Applied Soft Computing 11 (2011) 855–860

Contents lists available at ScienceDirect

Applied Soft Computing

journal homepage: www.elsevier.com/locate/asoc

Numerical solution of PDEs via integrated radial basis function networks with adaptive training algorithm

Hong Chen [a,b,*], Li Kong [a], Wen-Jun Leng [b]

[a] Department of Control Science and Engineering, Huazhong University of Science and Technology, Zhong Shan Road #450, Wuhan 430074, China
[b] Wuhan Second Ship Design and Research Institute, Wuhan 430064, China

ARTICLE INFO

Article history:
Received 14 February 2009
Received in revised form 3 January 2010
Accepted 17 January 2010
Available online 25 January 2010

Keywords:
Radial basis functions
Approximation
Partial differential equation
Collocation

ABSTRACT

This paper develops a mesh-free numerical method for solving PDEs, based on integrated radial basis function networks (IRBFNs) with adaptive residual subsampling training scheme. The multiquadratic function is chosen as the transfer function of the neurons. The nonlinear algebraic equation systems for weights training are solved by Levenberg–Marquardt algorithm. The performance of the proposed method is demonstrated in numerical examples by approximating several functions and solving nonlinear PDEs. The result of numerical experiments shows that the IRBFNs with the adaptive procedure requires less neurons to attain the desired accuracy than conventional radial basis function networks.

1. Introduction

Many problems in science and engineering modeled as partial differential equations (PDEs) can so far be solved by numerical methods relying on spatial and temporal discretising. Although mesh-based methods, such as finite difference method (FDM) and finite element method (FEM), have been successfully applied, their accuracy depends critically on mesh quality. In general, only low-order schemes are employed to found a continuous approximation of the function across the mesh but not its derivatives. The discontinuity of the approximation of the derivative can adversely affect the stability of the solution. While higher order schemes could be applied for more accurate approximations of the derivatives, they usually require more computational cost. Therefore, highly refined mesh with a higher density of elements is needed when higher accuracy is required, which also increase the computational cost. Moreover, even in low dimension problems, the difficulty is still hard to overcome when moving boundaries, free surfaces or complex boundaries are encountered.

Neural networks can be applied as a mesh-free approximation schemes. It has proved that radial basis function (RBF) networks with one hidden layer are capable of universal approximation [1]. Since Kansa [2,3] devised a RBFN-based method suitable for solving PDEs, many methods based on RBFNs have been developed so far for approximation and PDEs solving. Studies have showed that the RBFN-based method can be interpreted in the framework of standard pseudospectral method [4]. Some papers have been written to solve partial differential equations using the collocation method with different radial basis functions [5,6]. The choice of RBF and its training scheme are crucial for RBFN-based PDEs solution. Mai-Duy and Tran-Cong devised a integrated multiquadric (MQ) RBFNs method for the approximation of functions and their derivatives [7,8]. The properties of integrated RBFs (IRBFs) were examined by Sarra [9]. RBFs that have been integrated several times appear to be more capable of smoothing the derivative errors, and fewer neurons are needed for approximation the solution of PDEs in IRBF scheme. Numerical experiments and theoretical analysis indicate that for solving PDEs integrated radial basis function (IRBF) procedure is more accurate in comparison with direct RBF (DRBF) procedure. Numerical experiments for a range of PDEs show that the IRBF scheme is also more stable than DRBF [8,10].

Both in DRBF and IRBF, the accuracy of the solutions and conditioning of the collocation matrix are affected by the shape parameters, and the computational costs of the training process largely rely upon the numbers and the locations of neurons. The optimal choice of the shape parameters, as well as the number and location of neurons, is still an open problem. Gonzalez et al. [12] proposed an evolutionary optimisation method to found the size, shape, and position parameter of RBF network for function approximation. Sarra [13] proposed an adaptive radial basis function methods for time-dependent partial differential equations. In [14], Driscoll and Heryudono suggested a direct RBF adaptive algo-

* Corresponding author at: Department of Control Science and Engineering, Huazhong University of Science and Technology, Zhong Shan Road #450, Wuhan 430074, China.
E-mail address: chenhong.paper@gmail.com (H. Chen).

1568-4946/$ – see front matter © 2010 Elsevier B.V. All rights reserved.
doi:10.1016/j.asoc.2010.01.005

rithm in which neurons are added or removed based on residuals evaluated at a finer point set, and the shape parameters of RBF neurons are also adapted based on the distances between centres to prevent the growth of the conditioning of the approximation matrix. This algorithm is attractive for its simplicity and effectiveness, and adopted in this work for training the integrated radial basis function networks.

This paper present a novel method for approximation and solving PDEs, based on integrated radial basis function networks with adaptive residual subsampling training method. The rest of present paper is organised as follows. Brief explanation of the IRBFNs for approximation and solving PDEs are given in Section 2. Section 3 describes the adaptive residual subsampling training procedure. Several numerical examples are presented in Section 4, followed by a conclusion summary in Section 5.

2. Integrated radial basis function network for approximation and solving PDEs

2.1. Radial basis function networks and approximation

Radial basis function networks often consist of two layers: a hidden radial basis function neural layer and an output linear layer. The output of RBF network with N neuron is given by

$$F(x) = \sum_{j=1}^{N} w_j \phi(||x - \xi_j||_2), \tag{1}$$

where the transfer function of the hidden neurons, ϕ, is a radial basis function, $||.||_2$ denotes the Euclidean distance between two points, ξ_i are the location of the centres collocations, and w_j are the input weights of the output linear layer. There are several types of RBF. In this work we are interested in the popular multiquadric (MQ) RBF, which is given by

$$\phi(r) = \sqrt{1 + (\varepsilon r)^2}, \tag{2}$$

where ε is the shape parameter of the radial basis function. MQs respond rather sensitively to ε. The limit $\varepsilon \to \infty$ produces piecewise linear approximation, and the limit $\varepsilon \to 0$ produces global polynomial approximation [5]. Given scalar function values $f_i = f(x_i)$, the weights, w_j, are obtained by solving a system of linear equation:

$$A \begin{bmatrix} w_1 \\ \vdots \\ w_N \end{bmatrix} = \begin{bmatrix} f_1 \\ \vdots \\ f_N \end{bmatrix}, \tag{3}$$

where the interpolation matrix A satisfies $a_{ij} = \phi(||x_i - x_j||)$. Under certain conditions, the infinitely smooth radial functions exhibit exponential or spectral convergence as a function of centre spacing. With the model F decomposed into N fixed radial basis functions, the unknown weights $\{w_j\}_{j=1\ldots N}$ can be found by $W = A^{-1}F$, or more generally, via linear least-squares methods, by minimising the sum squared error:

$$SSE = \sum_{j=1}^{N} \left(\sum_{j=1}^{N} w_j \phi(||x - \xi_j||_2) - f(x_j) \right)^2, \tag{4}$$

with respect to the weights, w_j, of F.

2.2. Radial basis function networks for solving differential equations

Numerical solution of differential equations is intimately connected with approximating function and its derivatives [7]. The solution F and its derivatives can be approximated in terms of a set of radial basis functions. Given (1), the derivatives of the function $F(x)$ are calculated by

$$\frac{\partial^k f_j}{\partial x_m \cdots \partial x_n} = \sum_{j=1}^{N} w_j \frac{\partial^k \phi_j}{\partial x_m \cdots \partial x_n}, \tag{5}$$

$$\frac{\partial F}{\partial x} = \sum_{j=1}^{N} w_j h_j(x), \tag{6}$$

$$\frac{\partial^2 F}{\partial x^2} = \sum_{j=1}^{N} w_j \tilde{h}_j(x), \tag{7}$$

where

$$h_j(x) = \frac{\partial \phi_j}{\partial x} = \frac{\varepsilon^2 (x - \xi_j)}{\sqrt{1 + (\varepsilon r)^2}}, \tag{8}$$

$$\tilde{h}_j(x) = \frac{\partial h_j}{\partial x} = \frac{\varepsilon^2}{(1 + (\varepsilon r)^2)^{\frac{3}{2}}}, \tag{9}$$

Consider the following partial differential equations over the domain Ω:

$$\mathcal{L}\psi(x) = g(x), \quad in \quad \Omega, \tag{10}$$

$$\psi(r) = p_1(x), \quad on \quad \partial\Omega_1, \tag{11}$$

$$n \cdot \nabla\psi = p_2(x), \quad on \quad \partial\Omega_2, \tag{12}$$

where $\mathcal{L}$ is a differential operator, $g(x)$ is an known function, $\partial\Omega_1$ and $\partial\Omega_2$ are the Dirichlet boundaries and the Neuman boundaries of the domain, respectively, $\partial\Omega_1 \cap \partial\Omega_2 = \Phi$, $\partial\Omega_1 \cup \partial\Omega_2 = \partial\Omega$, n is the outward unit normal, ∇ is the gradient operator, p_1 and p_2 are known functions of x. To obtain an approximate solution of PDE, a set of collocation in the analysis domain can be adopted to act as neuron centres.

A equation system respect to w_j depend on the PDE can be derived by operating $\mathcal{L}$ on matrix A directly. For example, given a set of PDE with second order in Poisson's equation, by substituting (10)–(12) by (5)–(9), we can get a linear equation system similar to (3) directly. A augmented matrix is obtained by replacing the interpolation matrix A in (3) together with the boundary conditions. The right side of (3) is also replaced by a vector augmented by $p_1(x)$ and $p_2(x)$. The deferential equation and boundary conditions are forced to satisfy. The new linear equation system:

$$\begin{bmatrix} \mathcal{L}A \\ \cdots \\ A \\ \cdots \\ h \end{bmatrix} \begin{bmatrix} w_1 \\ \vdots \\ w_N \end{bmatrix} = \begin{bmatrix} g \\ \cdots \\ p_1 \\ \cdots \\ p_2 \end{bmatrix}, \tag{13}$$

is solved by linear least-square programming which is available in many software, and the unknown weights of the RBF network are obtained by minimising the sum square error.

2.3. Integrated radial basis function networks for solving differential equations

In the integrated radial basis function networks, the formulation of the problem starts with the RBF decomposition of the highest order derivative of the function. The derivative expansion then integrated to yield an expression for the solution of the differential equation.

In the case of functions with up to second derivatives, the relevant expressions of IRBF networks are

$$\frac{\partial^2 F}{\partial x^2} = \sum_{j=1}^{N} w_j \phi_j(x) = \sum_{j=1}^{N} w_j \sqrt{1+(\varepsilon r)^2}, \tag{14}$$

$$\frac{\partial F}{\partial x} = \sum_{j=1}^{N} w_j H_j(x) + C_1, \tag{15}$$

$$F(x) = \sum_{j=1}^{N} w_j \tilde{H}_j(x) + C_1 x_j + C_2, \tag{16}$$

where C_1 and C_2 are functions of independent variables other than x_j and

$$H_j(x) = \int \phi_j(x) dx_m = \frac{\varepsilon(x_m - \xi_m)\sqrt{1+(\varepsilon r)^2} + \sinh^{-1}(\varepsilon r)}{2\varepsilon}, \tag{17}$$

$$\tilde{H}_j(x) = \int H_j(x) dx_m = \frac{(-2+(\varepsilon r)^2)\sqrt{1+(\varepsilon r)^2} + 3\varepsilon(x_m - \xi_m) sinh^{-1}(\varepsilon r)}{6\varepsilon^2}. \tag{18}$$

With the help of a computer algebra system, higher order derivative and integration of MQ RBFs can be easily obtained. By means of the method presented in Section 2.2, the PDEs (10)–(12) are to be solved by substituting integrated radial basis function networks (14)–(18) for DRBF networks (5)–(9) to get a new linear system similar to (13), thus could be solved in the same way. For a nonlinear dynamical system, a nonlinear algebraic equation system is obtained, and the training for the weights of RBFNs can be carried out by Levenberg–Marquardt algorithm which is one of the most widely used optimisation algorithms.

3. Adaptive method for training the IRBF networks

In this paper, the residual subsampling scheme suggested in [14] is adopted to train the IRBF network. In the training process, neurons are added and removed based on residuals evaluated at a finer point set, and the shape parameter adjusting scheme is modified for suiting the IRBF neuron behaviour which is different from DRBF. The study on the properties of integrated multiquadric radial basis function (IMQ) suggested that less adjusting of the shape parameter is required in comparison with MQ [9]. Therefore, we simply chose the shape parameters by multiplying the distances between two neighbour neurons with a fixed coefficient. The adaptive process for determining the locations of neurons in integrated radial basis function networks for approximating a one-dimensional function is described as follows.

Step 1. Generate an initial IRBF network, using N equally space points as the centre points ξ_j of the neurons.
Step 2. Feed forward the IRBFNs to approximate the function.
Step 3. Compute the approximation error at points in the middle of two neighbour neurons.
Step 4. Choose the points at which the residual exceeds a threshold to be neurons, and remove the centres that lie between two points whose residual is below a smaller threshold.
Step 5. Choose the shape parameter ε_j of each neuron based on the spacing with nearest neighbours.
Step 6. Recompute the new IRBF network, until end criteria are satisfied.

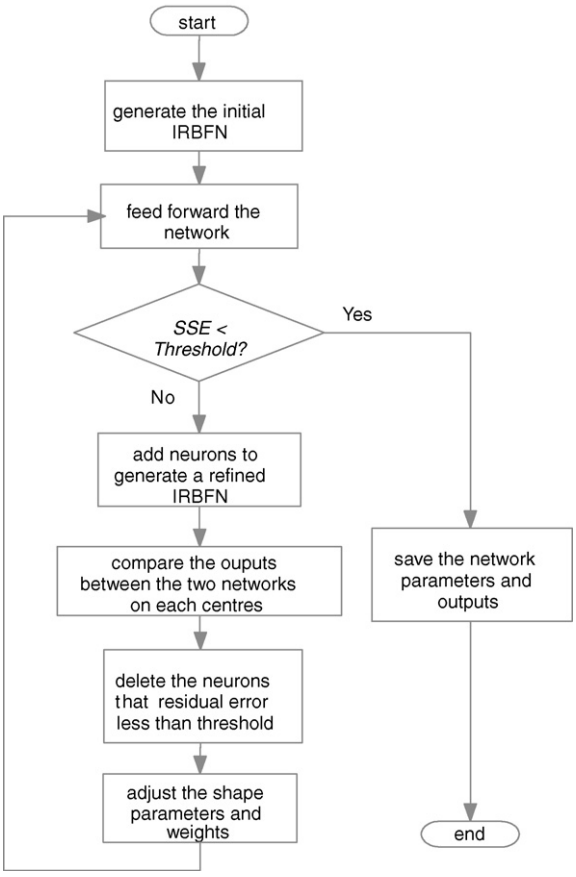

Fig. 1. Flowchart of the training procedure of IRBFNs.

During the training procedure, the two neurons whose centres are end points are always kept fixed.

The training procedure for solving PDEs is straightforwardly derived from approximation, as the flowchart illustrated in Fig. 1. More details about adaptive residual subsampling methods for higher dimensional approximation and solving PDEs are referred to [14].

4. Numerical experiments

In this section, we demonstrate the effectiveness of the proposed method by several numerical experiments in one dimension. All programs are run in MATLAB on a Windows XP platform running at 1.7 GHz.

4.1. Approximation

We start our numerical experiments with nonlinear function approximation for the reason that the ability to solving PDEs via RBFNs is essentially dependent upon the ability to approximation. The errors for determining the accuracy in our numerical experiment is measured by:

$$e_{\max} = \max |\psi(x) - F(x)|, \tag{19}$$

where $\psi(x)$ is the function to be approximate and $F(x)$ is the approximation, and the *SSE* is defined as (4). The first function for testing the approximation performance of IRBFNs with adaptive subsampling method is

$$f_1(x) = 0.02(7.2x^3 - 3.5x^2 + 3x + 12) \cdot (1 + \cos(4\pi x)) \cdot (1 + 0.8\sin(3\pi x)). \tag{20}$$

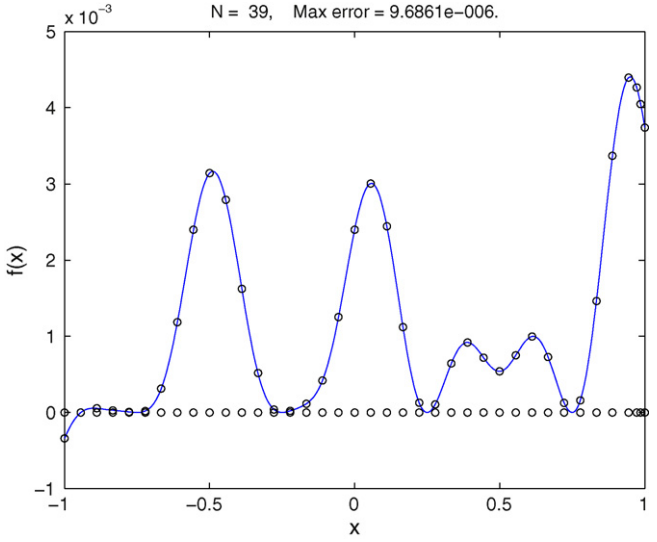

Fig. 2. Approximation of a smooth function (20). The distribution of the centres of neurons and the feed forward values of the IRBFN are shown as circles.

The approximation starts from 10 neurons. The adaptive iteration ended with $N = 39$ neurons when the maximum error is less than 10^{-5}, and the results are shown in Fig. 2.

The second function for test is

$$f_2(x) = |x + 0.04|. \tag{21}$$

This function has a corner feature, and need more iterative steps when the shape parameter is not large enough. The adaptive iteration converges in 12 iterations with $N = 2288$ final total centres. The conditioning of the interpolation matrix A increases rapidly, consistent to the result in [14]. When the number of neuron increases up to 2000, the condition number which is defined as

$$\kappa(A) = ||A|| \cdot ||A^{-1}|| = \sigma_{\max}/\sigma_{\min}, \tag{22}$$

increased to more than 10^{14}, and the warning of the singularity of matrix A was triggered when $\kappa(A) = o(10^{17})$. To counter this ill-condition problem, we applied Moore–Penrose pseudoinverse of matrix based on singular value decomposition (SVD) instead of the LU decomposition method. The result is shown in Fig. 3, and the error convergence procedure is shown in Fig. 4.

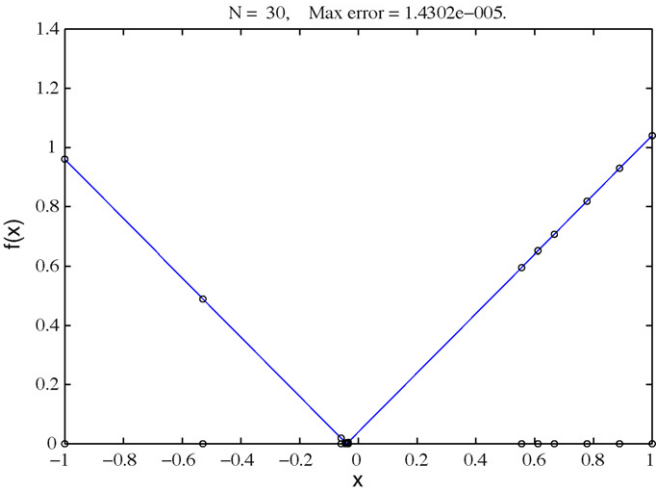

Fig. 3. Approximation of a function with corner feature (21).

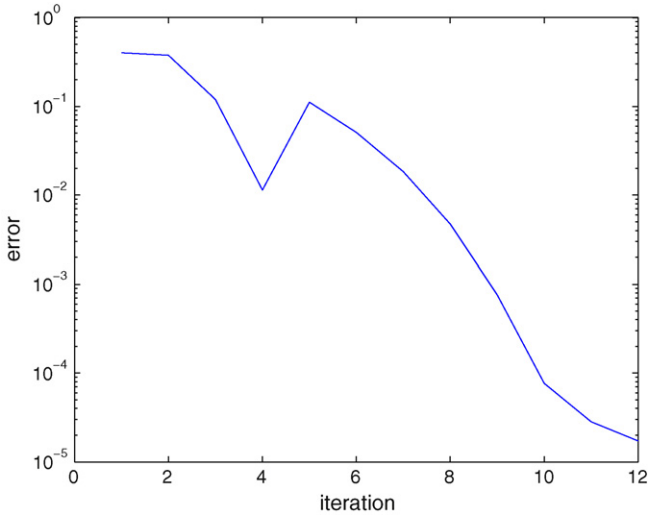

Fig. 4. Error convergence procedure of the IRBFN approximation.

We also test a function with a steep feature:

$$f_3(x) = \tanh(60x - 0.01), \tag{23}$$

and the result is shown in Fig. 5.

4.2. Solving PDEs

The first PDE chosen for testing the proposed method is a boundary problem:

$$\frac{\partial^2 u}{\partial x^2} = -16\pi^2 \sin(4\pi x), \tag{24}$$

with boundary condition: $u(0) = 2, u(1) = 2$. The analytical solution of (24) is

$$u(x) = 2 + \sin(4\pi x).$$

As Fig. 6 shows, the neurons in IRBF network distribute adaptively around the boundary and the turning point, which is able to obtain a higher accuracy solution than the networks built by equidistance neurons.

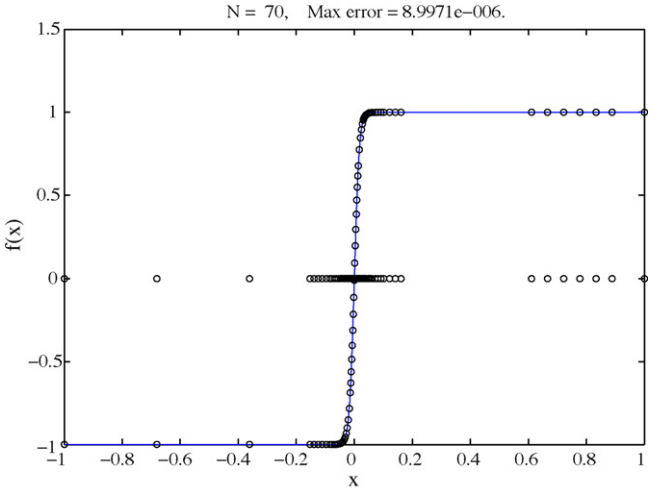

Fig. 5. Approximation of a function with steep feature (23).

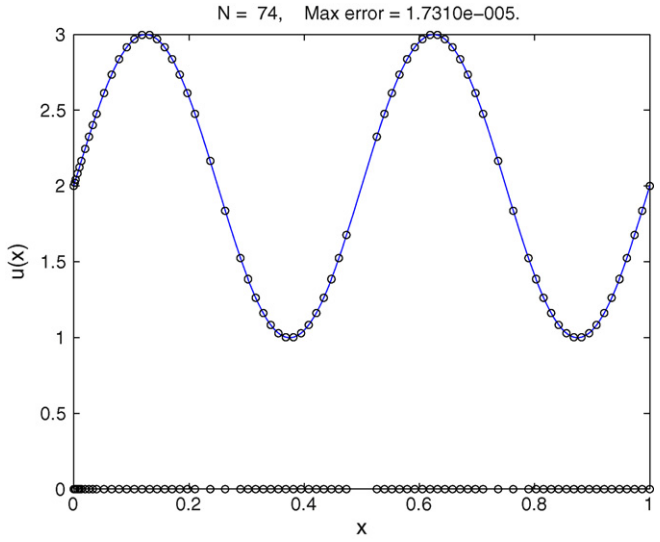

Fig. 6. Numerical solution of a linear PDE (24).

The stationary Burgers' equation is a typical nonlinear PDE, which is given by:

$$\begin{aligned} &\nu u_{xx} - u u_{xx} = 0 \\ &\nu u_{xx}(0) - \kappa(u(0) - \alpha) = 0 \\ &\nu u_{xx}(1) + \kappa(u(1) + \alpha) = 0 \end{aligned} . \tag{25}$$

The closed form solution is [14]:

$$u(x) = -\beta \tanh\left(\frac{1}{2}\beta\nu^{-1}\left(x - \frac{1}{2}\right)\right). \tag{26}$$

where β satisfies

$$-\frac{1}{2}\beta^2 \sec h^2\left(\frac{1}{4}\beta\nu^{-1}\right) + \kappa\left(\alpha - \beta\tanh\left(\frac{1}{4}\beta\nu^{-1}\right)\right) = 0. \tag{27}$$

In our numerical experiment the parameter are chosen as $\nu = 5 \times 10^{-3}$, $\alpha = 1$, $\kappa = 2$. We use Levenberg–Marquardt algorithm to solve the obtained nonlinear algebraic equation. As shown in Fig. 7, the iteration is ended up with $N = 78$ neurons. By comparison, the DRBF network with the same adaptive scheme uses 396 neurons to obtain the same accuracy.

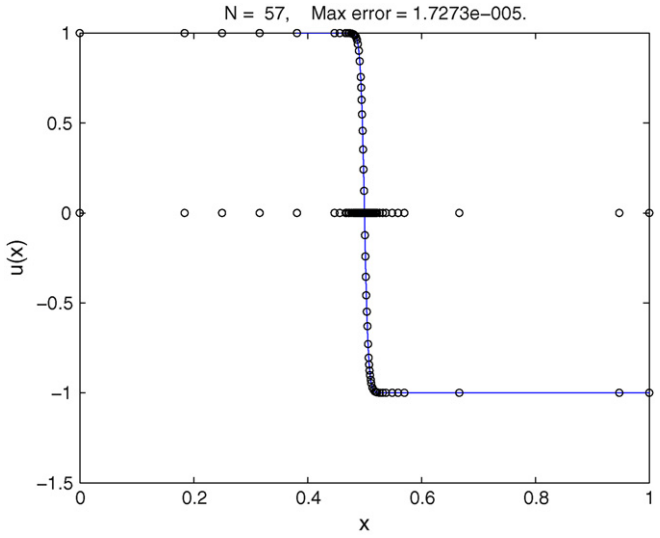

Fig. 7. Numerical solution of stationary Burgers's equation.

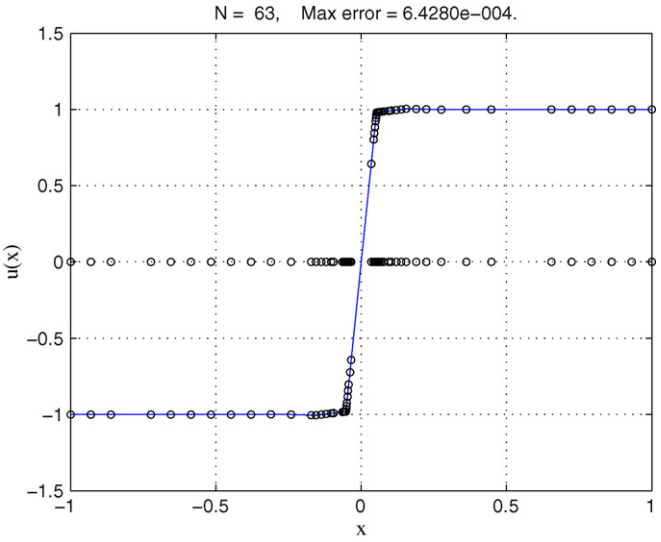

Fig. 8. Numerical solution of Allen–Cahn equation.

We also test the stationary Allen–Cahn equation, given by

$$\begin{aligned} &\nu u_{xx} + u - u^3 = 0 \\ &u(-1) = -1 \\ &u(1) = 1 \end{aligned} . \tag{28}$$

where the parameter ν is chosen as 10^{-4} in this study. The adaptive procedure ends up after 32 iterations, and the result is shown in Fig. 8. By comparison, the DRBF network with the same adaptive scheme uses 132 neurons, and *bvp4c* in MATLAB needs 270 nodes [14].

5. Conclusion

In this paper, a method for approximation and solving PDEs based on indirect radial basis function networks with adaptive residual subsampling training method are presented. The numerical experiment shows that this method is effective. For solving PDEs, IRBFNs are capable to smooth the derivative errors. With the proposed adaptive procedure, IRBFNs requires less neurons to attain the accuracy than direct radial basis function network.

Approximations based on smooth IRBFNs are highly effective in approximating smooth functions, even if the neuron set is relatively coarse. Observation on numerical experiments for RBF-based approximation [11] have shown that in the case of very fine resolution node set, edge errors on turning point dominate over interior errors. The adaptive method applied for training in this paper is an effective technique for dealing with the steep and corner feature of the PDEs solutions; and the IRBF networks contribute to improve the accuracy of solving PDEs. Therefore, the combination of IRBF and adaptive algorithm is a promising approach for mesh-free solution of PDEs.

The proposed method can easily be applied for solving higher dimensions problem and time-depend nonlinear equations. The application of this method in nonlinear system modeling and control is also an interesting issue deserving further studies.

Acknowledgement

This work is partly supported by Hydrodynamics Research Foundation under the grant 9140A14030308CB49.

References

[1] J. Park, I. Sandberg, Universal approximation using radial-basis function networks, Neural Comput. 3 (1991) 246–257.
[2] E. Kansa, Multiquadrics—a scattered data approximation scheme with applications to computational fluid-dynamics. I. Surface approximations and partial derivative estimates, Comput. Math. Appl. 19 (1990) 127–145.
[3] E. Kansa, Multiquadrics—a scattered data approximation scheme with applications to computational fluid-dynamics. II. Solutions to parabolic, hyperbolic and elliptic partial differential equations, Comput. Math. Appl. 19 (1990) 147–161.
[4] G. Fasshauer, Solving partial differential equation by collocation with radial basis functions, in: A. LeMehaute, C. Rabut, L. Schumaker (Eds.), Surface Fitting and Multiresolution Methods, Vanderbilt University Press, Nashville, TN, 1997, pp. 131–138.
[5] E. Larsson, B. Fornberg, A numerical study of some radial basis function based solution methods for elliptic PDEs, Comput. Math. Appl. 46 (2003) 891–902.
[6] I. Dag, Y. Dereili, Numerical solutions of KDV equation using radial basis functions, Appl. Math. Model. 32 (2008) 535–546.
[7] N. Mai-Duy, T. Tran-Cong, Approximation of function and its derivatives using radial basis function networks, Appl. Math. Model. 27 (2003) 197–220.
[8] N. Mai-Duy, T. Tran-Cong, Numerical solution of differential equations using multiquadric radial basis function networks, Neural Netw. 14 (2001) 185–199.
[9] S. Sarra, Integrated multiquadric radial basis function approximation methods, Comput. Math. Appl. 51 (2006) 1283–1296.
[10] C. Shu, L. Wu, Integrated radial basis functions-based differential quadrature method and its performance, Int. J. Numer. Methods Fluids 53 (2007) 969–984.
[11] B. Fonberg, T. Driscoll, G. Wright, R. Charles, Observations on the behavior of radial basis function approximations near boundaries, Comput. Math. Appl. 43 (2002) 473–490.
[12] J. Gonzalez, I. Rojas, J. Ortega, Multiobjective evolutionary optimization of the size, shape, and position parameter of radial basis function network for function approximation, IEEE Trans. Neural Netw. 14 (2003) 1478–1495.
[13] S. Sarra, Adaptive radial basis function methods for time dependent partial differential equations, Appl. Numer. Math. 54 (2005) 79–94.
[14] T. Driscoll, A. Heryudono, Adaptive residual subsampling methods for radial basis function approximation and collocation problems, Comput. Math. Appl. 53 (2007) 927–939.