hot topic [7–9]. However, a field has the characteristics of continuous spatial distribution, and field prediction needs to consider each spatial point of the huge amount of field data rather than one or more individual features. If a traditional fully connected neural network is directly used to predict the field, the number of nodes in the network output layer will be very large, making network training extremely difficult. By selecting an appropriate set of basis functions, fitting the huge amount of field data as low-dimensional vectors, and then training an ordinary neural network on the low-dimensional data, a better neural network model can

be obtained [10,11]. At present, there exist some methods integrating neural networks and basis functions, such as the B-spline neural network (BSNN), trigonometric-function neural network, and Walsh-function neural network [12,13].

Machine learning has a powerful ability to process complicated data. Its deep model hierarchical structure can automatically extract the inherent feature information of the original complex data and perform state recognition. Using machine-learning methods to predict spatially continuous fields has generated many concerns recently and the related challenges are still great. Convolutional neural networks (CNN) [14–16], generative adversarial networks (GAN) [17–21], and physics-informed neural networks (PINN) [22–26] are now widely used machine-learning methods to predict things with field features. As one of the most popular machine-learning methods, GAN is suited to predict the output field variables of complicated systems. It has been successfully utilized in many fields, including stress, deformation, and temperature fields, owing to its remarkable power in dealing with strongly nonlinear problems [27–30]. For the development of GAN, many novel GAN-based methods emerge, such as cGAN [18], StressGAN [31], and DCGAN [32]. PINN, a machine-learning method designed for the prediction of physical fields, has been successfully used to solve partial differential equations (PDEs) or PDE-based problems by adding data constraints and physics constraints in the loss function to constrain the space for admissible solutions. Benefiting from physics constraints, the PINN can reduce the requirement for labelled data and can be easily applied to a small data regime [33,34]. Xie et al. [35] predicted a 3D temperature field using a physics–data hybrid PINN model and presented a good predicted result. Zhu et al. [36] applied the PINN to predict the temperature and melt pool dynamics during metal AM processes.

For the learning and prediction problems of continuous physical systems, an ordinary neural network is trained by directly using the input and output data of the system, while a basis function–neural network combined method first fits the continuously distributed data using a selected function set and then obtains the weight vectors with small data, which are used to train the neural network ordinarily, finally obtaining the surrogate model of the physical system. The B-spline function has many advantages, such as local piecewise, which allows the function approximation to be adjusted locally; that is, the weight and parameter changes in a basis function only affect the local approximation, rather than the global. This feature of the B-spline function greatly enhances the stability of approximating complex functions [37] and can obtain a good fit for complex curves or surfaces [38]. A B-spline neural network learning a continuous system must learn a representing vector that can well approximate continuous data samples. Compared with directly using a fully connected neural network, BSNN is a local weight update method with the advantages of a fast convergence rate and low computational complexity and can greatly reduce the data size and training time [39,40]. Currently, BSNN has many successful applications in system identification and the prediction of zero-dimensional time series. Zhang et al. [41] used BSNN to approximate the nonlinear term of a satellite communication antenna system and compared it with a variety of traditional methods as well as radial basis function neural networks. Their results indicated that the BSNN has better decoupling and anti-interference abilities and predicts a more accurate and stable result. This is because B-spline functions have a good local adjustment feature, which leads to their good nonlinear approximation capabilities [42]. In recent years, the basis function neural network has had many other applications, and its superiority has been continuously verified.

The application of basis functions in neural network modeling is becoming more and more extensive, and the ability to predict physical fields using a neural network method is also on the way; however, the process of combining basis functions and neural networks to predict spatially continuous fields, which has great potential, still needs more research. This work proposed a best-fitting B-spline neural network (bBSNN). The process of fitting the physical field and minimizing the error draws on the coding process and best coding scheme in information theory. Its accuracy and performance were tested by a series of PDE numerical experiments with different vector sizes, field gradients, and field states, and

verified through comparison with GAN and PINN. The effectiveness of the bBSNN was validated and the prediction error was analyzed and discussed.

## 2. Theory

### 2.1. Best-Fitting B-Spline Function

A one-dimensional B-spline function set with an order of  $k$  and an element number of  $m$  can be defined as follows:

$$\begin{cases} N_j^k(x) = \frac{x-t_j}{t_{j+k}-t_j} N_j^{k-1}(x) + \frac{t_{j+k+1}-x}{t_{j+k+1}-t_{j+1}} N_{j+1}^{k-1}(x) \\ N_j^0(x) = \begin{cases} 1, & t_j \leq x \leq t_{j+1} \\ 0, & \text{elsewhere} \end{cases} \end{cases} \quad (1)$$

where  $\{t_j\}$  is a knot sequence,  $\{t_j | j = 1, 2, \dots, m + k + 1\}$ . Assuming that  $x$  goes from  $x_{\min}$  to  $x_{\max}$ , the knot sequence for a uniform B-spline function can then be provided in ascending order as follows:

$$t_1 < t_2 < \dots < t_{k+1} = x_{\min} < t_{k+2} < \dots < t_m < x_{\max} = t_{m+1} < \dots < t_{m+k+1} \quad (2)$$

Here,  $t_j$  ( $j < k + 1$  or  $j > m + 1$ ) are external knots,  $t_j$  ( $j > k + 1$  or  $j < m + 1$ ) are internal knots, and the other two are end knots. For a higher-dimension B-spline function, all the one-dimensional functions defined in each dimension need to be multiplied, as in Equation (1). Giving a two-dimensional function  $\phi(x, y)$ , we can have approximation  $\phi^*(x, y)$  composed of a linear combination of B-splines and a weight vectors  $\mathbf{a}$ , as follows:

$$\phi^*(x, y) = \sum_{l=1}^{m_y} \sum_{j=1}^{m_x} a_{lj} N_j^{k_x}(x) N_l^{k_y}(y) = \mathbf{N}(x, y)^T \mathbf{a} \quad (3)$$

Here,  $m_x$  and  $m_y$  are the numbers of B-spline functions in the  $x$  and  $y$  dimensions, respectively, and  $k_x$  and  $k_y$  are their corresponding orders. Rewriting this formula in terms of matrices, we have:

$$\mathbf{N}(x, y) = \left\{ [N_{11} \dots N_{1j} \dots N_{1m_x}] \dots [N_{l1} \dots N_{lj} \dots N_{lm_x}] \dots [N_{m_y1} \dots N_{m_yj} \dots N_{m_y m_x}] \right\}^T \quad (4)$$

and

$$\mathbf{a} = \left\{ [a_{11} \dots a_{1j} \dots a_{1m_x}] \dots [a_{l1} \dots a_{lj} \dots a_{lm_x}] \dots [a_{m_y1} \dots a_{m_yj} \dots a_{m_y m_x}] \right\}^T \quad (5)$$

where  $N_{lj} = N_j^{k_x}(x) N_l^{k_y}(y)$ . In practice, function  $\phi(x, y)$  is often given by  $n_x \times n_y$  discrete points,  $\phi(x_i, y_p)$ , for which we have a fitting error:

$$E = \frac{1}{2} \sum_{p=1}^{n_y} \sum_{i=1}^{n_x} [\phi(x_i, y_p) - \phi^*(x_i, y_p)]^2 \quad (6)$$

Applying the least square method to minimize the fitting error  $E$ , we have a best-fitting weight vector  $\mathbf{a}$ :

$$\mathbf{a} = [\mathbf{A}^T \mathbf{A}]^{-1} [\mathbf{A}^T \mathbf{B}] \quad (7)$$

Here

$$\mathbf{A} = \left\{ [\mathbf{N}(x_1, y_1) \dots \mathbf{N}(x_i, y_1) \dots \mathbf{N}(x_{n_x}, y_1)] \dots [\mathbf{N}(x_1, y_p) \dots \mathbf{N}(x_i, y_p) \dots \mathbf{N}(x_{n_x}, y_p)] \dots [\mathbf{N}(x_1, y_{n_y}) \dots \mathbf{N}(x_i, y_{n_y}) \dots \mathbf{N}(x_{n_x}, y_{n_y})] \right\}^T \quad (8)$$

and

$$\mathbf{B} = \left\{ [\phi(x_1, y_1) \dots \phi(x_i, y_1) \dots \phi(x_{n_x}, y_1)] \dots [\phi(x_1, y_p) \dots \phi(x_i, y_p) \dots \phi(x_{n_x}, y_p)] \dots [\phi(x_1, y_{n_y}) \dots \phi(x_i, y_{n_y}) \dots \phi(x_{n_x}, y_{n_y})] \right\}^T \quad (9)$$

By using the best fitting, a function  $\phi(x, y)$  or its discrete version  $\phi^*(x, y)$  can be represented by a weight vector and a pre-selected B-spline function set; as a consequence, a data set including amounts of  $\phi^*(x, y)$  can be degraded into a series of corresponding weight vectors along with a known B-spline function set. In general, the scale of  $\phi^*(x, y)$  is far greater than that of the corresponding weight vector, which is determined by the number of B-splines in the selected function set. Additionally, B-spline fitting can filter the high-frequency modes in  $\phi^*(x, y)$ ; the extent of this can be modified by changing the number or the order of the B-splines in the function set. Since the B-spline function is piecewise, the B-spline fitting can well approximate the high-frequency modes through the addition of more low-order B-splines to the function set instead of higher-order basis functions.

2.2. Combination of Neural Network and B-Spline Function

The architecture of the bBSNN is shown in Figure 1, which combines a fully connected neural network and the B-spline function fitting. A data set {OUT:  $\phi(x, y)$ } with  $r$  samples along with  $r$  corresponding control vectors {IN:  $\mathbf{p}$ } can be used as an example.

$$\begin{aligned} \{\text{In} : \mathbf{p}\} &= \{p_1, p_2, \dots, p_r\} \\ \{\text{Out} : \phi(x, y)\} &= \{\phi_{1,1}, \phi_{1,2}, \dots, \phi_{n_x, n_y}\} \end{aligned} \tag{10}$$

The training and prediction of this method can be illustrated as follows (Please refer to the Supplementary Materials for the bBSNN code.):

- (1) Use the numbers  $m_x$  and  $m_y$  and the orders  $k_x$  and  $k_y$ , and calculate the knot sequence in each dimension according to the domain boundaries of the data set, then generate the B-spline function set;
- (2) Calculate the best-fitting weight vector  $\mathbf{a}$  using the least square method; thus, the data set can be replaced by:

$$\{\text{Out} : \phi(x, y)\} = \{a_{1,1}, a_{1,2}, \dots, a_{m_x, m_y}\} \tag{11}$$

- (3) Train an ordinary fully connected neural network whose input and output are now as follows:

$$\begin{aligned} \{\text{In} : \mathbf{p}\} &= \{p_1, p_2, \dots, p_r\} \\ \{\text{Out} : \mathbf{a}\} &= \{a_{1,1}, a_{1,2}, \dots, a_{m_x, m_y}\} \end{aligned} \tag{12}$$

- (4) Once the neural network is well trained, a new control vector  $\mathbf{p}$  can be used to predict a new weight vector  $\mathbf{a}$ ; predicted function  $\phi^*(x, y)$  is then restored by Equation (3).

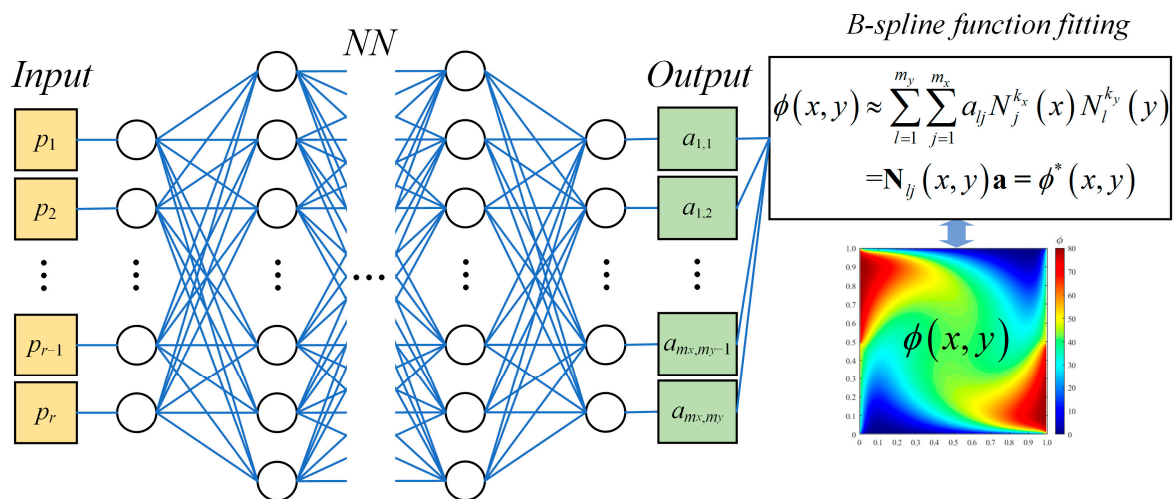


Figure 1. A schematic of the architecture of the best-fitting B-spline neural network.



### 3. Numerical Experiments and Discussions

In this section, we focus on the numerical experiments conducted on general advection–diffusion physical fields, using absorption and source terms to examine the performance of the bBSNN, in which the boundary conditions and the scalar field are regarded as the input control vector {IN:  $\mathbf{p}$ } and the output data {OUT:  $\phi(x, y)$ }, respectively. Here, we consider the physical fields as follows:

$$a\nabla^2\phi - b\nabla \cdot (\mathbf{u}\phi) - c\phi + d = 0 \tag{13}$$

where  $\phi$  is a scalar field and  $\mathbf{u}$  is a vector field, such as the velocity field.  $a, b, c,$  and  $d$  are the coefficients of the diffusion, convection, absorption, and source terms, respectively. In the numerical experiments, a vortical velocity field with an average velocity of 1 is implanted into the PDE and expressed as follows:

$$\mathbf{u} = \begin{bmatrix} u(x, y) \\ v(x, y) \end{bmatrix} = 2 \begin{bmatrix} 0 & 1 \\ -1 & 0 \end{bmatrix} \begin{bmatrix} x/(0.5L_x) - 1 \\ y/(0.5L_y) - 1 \end{bmatrix} \tag{14}$$

Here,  $L_x$  and  $L_y$  are the dimensions of the domain;  $u = u(x, y)$  and  $v = v(x, y)$  are the velocity components. The physical fields can be characterized by three parameters: (1)  $\alpha = a/b$ , representing the diffusion rate of the scalar field; (2) Péclet number  $Pe = LU/\alpha$ , a dimensionless number representing the ratio of the convection intensity to the diffusion intensity of the scalar field; (3)  $My = L^2C/a$ , a dimensionless number representing the ratio of the absorption intensity to the diffusion intensity of the scalar field. Here,  $L, U,$  and  $C$  are the characteristic length, velocity, and absorption rate of the system, respectively. (Please refer to the Supplementary Materials for the finite volume method calculation code of the above physical field.)

#### 3.1. Numerical Experiment Setup and Preliminary Verification

For the numerical experiment setup, a square domain is configured for the physical fields, and two independent and variable conditions and two fixed conditions are set at the four boundaries of the domain for preliminary verification, as shown in Figure 2, yielding an input control vector  $\mathbf{p} = [B_1, B_2]^T$ , whose values are slightly smoothed in the vicinity of their junction points by a hyperbolic tangent function. The parameters of the physical fields are given as  $a = 10, b = 200,$  and  $c = 0.5, d = 1000,$  which yields a field state with  $\alpha = 0.05, Pe = 20,$  and  $My = 0.05$ . The numerical result of the scalar field  $\phi(x, y)$  is obtained with a grid resolution of  $128 \times 128$  using a finite volume method. The configuration of the B-spline function set is  $k_x = k_y = 2$  and  $m_x = m_y = 10,$  determining a weight vector  $\mathbf{a}$  with a size of  $100 \times 1$ .

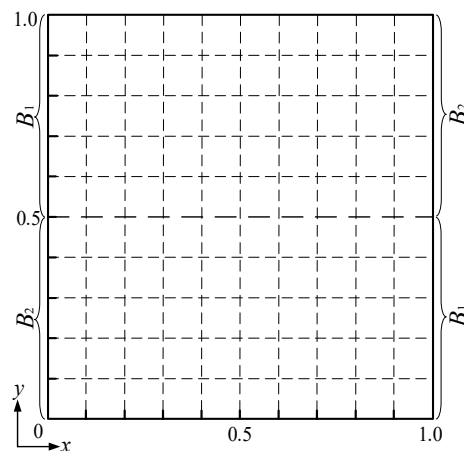
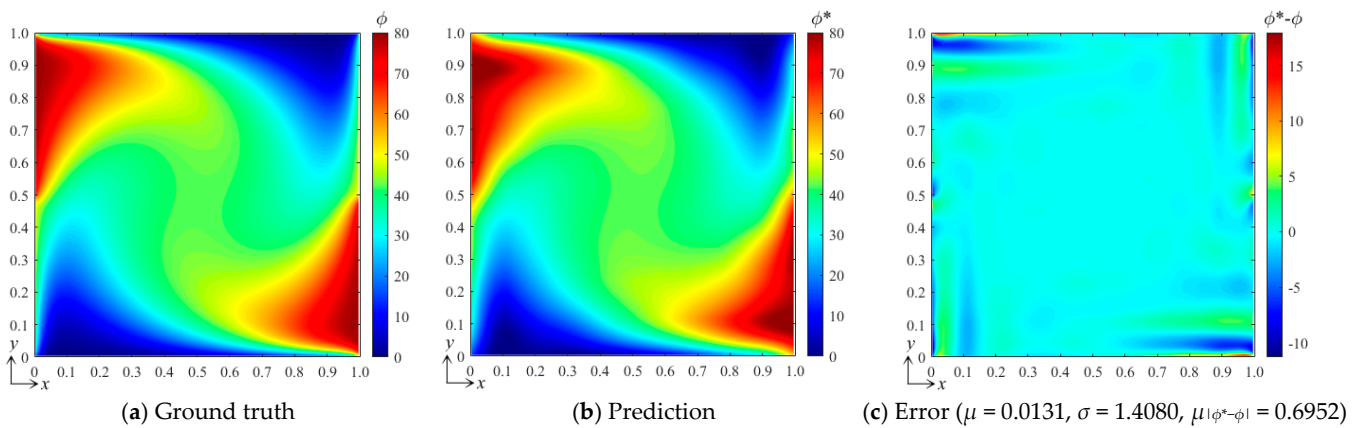


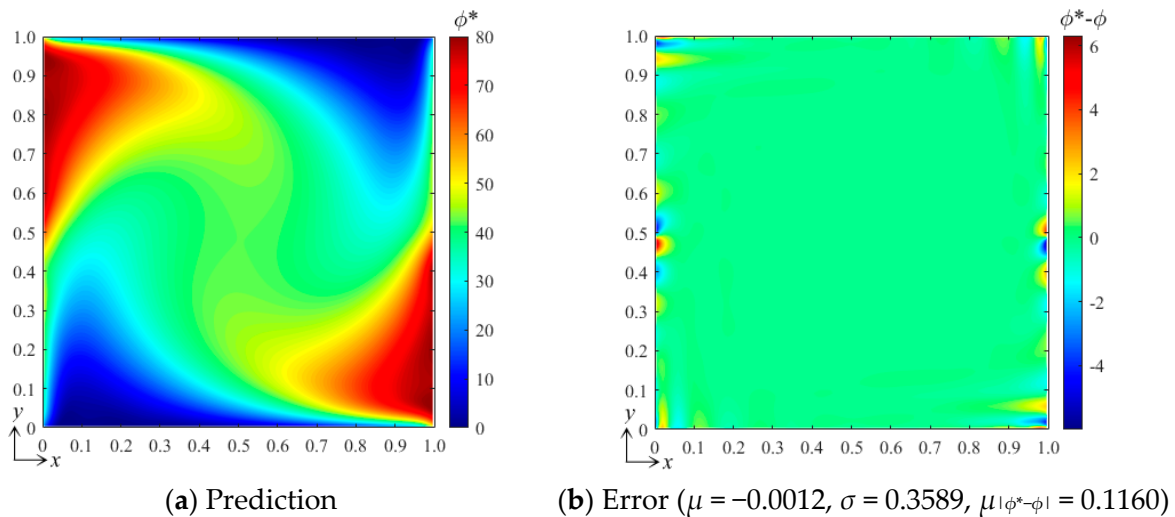
Figure 2. Configurations of the numerical experiments.

For the preliminary tests and verification, we generated a data set with 100 samples to train the neural network, where the input control vectors  $\mathbf{p}$  were randomly generated and  $B_1, B_2 \in [0, 100]$ . By computing the DSSA physical fields numerically, 100 corresponding field data  $\phi(x, y)$  can be obtained, which yields 100 corresponding weight vectors  $\mathbf{a}$  when using the best fit and the selected B-spline function set. Three hidden layers with 8, 10, and 12 nodes, respectively, and one output layer were configured for the fully connected neural network, while two S-shaped tangent functions and two pure linear functions were selected for the corresponding layers. After the network was trained, we verified the bBSNN method via predicting a case with  $[B_1 \ B_2]^T = [80 \ 50]^T$ , which is a new sample that has not appeared in the training data set. Figure 3 shows the ground truth and predicted fields of this case, as well as the error between them. To evaluate the error field, we calculated the averaged value  $\mu$ , standard deviation  $\sigma$ , and the averaged absolute error  $\mu_{|\phi^*-\phi|}$ . When comparing Figure 3a with Figure 3b, they are shown to be extremely similar. Figure 3c shows the error field, and it can be seen that most of the errors are small, except those reaching about 10 at the corners or close to the boundaries. The average error  $\mu = 0.0131$ , the standard deviation  $\sigma = 1.4080$ , and the averaged absolute error  $\mu_{|\phi^*-\phi|} = 0.6952$ . This quantitatively indicates that the agreement between the prediction and the ground truth is good.



**Figure 3.** (a) Ground-truth steady-state field obtained by computing the DSSA physical fields using the finite volume method on a grid resolution of  $128 \times 128$ ; (b) result predicted by the bBSNN with second-order  $10 \times 10$  functions; (c) the error field between the ground truth and the prediction, where  $\mu$ ,  $\sigma$ , and  $\mu_{|\phi^*-\phi|}$  are the averaged error, standard deviation, and the averaged absolute error, respectively.

Qualitatively, it can be seen from Figure 3 that where the gradient is large, the error is likely to be large. To further reduce the error, we added more B-splines to the function set to increase the resolution of the high-frequency components of the field when finding the best fitting. Figure 4a shows the prediction results when using the B-spline function set with  $k_x = k_y = 3$  and  $m_x = m_y = 15$ . The error field in Figure 4b shows that the averaged error  $\mu = -0.0012$ , the standard deviation  $\sigma = 0.3589$ , the averaged absolute error  $\mu_{|\phi^*-\phi|} = 0.1160$ , and the maximum error is about 6. This indicates that the error can be greatly reduced by increasing the number of functions of the selected B-spline set. However, the computational cost and complexity of the training are also greatly increased.



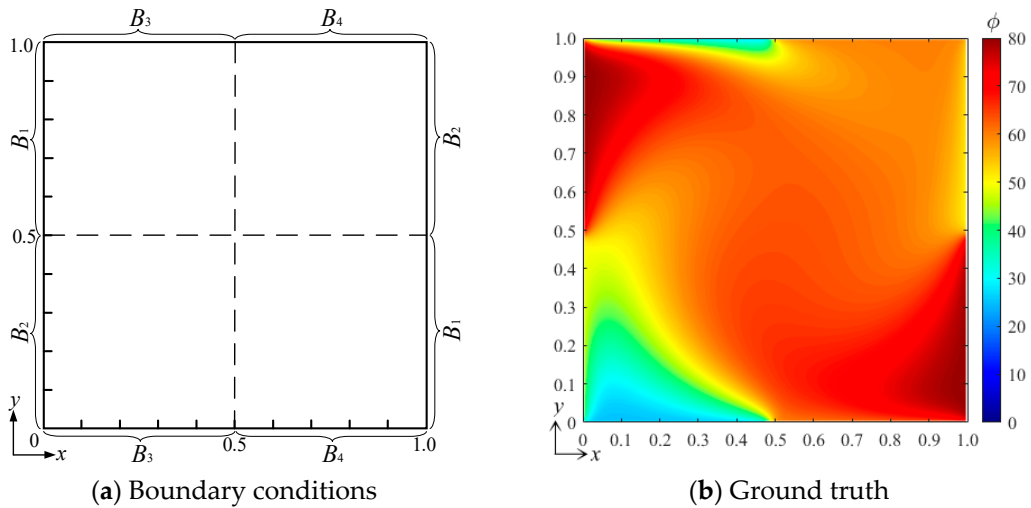
**Figure 4.** (a) The result predicted by the bBSNN with the second-order  $15 \times 15$  functions and (b) the error field.

### 3.2. Effect of the Size of the Input Control Vectors

To examine the performance of the bBSNN method when predicting more complicated DSSA physical fields, we add more independent and variable boundary conditions to expand the size of the input control vectors. In this subsection, two numerical experiments are carried out, where the configuration of the selected B-spline function set is still so that  $k_x = k_y = 2$  and  $m_x = m_y = 10$ , while the independent and variable boundary conditions are increased to 4 and 6, respectively. The data sets of these two numerical experiments are given 150 and 200 random samples, respectively, with the control vector  $B_i \in [0, 100]$ . For the architecture of the fully connected neural network, we only change the nodes of the input layer to match the size of the control vector and keep the hidden layers, the output layer, and the activation functions the same as those in Section 3.1.

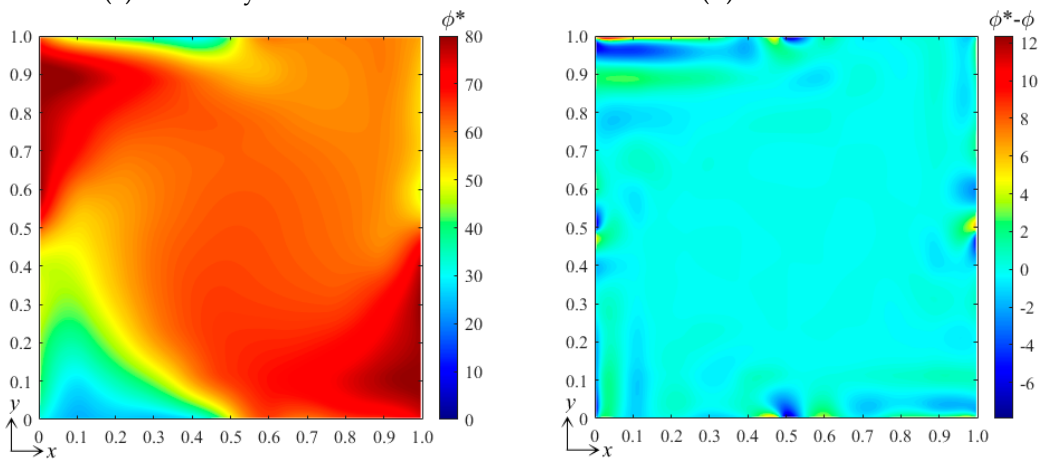
Figure 5a shows the configuration of the numerical experiment with four independent boundary conditions, and the DSSA physical fields with the same parameters used in Section 3.1 were still computed at a grid resolution of  $128 \times 128$ , obtaining the ground-truth field under a tested condition  $[B_1 \ B_2 \ B_3 \ B_4]^T = [80 \ 50 \ 25 \ 60]^T$ , as shown in Figure 5b. After the neural network was well trained, the predicted field obtained under the same conditions by the bBSNN is shown in Figure 5c, which is also very close to the ground-truth field. Figure 5d shows the error field, where  $\mu = -0.0061$ ,  $\sigma = 0.8270$ ,  $\mu_{|\phi^*-\phi|} = 0.3991$ , and the maximum error is about 12. Compared with the experiment with two independent boundary conditions, the characteristics of the error field are similar, while the error statistics are even smaller. This is likely due to the smaller field gradient under the tested condition.

Similarly, Figure 6 shows the configuration of the numerical experiment with six independent boundary conditions and the ground-truth field, as well as that predicted under a tested boundary condition  $[B_1 \ B_2 \ B_3 \ B_4 \ B_5 \ B_6]^T = [80 \ 50 \ 25 \ 60 \ 100 \ 40]^T$ . Figure 6d shows the error field, where  $\mu = -0.0138$ ,  $\sigma = 0.9471$ ,  $\mu_{|\phi^*-\phi|} = 0.4792$ . Compared with the experiment with four independent boundary conditions, the error slightly increases, while it is still less than that of the experiment with two. These numerical experiments indicate that the size of the input control vector has little effect on the accuracy of the bBSNN in predicting the steady-state DSSA physical fields, and the error variations are mainly due to the differences in the local field gradients.



(a) Boundary conditions

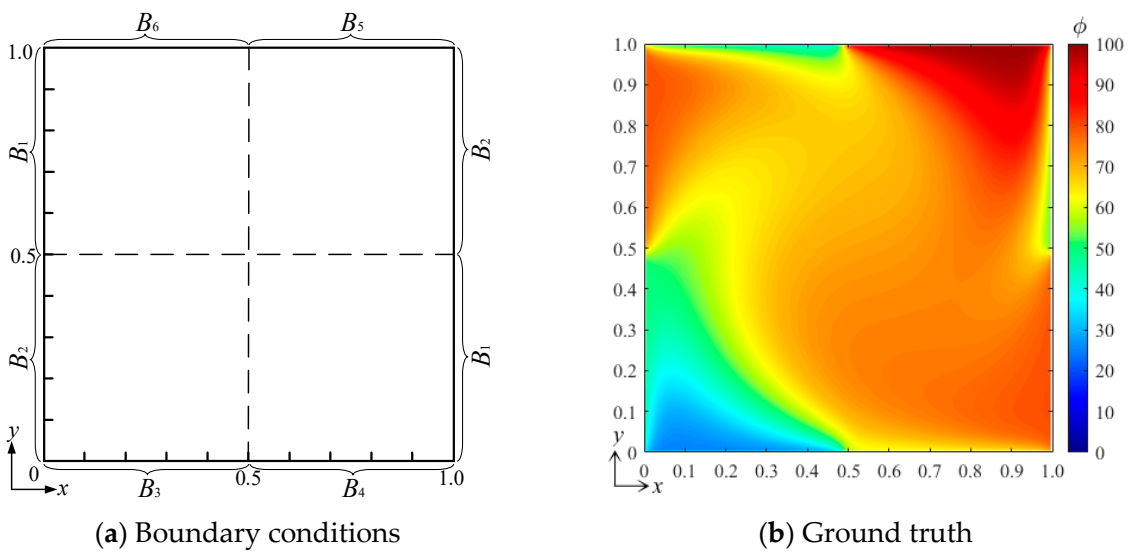
(b) Ground truth



(c) Prediction

(d) Error ( $\mu = 0.0061, \sigma = 0.8270, \mu_{|\phi^*-\phi|} = 0.3991$ )

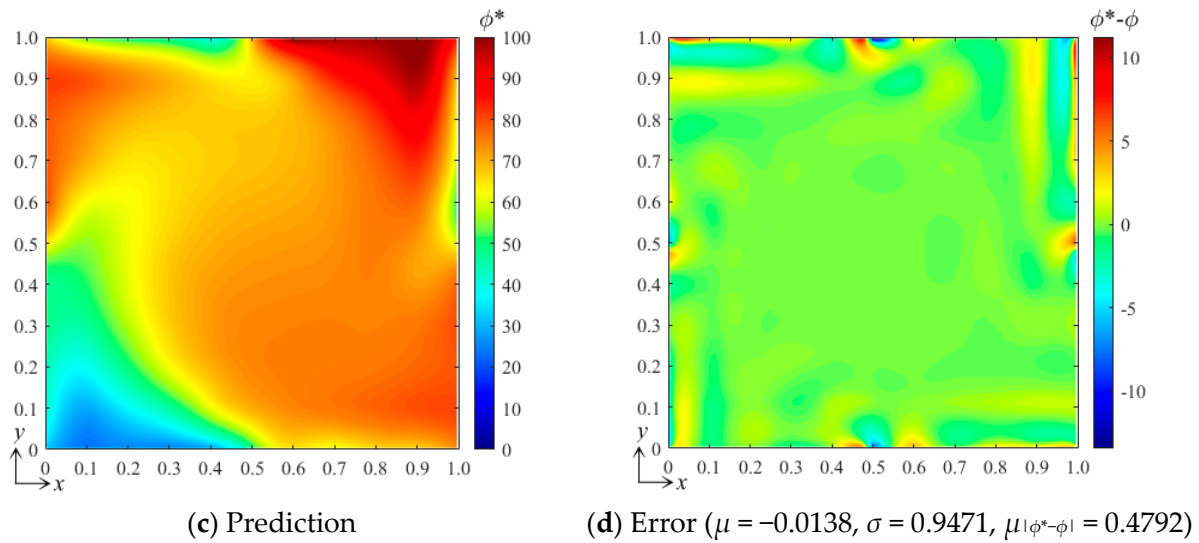
**Figure 5.** (a) The configuration of the numerical experiment with four independent boundary conditions, (b) the ground-truth field under the tested conditions  $[B_1 \ B_2 \ B_3 \ B_4]^T = [80 \ 50 \ 25 \ 60]^T$ , (c) the predicted field, and (d) the error field of steady-state DSSA physical fields.



(a) Boundary conditions

(b) Ground truth

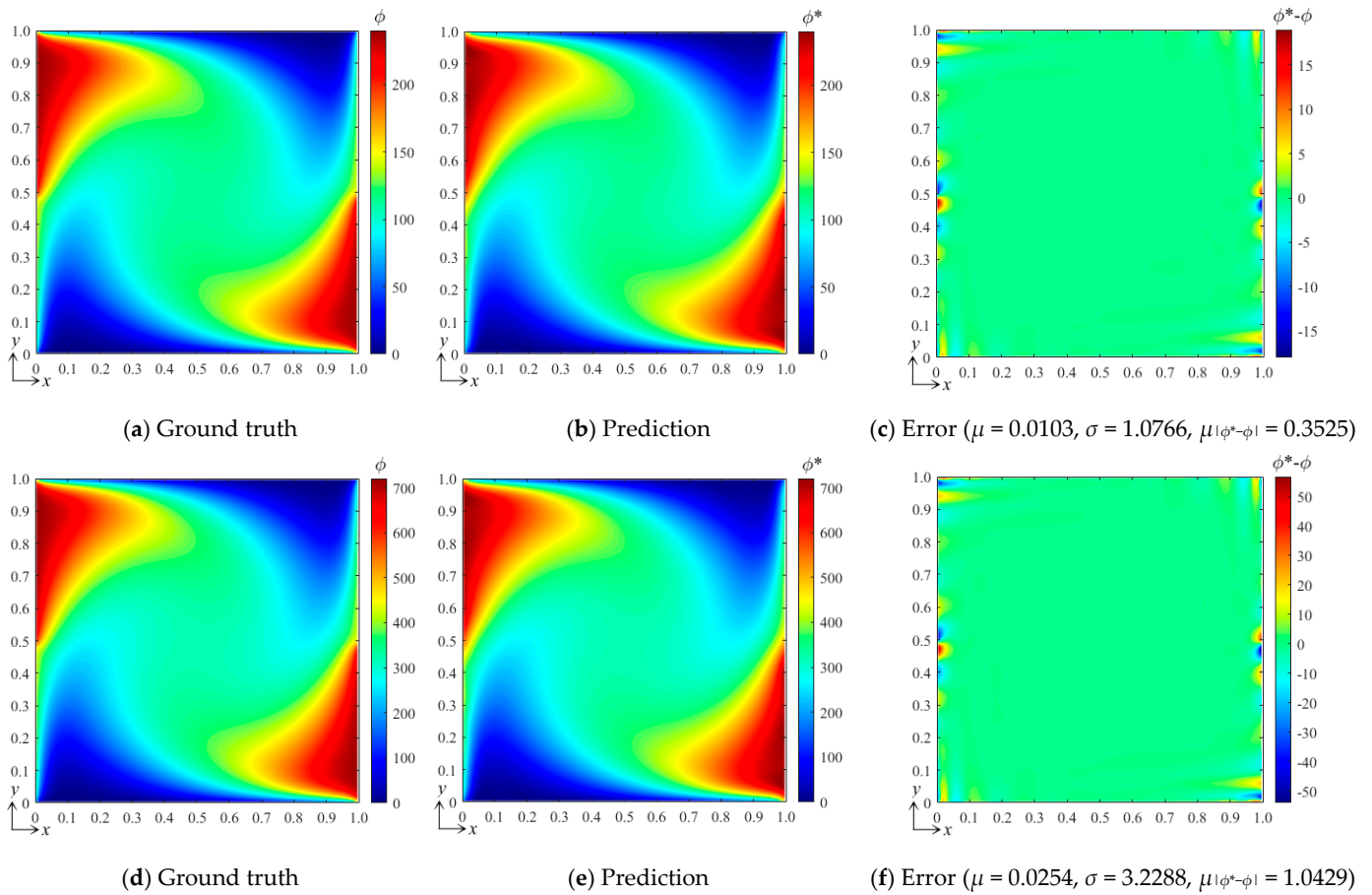
**Figure 6.** Cont.



**Figure 6.** (a) The configuration of the numerical experiment with six independent boundary conditions, (b) the ground-truth field under the tested conditions  $[B_1 B_2 B_3 B_4 B_5 B_6]^T = [80 50 25 60 100 40]^T$ , (c) the predicted field, and (d) the error field of a steady-state DSSA physical fields.

### 3.3. Effect of the Field Gradient

The aforementioned numerical experiments also imply a potential great influence from the field gradient on the accuracy of the bBSNN method. To further examine the effect of the field gradient, we change the value range of the boundary conditions to generate two data sets with greatly different field gradients in the physical fields with two independent boundary conditions, where  $B_1, B_2 \in [0, 300]$  and  $B_1, B_2 \in [0, 900]$ , respectively. Additionally, the configuration of the domain, the selected B-spline function set ( $k_x = k_y = 3$  and  $m_x = m_y = 15$ ), and the architecture of the fully connected neural network are the same as those used in the second test in Section 3.1. For these two data sets, we randomly generated 150 samples in their corresponding ranges. After these two networks were trained, the cases with conditions  $[B_1 B_2]^T = [240 150]^T$  and  $[720 450]^T$  were predicted, respectively. Figure 7a–c show the ground-truth field, predicted field, and the error field under the tested condition  $[B_1 B_2]^T = [240 150]^T$ , respectively, where  $\mu = 0.0103$ ,  $\sigma = 1.0766$ , and  $\mu_{|\phi^*-\phi|} = 0.3525$ , and the maximum error is about 18. Figure 7d–f show the corresponding results under the tested conditions  $[B_1 B_2]^T = [720 450]^T$ , where  $\mu = 0.0254$ ,  $\sigma = 3.2288$  and  $\mu_{|\phi^*-\phi|} = 1.0429$ , and the maximum error is about 55. It can be seen that the error fields of these two cases are fairly similar except for their exact values. These results indicate that the prediction error of the bBSNN method is sensitive to the local field gradient. Specifically, comparing these two cases and the second case in Section 3.1, where  $B_1, B_2 \in [0, 100]$ , we can further conclude that the error of the bBSNN when predicting the steady-state DSSA physical fields is approximately proportional to the field gradient.

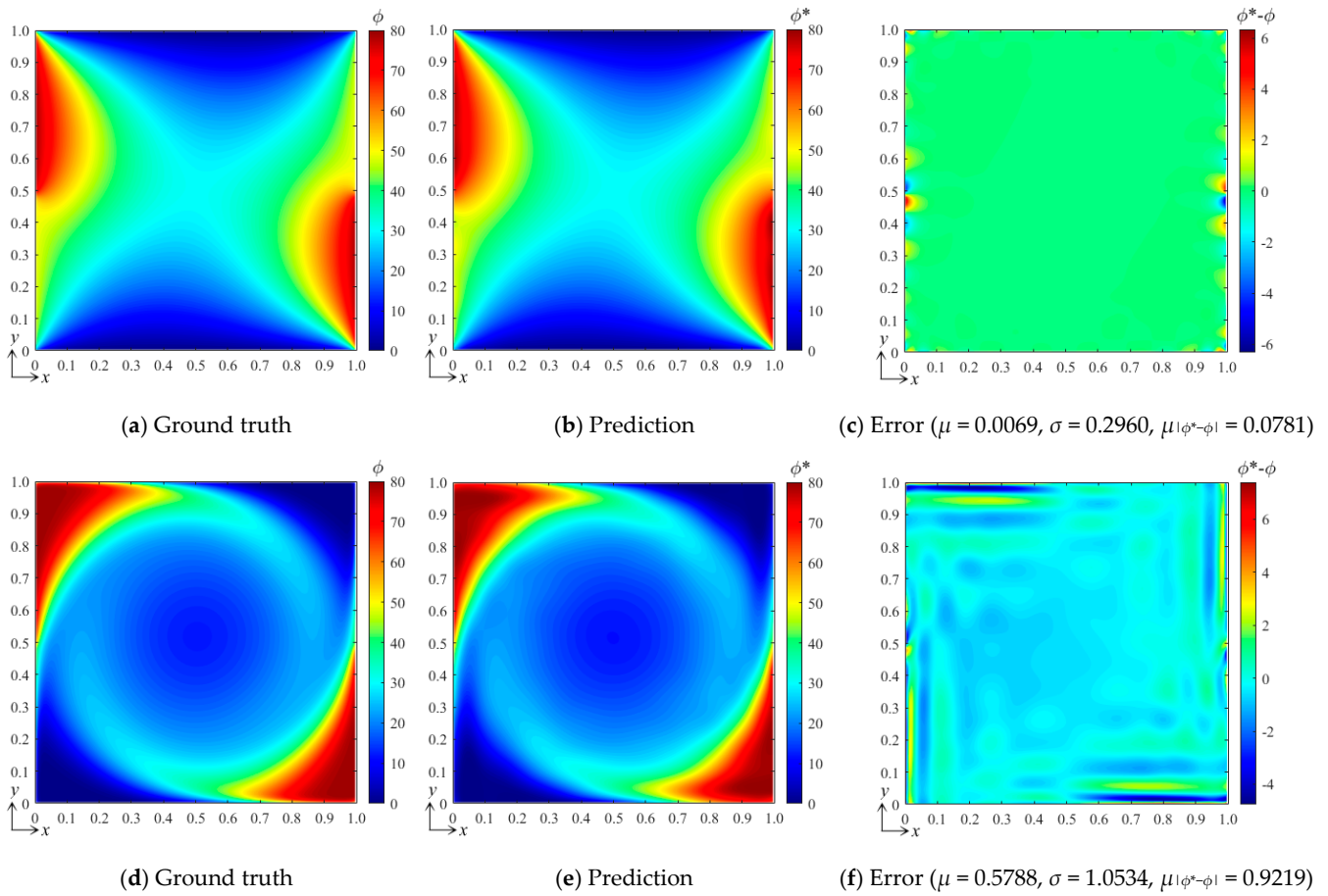


**Figure 7.** (a) The ground-truth field under the tested condition  $[B_1 \ B_2]^T = [240 \ 150]^T$ , (b) the predicted field by the bBSNN trained with a data set generated from  $B_1$  and  $B_2 \in [0, 300]$ , and (c) the error field; (d) the ground-truth, (e) predicted, and (f) error fields under the tested condition  $[B_1 \ B_2]^T = [720 \ 450]^T$  and  $B_1, B_2 \in [0, 900]$ .

### 3.4. Effect of the Field State

In this subsection, we further examine the performance of the bBSNN in predicting the physical fields with various field states, which are governed by the coefficients  $a, b, c,$  and  $d$  of Equation (13). Here, we carry out two representative numerical experiments: (1)  $a = 1, b = 1, c = 1,$  and  $d = 1,$  yielding the system parameters  $\alpha = 1, Pe = 1,$  and  $My = 1$ ; (2)  $a = 1, b = 100, c = 10,$  and  $d = 10,$  yielding the system parameters  $\alpha = 0.01, Pe = 100, My = 10$ . The system parameters indicate that the conditions are moderate for the first experiment while they are much more severe for the second one. Specifically, the second experiment will suffer from a much lower diffusion rate alongside much stronger advection and absorption effects, resulting in much higher local field gradients. Additionally, the domain configuration, B-spline function set ( $k_x = k_y = 3$  and  $m_x = m_y = 15$ ), and the architecture of the fully connected neural network are kept the same as those used in the second test in Section 3.1, where 100 samples are used for each data set to train the network. The tested conditions for these two numerical experiments are both  $[B_1 \ B_2]^T = [80 \ 50]^T$ . Their ground-truth and predicted results, as well as the error fields, are shown in Figure 8. It can be seen that the pattern agreements between the predicted and the ground-truth fields for these two experiments are very good; however, the patterns of the error field are quite different, although the maximum errors are very close to each other. We calculated the error statistics for these:  $\mu = 0.0069, \sigma = 0.2960, \mu_{|\phi^* - \phi|} = 0.0781$  for the first one and  $\mu = 0.5788, \sigma = 1.0534, \mu_{|\phi^* - \phi|} = 0.9219$  for the second one. This indicates that the error ranges for these two experiments with different field states are similar, while the error statistics of the experiment with severe conditions are much larger than those of the one with moderate conditions.





**Figure 8.** (a) The ground-truth field, (b) the predicted field, and (c) the error field of the numerical experiment with  $a = 1, b = 1, c = 1,$  and  $d = 1,$  leading to  $\alpha = 1, Pe = 1,$  and  $My = 1$  for the DSSA physical fields; (d) the ground-truth, (e) predicted, and (f) error fields of the numerical experiment with  $a = 1, b = 100, c = 10,$  and  $d = 10,$  leading to  $\alpha = 0.01, Pe = 100,$  and  $My = 10.$

### 3.5. Comparison with Analytical Solutions

The general form of the two-dimensional diffusion equation is as follows:

$$\frac{\partial \phi}{\partial t} = D \left( \frac{\partial^2 \phi}{\partial x^2} + \frac{\partial^2 \phi}{\partial y^2} \right) \tag{15}$$

where  $\Phi$  is the scalar field and  $D$  is the diffusion coefficient. If the above-mentioned diffusion equation has an analytical solution, it usually has specific boundary conditions and initial conditions. Therefore, we assume a case where there is an instantaneous point diffusion source in the region, as shown in Figure 3, and the initial conditions are as follows:

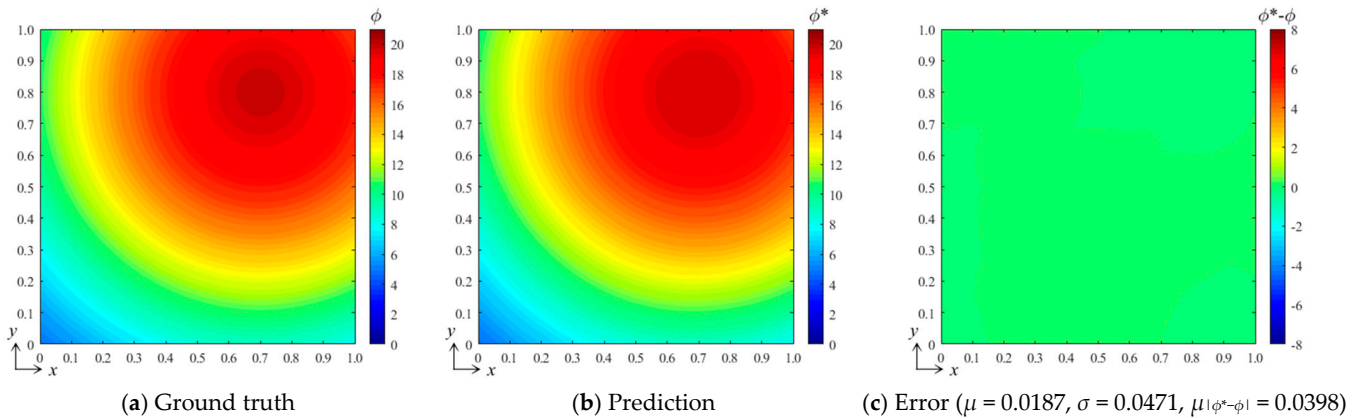
$$\phi(x, y, 0) = 50\delta(x - x_0, y - y_0) \tag{16}$$

Here,  $\delta$  is the Dirichlet function. Under such conditions, there exists an analytical solution for Equation (15), as follows:

$$\phi(x, y, t) = \frac{50}{4\pi Dt} e^{-\frac{(x-x_0)^2+(y-y_0)^2}{4Dt}} \tag{17}$$

Set  $D = 0.2,$  take the field distribution at  $t = 1$  s as the target, take any point in the region shown in Figure 3 as the diffusion source, and record its coordinates  $(x_0, y_0)$  as the control vector. Randomly generate 100 sets of control vectors and their corresponding field distributions as training data sets for the bBSNN, where the network structure and parameter

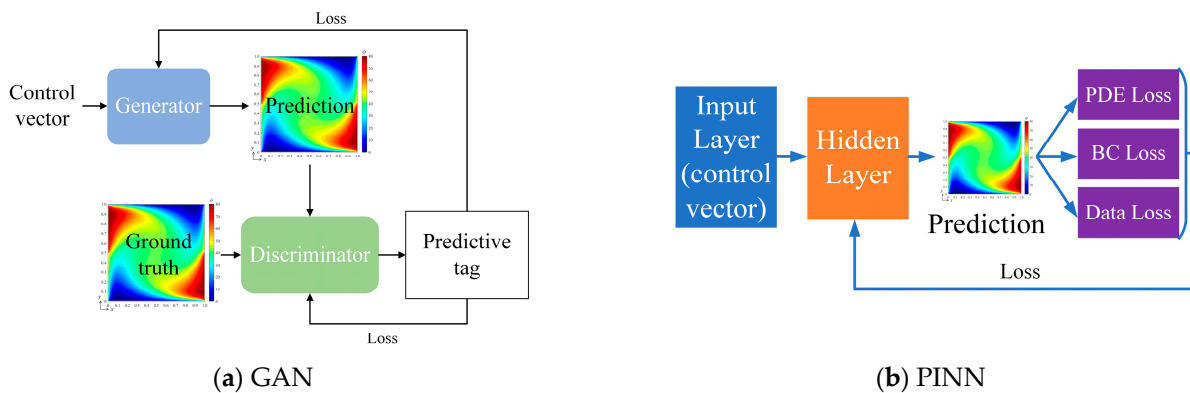
settings are the same as in Section 3.1. Finally, the prediction effect is verified with the diffusion source coordinates of (0.7, 0.8) and the verification results are shown in Figure 9. It can be seen from Figure 9a,b that the bBSNN prediction performance is good, the ground truth and prediction are extremely similar, and the error distribution is very average. The average error  $\mu = 0.0187$ , the standard deviation  $\sigma = 0.0471$ , and the averaged absolute error  $\mu_{|\phi^*-\phi|} = 0.0398$ . This shows that the bBSNN also shows great potential for predicting the diffusion field distribution with analytical solutions.



**Figure 9.** (a) Ground truth obtained by computing the diffusion fields using the analytical solutions; (b) the result predicted by the bBSNN with the third-order  $15 \times 15$  functions; (c) the error field between the ground truth and the prediction, where  $\mu$ ,  $\sigma$ , and  $\mu_{|\phi^*-\phi|}$  are the averaged error, standard deviation, and the averaged absolute error, respectively.

3.6. Method Comparison

To further demonstrate the effectiveness of bBSNN in predicting diffusion–advection–absorption–source physical fields, we compare it with GAN [18] and PINN [25]. The network structure of GAN is set up as shown in Figure 10a, mainly comprising a generator and discriminator. The generator is composed of seven transposed convolutional layers, and the discriminator is composed of six convolutional layers and one fully connected layer. The network structure of PINN is shown in Figure 10b, where the loss function includes three parts: partial differential structure loss (PDE loss), boundary value condition loss (BC loss), and real data condition loss (data loss), and the hidden layer is composed of six fully connected layers. The numerical example presented in Section 3.1 is used to train the GAN and PINN, and the boundary condition of the test group is also set as  $[B_1 \ B_2]^T = [80 \ 50]^T$ .

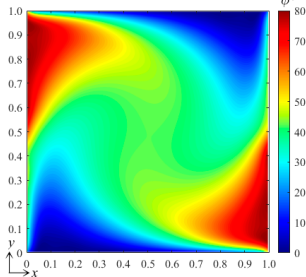
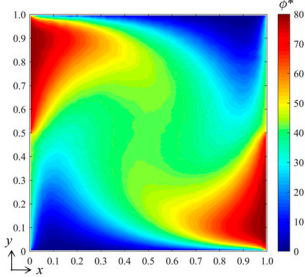
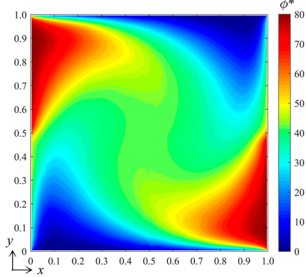
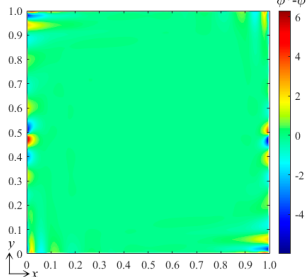
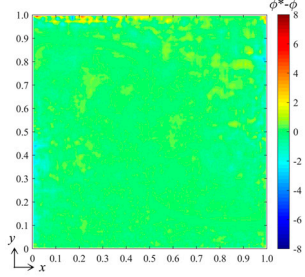
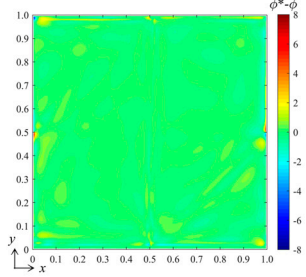


**Figure 10.** Schematics of the architectures of the used GAN and PINN.

All the numerical cases were computed on PYTHON 3.6 with PYTORCH 1.10 and a computer with the configurations of Intel Xeon E5-2678@2.50 GHz and an RAM of 128 GB. To compare the performances of the bBSNN, GAN, and PINN, we summarized the training

data set, training time, predicted results, and errors, as listed in Table 1. Notably, for the presented numerical problem, the bBSNN can be well trained using a data set with 150 images; however, the GAN cannot provide a converged result when the training data set is smaller than 1500, and we thus trained the GAN using a data set with 2000 images. Comparing their training times, the bBSNN has a much shorter training time than the GAN and PINN. Comparing their error maps, both the absolute averaged error  $|\mu|$  and the averaged absolute error  $\mu_{|\phi^*-\phi|}$  of the bBSNN are the smallest. These indicate that the training efficiency and the prediction accuracy of bBSNN are both much better than those of GAN and PINN, at least for the presented PDE problem.

**Table 1.** The comparison of bBSNN, GAN, and PINN.

	bBSNN	GAN	PINN
Training data set	150	2000 (<1500 non-converged)	-
Training time	5 min	290 min	15 min
Prediction			
Error map			
Error	$\mu = -0.0012, \sigma = 0.3589,$ $\mu_{ \phi^*-\phi } = 0.1160$	$\mu = 0.0028, \sigma = 0.3860,$ $\mu_{ \phi^*-\phi } = 0.2298$	$\mu = 0.0498, \sigma = 0.2039,$ $\mu_{ \phi^*-\phi } = 0.3451$

### 4. Conclusions

This work introduced a best-fitting B-spline neural network, which combines a fully connected neural network and a best-fitting B-spline function set. Especially, we carried out a numerical study on the bBSNN method predicting steady-state advection–diffusion physical fields with absorption and source terms, from which the performance of the bBSNN was verified, and the effects of the physical fields and B-spline function set on the error field were investigated and discussed in detail. The numerical experiments indicated that the bBSNN method could predict the physical fields very well. From the error analysis of the numerical experiments, we can conclude the following:

- (1) The error of the bBSNN is sensitive to the local field gradient, and where the gradient is higher, the error is likely larger, almost showing a proportional relationship. This can be understood as meaning that, in information theory, when a signal or piece of data has certain characteristics, its information entropy may change. There is a certain correspondence between the sensitivity of the error and the variability of information entropy.
- (2) The effect of the field state slightly affects the range of the error field while greatly increasing the error statistics.
- (3) The prediction error of the bBSNN method can be reduced by increasing the order or the number of the B-splines in the selected function set.

- (4) The data set used to train the bBSNN can be very small, even for the case with six independent boundary conditions, the bBSNN trained by a data set with only 200 random samples can yield a good predicted field, and the training efficiency is also very high.
- (5) Compared with GAN and PINN, bBSNN presents obvious advantages in terms of training efficiency and prediction accuracy.

Consequently, we believe the bBSNN method can be used as a good surrogate model to predict advection–diffusion physical fields under relatively complex conditions, and could also be used to accelerate the numerical computation of such physical fields. Although this work is carried out for steady-state advection–diffusion physical fields, the bBSNN method, if combined with a recurrent neural network, may have great potential to deal with transient physical fields, which could be our next step in the near future.

**Supplementary Materials:** The following supporting information can be downloaded at: <https://www.mdpi.com/article/10.3390/e26070577/s1>, Code: FVM&bBSNN\_code.

**Author Contributions:** Methodology, X.Z. and L.T.; Validation, X.Z.; Formal analysis, X.Z. and S.H.; Resources, F.G.; Writing—original draft, X.Z.; Writing—review & editing, X.Z. and X.A.; Funding acquisition, J.L. and X.A. All authors have read and agreed to the published version of the manuscript.

**Funding:** This research was funded by the National Natural Science Foundation of China (No. 52105504) and the National Key Research and Development Program of China (No. 2022YFB3403800).

**Institutional Review Board Statement:** Not applicable.

**Data Availability Statement:** Data are contained within the article and supplementary materials.

**Conflicts of Interest:** The authors declare that there are no conflicts of interest.

## References

1. Nishida, K.; Yoshida, S.; Shiozawa, S. Numerical model to predict water temperature distribution in a paddy rice field. *Agric. Water Manag.* **2021**, *245*, 106553. [[CrossRef](#)]
2. Yang, F.; Wu, T.; Jiang, H.; Jiang, J.; Hao, H.; Zhang, L. A new method for transformer hot-spot temperature prediction based on dynamic mode decomposition. *Case Stud. Therm. Eng.* **2022**, *37*, 102268. [[CrossRef](#)]
3. Eivazi, H.; Veisi, H.; Naderi, M.H.; Esfahanian, V. Deep neural networks for nonlinear model order reduction of unsteady flows. *Phys. Fluids* **2020**, *32*, 105104. [[CrossRef](#)]
4. Naderi, M.H.; Eivazi, H.; Esfahanian, V. New method for dynamic mode decomposition of flows over moving structures based on machine learning (hybrid dynamic mode decomposition). *Phys. Fluids* **2019**, *31*, 127102. [[CrossRef](#)]
5. Mazumder, S. Comparative assessment of the finite difference, finite element, and finite volume methods for a benchmark one-dimensional steady-state heat conduction problem. *J. Heat Transf.* **2017**, *139*, 071301. [[CrossRef](#)]
6. Chen, Z.; Wang, G.; Chen, H.; Sun, S. Direct estimation of transient temperature field of heat transfer system based on mapping characteristics fuzzy clustering. *Int. J. Heat Mass Transf.* **2022**, *190*, 122787. [[CrossRef](#)]
7. Sekar, V.; Khoo, B.C. Fast flow field prediction over airfoils using deep learning approach. *Phys. Fluids* **2019**, *31*, 57103. [[CrossRef](#)]
8. Michoski, C.; Milosavljević, M.; Oliver, T.; Hatch, D.R. Solving differential equations using deep neural networks. *Neurocomputing* **2020**, *399*, 193–212. [[CrossRef](#)]
9. Coelho, L.S.; Guerra, F.A. B-spline neural network design using improved differential evolution for identification of an experimental nonlinear process. *Appl. Soft Comput.* **2008**, *8*, 1513–1522. [[CrossRef](#)]
10. Gálvez, A.; Iglesias, A.; Puig-Pey, J. Iterative two-step genetic-algorithm-based method for efficient polynomial B-spline surface reconstruction. *Inf. Sci.* **2012**, *182*, 56–76. [[CrossRef](#)]
11. Coelho, L.S.; Pessoa, M.W. Nonlinear identification using a B-spline neural network and chaotic immune approaches. *Mech. Syst. Signal Process.* **2009**, *23*, 2418–2434. [[CrossRef](#)]
12. Wang, J.; Chen, B.; Yang, C. Approximation of algebraic and trigonometric polynomials by feedforward neural networks. *Neural Comput. Appl.* **2012**, *21*, 73–80. [[CrossRef](#)]
13. Dokur, Z.; Olmez, T. Heartbeat classification by using a convolutional neural network trained with Walsh functions. *Neural Comput. Appl.* **2020**, *32*, 12515–12534. [[CrossRef](#)]
14. Nakamura, T.; Fukami, K.; Hasegawa, K.; Nabaie, Y.; Fukagata, K. Convolutional neural network and long short-term memory based reduced order surrogate for minimal turbulent channel flow. *Phys. Fluids* **2021**, *33*, 025116. [[CrossRef](#)]
15. Bhatnagar, S.; Afshar, Y.; Pan, S.; Duraisamy, K.; Kaushik, S. Prediction of aerodynamic flow fields using convolutional neural networks. *Comput. Mech.* **2019**, *64*, 525–545. [[CrossRef](#)]



16. Zhang, J.; Zheng, Y.; Sun, J.; Qi, D. Flow prediction in spatio-temporal networks based on multitask deep learning. *IEEE Trans. Knowl. Data Eng.* **2020**, *32*, 468–478. [[CrossRef](#)]
17. Goodfellow, I.; Pouget-Abadie, J.; Mirza, M.; Xu, B.; Warde-Farley, D.; Ozair, S.; Courville, A.; Bengio, Y. Generative adversarial networks. *Commun. ACM* **2020**, *63*, 139–144. [[CrossRef](#)]
18. Mehdi, M.; Simon, O. Conditional generative adversarial nets. *arXiv* **2017**, arXiv:1411.1784.
19. Chen, M.T.; Mahmood, F.; Sweer, J.A.; Durr, N.J. GANPOP: Generative adversarial network prediction of optical properties from single snapshot wide-field images. *IEEE Trans. Med. Imaging* **2020**, *39*, 1988–1999. [[CrossRef](#)]
20. Na, B.; Son, S. Prediction of atmospheric motion vectors around typhoons using generative adversarial network. *J. Wind Eng. Ind. Aerodyn.* **2021**, *214*, 104643. [[CrossRef](#)]
21. Wen, S.; Shen, N.; Zhang, J.; Lan, Y.; Han, J.; Yin, X.; Zhang, Q.; Ge, Y. Single-rotor UAV flow field simulation using generative adversarial networks. *Comput. Electron. Agric.* **2019**, *167*, 105004. [[CrossRef](#)]
22. Tang, H.; Liao, Y.; Yang, H.; Xie, L. A transfer learning-physics informed neural network (TL-PINN) for vortex-induced vibration. *Ocean Eng.* **2022**, *266*, 113101. [[CrossRef](#)]
23. Li, Y.; Mei, F. Deep learning-based method coupled with small sample learning for solving partial differential equations. *Multimed. Tools Appl.* **2021**, *80*, 17391–17413. [[CrossRef](#)]
24. Lu, L.; Meng, X.; Mao, Z.; Karniadakis, G.E. DeepXDE: A deep learning library for solving differential equations. *SIAM Rev.* **2021**, *63*, 208–228. [[CrossRef](#)]
25. Wang, H.; Li, J.; Wang, L.; Liang, L.; Zeng, Z.; Liu, Y. On acoustic fields of complex scatters based on physics-informed neural networks. *Ultrasonics* **2023**, *128*, 106872. [[CrossRef](#)] [[PubMed](#)]
26. Du, Y.F.; Wang, M.Z.; Zaki, T.A. State estimation in minimal turbulent channel flow: A comparative study of 4DVar and PINN. *Int. J. Heat Fluid Flow* **2023**, *99*, 109073. [[CrossRef](#)]
27. Yi, J.; Chen, Z.; Li, D.; Li, J.; Liu, J. Conditional generative adversarial network for welding deformation field prediction of butt-welded plates. *J. Constr. Steel Res.* **2023**, *201*, 107755. [[CrossRef](#)]
28. Pollok, S.; Olden-Jørgensen, N.; Jørgensen, P.S.; Bjørk, R. Magnetic field prediction using generative adversarial networks. *J. Magn. Magn. Mater.* **2023**, *571*, 170556. [[CrossRef](#)]
29. Chen, J.; Zhu, F.; Han, Y.; Chen, C. Fast prediction of complicated temperature field using conditional multi-attention generative adversarial networks (CMAGAN). *Expert Syst. Appl.* **2021**, *186*, 115727. [[CrossRef](#)]
30. Meng, Y.; Rigall, E.; Chen, X.; Gao, F.; Dong, J.; Chen, S. Physics-guided generative adversarial networks for sea subsurface temperature prediction. *IEEE Trans. Neural Netw. Learn. Syst.* **2021**, *34*, 3357–3370. [[CrossRef](#)]
31. Jiang, H.; Nie, Z.; Yeo, R.; Farimani, A.B.; Kara, L.B. Stressgan: A generative deep learning model for two-dimensional stress distribution prediction. *J. Appl. Mech.* **2021**, *88*, 051005. [[CrossRef](#)]
32. Enomoto, K.; Sakurada, K.; Wang, W.; Fukui, H.; Matsuoka, M.; Nakamura, R.; Kawaguchi, N. Filmy cloud removal on satellite imagery with multispectral conditional generative adversarial nets. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition Workshops, Honolulu, HI, USA, 21–26 July 2017; pp. 48–56.
33. Liu, X.; Peng, W.; Gong, Z.; Zhou, W.; Yao, W. Temperature field inversion of heat-source systems via physics-informed neural networks. *Eng. Appl. Artif. Intell.* **2022**, *113*, 104902. [[CrossRef](#)]
34. Zhao, Z.; Stuebner, M.; Lua, J.; Phan, N.; Yan, J. Full-field temperature recovery during water quenching processes via physics-informed machine learning. *J. Mater. Process. Technol.* **2022**, *303*, 117534. [[CrossRef](#)]
35. Xie, J.; Chai, Z.; Xu, L.; Ren, X.; Liu, S.; Chen, X. 3D temperature field prediction in direct energy deposition of metals using physics informed neural network. *Int. J. Adv. Manuf. Technol.* **2022**, *119*, 3449–3468. [[CrossRef](#)]
36. Zhu, Q.M.; Liu, Z.L.; Yan, J.H. Machine learning for metal additive manufacturing: Predicting temperature and melt pool fluid dynamics using physics-informed neural networks. *Comput. Mech.* **2021**, *67*, 619–635. [[CrossRef](#)]
37. Hong, X.; Iplikci, S.; Chen, S.; Warwick, K. A model-based PID controller for Hammerstein systems using B-spline neural networks. *Int. J. Adapt. Control Signal Process.* **2014**, *28*, 412–428. [[CrossRef](#)]
38. Hong, X.; Chen, S. The system identification and control of Hammerstein system using non-uniform rational B-spline neural network and particle swarm optimization. *Neurocomputing* **2012**, *82*, 216–223. [[CrossRef](#)]
39. Folgheraiter, M. A combined B-spline-neural-network and ARX model for online identification of nonlinear dynamic actuation systems. *Neurocomputing* **2016**, *175*, 433–442. [[CrossRef](#)]
40. Deng, H.; Srinivasan, D.; Oruganti, R. A B-spline network based neural controller for power electronic applications. *Neurocomputing* **2010**, *73*, 593–601. [[CrossRef](#)]
41. Zhang, X.; Zhao, Y.; Guo, K.; Li, G.; Deng, N. An adaptive B-spline neural network and its application in terminal sliding mode control for a mobile satcom antenna inertially stabilized platform. *Sensors* **2017**, *17*, 978. [[CrossRef](#)]
42. Cheng, K. Self-structuring fuzzy-neural back stepping control with a B-spline-based compensator. *Neurocomputing* **2013**, *117*, 138–149. [[CrossRef](#)]

**Disclaimer/Publisher’s Note:** The statements, opinions and data contained in all publications are solely those of the individual author(s) and contributor(s) and not of MDPI and/or the editor(s). MDPI and/or the editor(s) disclaim responsibility for any injury to people or property resulting from any ideas, methods, instructions or products referred to in the content.