CENTRO DE INVESTIGACIÓN Y DE ESTUDIOS AVANZADOS DEL INSTITUTO POLITÉCNICO NACIONAL

UNIDAD ZACATENCO

DEPARTAMENTO DE CENTRO CONTROL AUTOMATIC

“ Control y Modelado con ecuaciones difusas y numero Z”

TESIS

Que presenta

Raheleh Jafari

Para obtener el grado de

Doctora en Ciencias

EN LA ESPECIALIDAD DE

CONTROL AUTOMATIC

Director de Tesis:

Dr. Wen Yu Liu

Ciudad de Mexico mayo, 2017

CENTER FOR RESEARCH AND ADVANCED STUDIES OF THE NATIONAL POLYTECHNIC INSTITUTE

Cinvestav

UNIDAD ZACATENCO

DEPARTMENT OF AUTOMATIC CONTROL

“ Fuzzy modeling and control with fuzzy equations and z-number”

D I S S E R T A T I O N

Presented by

Raheleh jafari

To obtain the grade of

DOCTOR OF SCIENCE

WITH THE SPECIALTY IN

AUTOMATIC CONTROL

Supervissor:

Dr. Wen Yu Liu

Mexico city may,2017

Contents

List of Figures

Chapter 1

Introduction

Fuzzy control can be divided into direct and indirect methods [83]. The direct fuzzy uses a fuzzy system as a controller, while the indirect fuzzy control uses a fuzzy model to approximate the nonlinear system. The indirect fuzzy controller utilizes the simple topological structure and universal approximation ability of fuzzy model. It has been widely used in uncertain nonlinear system control. We use indirect fuzzy control in this research work.

Conventional mathematical tools for example difference equations, algebraic systems, interpolation polynomial, have been used extensively for system modeling and parameter identification. Uncertain nonlinear system modeling is a mature subject with a variety of powerful methods and a long history of successful industrial applications. Fuzzy modeling method is a popular tool for uncertain nonlinear system modeling. The fuzzy model usually comes from several fuzzy rules [202]. These fuzzy rules represent the controlled nonlinear system. Since any nonlinear system can be approximated by several piecewise linear systems (Takagi-Sugeno fuzzy model) or known nonlinear systems (Mamdani fuzzy model) [140], fuzzy models can approximate a large class of nonlinear systems while keeping the simplicity of the linear models. In this work, we discuss another type of fuzzy model. The basic idea is that many nonlinear systems can be expressed by linear-in-parameter models, such as Lagrangian mechanical systems [194]. The parameters of these models are uncertain and the uncertainties satisfy the fuzzy set theory [219]. In this way the inconvenience problems in nonlinear modeling, such as complexity and uncertainty, are solved by the fuzzy logic theory and linear-in-parameter structure. In recent days, many methods involving uncertainties have used fuzzy numbers [55][76][110][187], where the uncertainties of the system are represented by the fuzzy coefficients. The models are fuzzy equations or fuzzy differential equations (FDEs). The modeling process with the fuzzy equation is to find the fuzzy coefficients of the linear-in-parameter model such that the fuzzy equation can represent the uncertain nonlinear system. The application of the fuzzy equations and the FDEs are in direct connection with the nonlinear modeling and control.

The decisions are carried out based on knowledge. In order to make the decision fruitful, the knowledge acquired must be credible. Z-numbers are associated with the reliability of knowledge [221]. Many fields related to the analysis of the decisions actually use the ideas of Z-numbers. Z-numbers are much less complex for calculation in comparison with nonlinear system modeling methods. The Z-number is abundantly adequate number compared with the fuzzy number. Although Z-numbers are implemented in many literatures, from theoretical point of view this approach is not certified completely. Here the uncertainties are in the sense of Z-numbers.

1.1 Motivations

The nonlinear system modeling corresponds to find the fuzzy parameters of the fuzzy equations or FDEs and the fuzzy control is to design suitable nonlinear functions in the fuzzy equation or FDE. Conventional mathematical tools for example difference equations, algebraic systems, interpolation polynomial have been used extensively for system modeling and parameter identification. A special case of uncertain system modeling with fuzzy equation is fuzzy polynomial interpolation. There are various interpolation techniques. However, if there are uncertainties in the interpolation points, some methods cannot work well. This is the reason why we use the fuzzy polynomial interpolation.

The general form of fuzzy polynomial is fuzzy equation. It can be applied directly for nonlinear control. The nonlinear system modeling corresponds to find the fuzzy parameters of the fuzzy equation or FDE and the fuzzy control is to design suitable nonlinear functions in these equations. Both fuzzy modeling and fuzzy control via fuzzy equations and FDEs require solution of these equations. There are various approaches. However, all of analytical methods for the solutions of fuzzy equations or FDEs are very difficult to be applied, especially for nonlinear fuzzy equations or FDEs. Moreover, the numerical methods are very complex and the approximation accuracy of the numerical calculations are normally less. Neural networks can give a good estimation for the solutions of fuzzy equations but since the structure of the neural network is not suitable for FDE, the approximation accuracy is poor. Thats why we use a new model named Bernstein neural network which has good properties of Bernstein polynomial for FDE. Since Z-numbers have higher potential to illustrate the information of the human being, in this research work the uncertainties are in the sense of Z-numbers.

1.2 Contributions of this thesis

Many uncertain nonlinear systems can be modeled by linear-in-parameter models. The uncertainties can be regarded as parameter changes, which can be described as fuzzy numbers. These models are fuzzy equations or FDEs. They are alternative models for uncertain nonlinear systems. The modeling of the uncertain nonlinear systems is to find the coefficients of these equations. The solutions of them are the controllers and are applied to analyze many engineering problems. However, it is very difficult to obtain solutions of the mentioned equations.

In this research work the modeling and controlling uncertainty nonlinear systems with fuzzy equations is proposed. We use fuzzy equations to model uncertain nonlinear systems. The uncertainty is represented by fuzzy numbers. Since normal modeling methods cannot be applied for fuzzy number and fuzzy equation directly, we transform the fuzzy equation into a neural network. Then we modify the gradient descent method for fuzzy numbers and propose a back-propagation learning rule for fuzzy equations. In continue we propose a novel fuzzy controller via dual fuzzy equations which are the general case of fuzzy equations. The controllability condition is given for the fuzzy control through these equations. Two types of neural networks are applied to approximate the solutions of the mentioned equations. These solutions are then transformed into the fuzzy controllers. Afterwards approximation of the solutions of FDEs by two types of Bernstein neural networks is suggested. We first transform the FDE into four ordinary differential equations (ODEs) with Hukuhara differentiability. Then we construct neural models with the structure of ODEs. With modified backpropagation method for fuzzy variables, the neural networks are trained. Also a methodology involving novel iterative technique considering neural networks is suggested to extract approximate solution for the second-order nonlinear partial differential equations (PDEs) with real constant coefficients (RCCs) taking into account initial and boundary conditions. This perspective is designed to grant good approximation on the basis of learning technique which is associated with quasi-Newton rule. The constructed neural network has the regularizing parameters (weights and biases), which can be utilized to make the error function minimal. The construction of the model leads to the satisfaction of the initial and boundary conditions along with the training of neural network which satisfies the PDEs. Also a sophisticated methodology is provided in order to solve PDEs on the basis of the application of Bernstein polynomial that is modeled with the help of two pattern of neural networks. In continue, the uncertainties are in the sense of Z-numbers. We use dual fuzzy equations as the models. The conditions of controllability are proposed. Two types of neural networks are implemented to approximate solutions of the fuzzy equations with Z-number coefficients.

1.3 Resumen

Muchos sistemas no lineales inciertos pueden ser modelados por modelos lineales en parámetros. Las incertidumbres pueden ser consideradas como cambios de parámetros, que pueden ser descritos como números difusos. Estos modelos son ecuaciones difusas o FDEs. El modelado de los sistemas no lineales inciertos es encontrar los coeficientes de estas ecuaciones. Las soluciones de ellos son los controladores y se aplican para analizar muchos problemas de ingeniería. Sin embargo, es muy difícil obtener soluciones de las ecuaciones mencionadas.

En este trabajo de investigación se propone el modelado y control de sistemas no lineales de incertidumbre con ecuaciones difusas. Utilizamos ecuaciones difusas para modelar sistemas no lineales inciertos. La incertidumbre está representada por números difusos. Dado que los métodos normales de modelado no pueden aplicarse directamente al número difuso ya la ecuación difusa, transformamos la ecuación difusa en una red neuronal. A continuación, modificamos el método de descenso de gradiente para la actualización de números difusos y proponemos una regla de aprendizaje de retropropagación para ecuaciones difusas. En continuo proponemos un nuevo controlador difuso a través de ecuaciones difusas dobles que son el caso general de las ecuaciones difusas. La condición de controlabilidad se da para el control difuso a través de estas ecuaciones. Se aplican dos tipos de redes neuronales para aproximar las soluciones de las ecuaciones mencionadas. Estas soluciones se transforman a continuación en los controladores difusos. Posteriormente se sugiere la aproximación de las soluciones de FDEs por dos tipos de redes neuronales de Bernstein. Primero transformamos el FDE en cuatro ecuaciones diferenciales ordinarias (ODEs) con diferenciación de Hukuhara. Entonces construimos modelos neuronales con la estructura de ODEs. Con el método de retropropagación modificado para las variables difusas, se forman las redes neuronales. También se sugiere una metodología que implique una nueva técnica iterativa considerando redes neuronales para extraer la solución aproximada para las ecuaciones diferenciales parciales no lineales (EDP) de segundo orden con coeficientes constantes reales (CCR) teniendo en cuenta las condiciones iniciales y fronterizas. Esta perspectiva está diseñada para conceder una buena aproximación sobre la base de la técnica de aprendizaje que se asocia con la regla cuasi-Newton. La red neural construida tiene los parámetros de regularización (pesos y sesgos), que pueden utilizarse para hacer que la función de error sea mínima. La construcción del modelo conduce a la satisfacción de las condiciones iniciales y fronterizas junto con el entrenamiento de la red neuronal que satisface las PDEs. También se proporciona una sofisticada metodología para resolver las PDEs sobre la base de la aplicación del polinomio de Bernstein que es modelado con la ayuda de dos patrones de redes neuronales. Para continuar, las incertidumbres están en el sentido de números Z. Utilizamos ecuaciones difusas duales como los modelos. Se proponen las condiciones de controlabilidad. Se implementan dos tipos de redes neuronales para aproximar soluciones de las ecuaciones

difusas con coeficientes de número Z.

1.4 Acknowledgments

I would like to express my sincere gratitude to my advisor Prof. Wen Yu Liu for the continuous support of my Ph.D study and research, for his patience, motivation, enthusiasm and immense knowledge. His guidance helped me throughout my research and writing of this thesis. I could not have imagined having a better advisor and mentor for my Ph.D study.

Besides my advisor, I would like to thank my thesis committee members: Prof. Cristóbal Vargas Jarillo, Prof. Alejandro Justo Malo Tamayo, Prof. Sergio Salazar and Prof. Tovar Rodriguez Julio Cesar. Also I would like to thank my friends in my institution.

Last but not the least, I would like to thank my family: my parents, my husband and brothers for supporting me spiritually throughout writing this thesis and my life in general.

1.5 Publications

Most contributions described in this thesis have appeared in various publications. Below are the list of publications:

1.5.1 International journals

1. R. Jafari, W. Yu, "Fuzzy Control for Uncertainty Nonlinear Systems with Dual Fuzzy Equations", *Journal of Intelligent and Fuzzy Systems*, Vol.29, pp.1229-1240, 2015.

2. R. Jafari, W. Yu, Uncertainty Nonlinear Systems Modeling with Fuzzy Equations, Mathematical problems in Engineering, Vol. 2017, Article ID 8594738, 10 pages. https://doi .org/10.1155/2017/8594738

3. R. Jafari, W. Yu, Fuzzy Differential Equation for Nonlinear System Modeling with Bernstein Neural Networks, IEEE Access, 2017. doi:10.1109/ACCESS.2017.2647920.

4. R. Jafari, W. Yu, Uncertain nonlinear system control with fuzzy equations and Z-numbers, Intelligent automation and Soft Computing, (Submitted to Intelligent Automation and Soft Computing).

5. R. Jafari, W. Yu, Uncertain nonlinear system control with Fuzzy Differential Equations and Z-numbers, International Journal of Computational Intelligence Systems, (Submitted to Intelligent Automation and Soft Computing).

1.5.2 International conferences

1. R. Jafari, W. Yu, “Uncertainty Nonlinear Systems Control with Fuzzy Equations”, *2015 IEEE International Conference on Systems, Man, and Cybernetics (SMC15)*, Hong Kong, China, pp. 2885-2890, 2015.

2. R. Jafari, W. Yu, “Uncertainty Nonlinear Systems Modeling with Fuzzy Equations”, *16th IEEE International Conference on Information Reuse and Integration (IRI15)*, San Fracisco, USA, pp. 182-187, 2015.

3. R. Jafari, W. Yu, “Artificial Neural Network Approach for Solving Strongly Degenerate Parabolic and Burgers-Fisher Equations”, *12th International Conference on Electrical Engineering, Computing Science and Automatic Control (CCE’15)*, Mexico City, Mexico, 2015.

4. R. Jafari, W. Yu, “Solving Fuzzy Differential Equation with Bernstein Neural Networks”, *2016 IEEE International Conference on Systems, Man, and Cybernetics (SMC16)*, Budapest, Hungary, 2016.

5. R. Jafari, W. Yu, Fuzzy controller design with fuzzy differential equation and Z- numbers, 18th IEEE International conference on industrial technology , Canada, 2017.

6. R. Jafari, W. Yu, Fuzzy controller design with fuzzy equations and z-numbers, IEEE International conference on fuzzy system, Italy, 2017.

Chapter 2

Fuzzy equations for the modeling and control of uncertain nonlinear systems

A review of the methodologies associated with the modeling and control of uncertain nonlinear systems has been given due importance in this chapter. The basic criteria that highlights the work is relied on the various patterns of techniques incorporated for the solutions of fuzzy equations as well as differential equations that correspond to controllability constraint related to fuzzy control. The solutions which are generated by these equations are considered to be the controllers. Currently, numerical techniques have come out as superior techniques in order to solve these type of problems. Taking into consideration the modeling case as well as controlling uncertain nonlinear systems, the implementation of neural networks technique has contributed in the complex way of dealing the appropriate coefficients and solutions of the fuzzy systems. In the current context, few types of neural networks to be mentioned as feed-forward (static) as well as recurrent (dynamic) at par with least mean square and quasi-Newton learning techniques have been demonstrated. In the context of this review, the applications and the effectiveness of fuzzy control design techniques in real world have been reviewed.

2.1 Introduction

An exceptional instance of uncertain system modeling at par with fuzzy equation is fuzzy polynomial interpolation [159]. Polynomials have been used with fuzzy coefficients in order to interpolate uncertain data that have been expressed using fuzzy numbers [39]. Interpolation methodolgy has been broadly utilized for function approximation as well as system identification [47, 144, 179]. The theory constraint associated with polynomial interpolation is researched in [189]. It elaborates that the interpolation of the function $f(x)$ includes $O(n)$ time complexity at par with n data points. In [223], two-dimensional polynomial interpolation is demonstrated. The constraint associated to multivariable interpolation has been investigated in [159]. In [169], the multivariate Vandermode matrix is utilized. Sparse grid interpolation is an extra technique. Smooth function approximation has been broadly implemented currently [200][207]. It yields a model by utilizing Lagrange interpolating polynomials, at the points of product grids [215][39]. Whatsoever if it involves uncertainties in the interpolation points, the suggested techniques will not work appropriately. For this purpose we use the fuzzy polynomial interpolation.

The generalized form of fuzzy polynomial is termed as fuzzy equation. In comparison with the normal systems, fuzzy equations are considered to be very noncomplex [140][202]. It is feasible for them to apply directly for nonlinear control. The nonlinear system modeling is concerned to obtain the fuzzy parameters related to fuzzy equation. The approach of fuzzy control is associated with the design of appropriate nonlinear functions in the fuzzy equation. Fuzzy modeling as well as fuzzy control through fuzzy equations in combination require solution of the fuzzy equation. Several approaches are incorporated. [85] utilized the parametric mode of fuzzy numbers and restored the original fuzzy equations using crisp linear systems. In [56], the extension principle is implemented and it suggests that the coefficients can be either real or complex fuzzy numbers. Whatsoever, the validation of the solution is not assured. [4] prescribed the homeotypic analysis methodology. [9] inducted the Newton's technique. In [29], the solution of fuzzy equations are extracted using the fixed point methodology. One of the methodology which is most talked recently is the method of α-level [91]. By utilizing the technique of superimposition of sets, resolving of the fuzzy numbers will be possible. In current days, fuzzy fractional differential as well as integral equations have been researched abundantly in [14][32][145][186][211]. Whatsoever the above methodologies are somewhat complicated.

The numerical solution associated with fuzzy equation can be fetched using the iterative technique [119][136], interpolation technique [213], and Runge-Kutta technique [174]. It can also be implemented to FDEs [138]. These techniques are also complicated for implementation. Both neural networks as well as fuzzy logic are considered to be the universal estimators which can estimate any nonlinear function to any notified precision [69]. Current outcomes demonstrate that the fusion methodology of these two different techniques appears to be highly efficient for nonlinear systems identification [218]. Neural networks can also be implemented for resolving the fuzzy equation. A generalized fuzzy quadratic equation is resolved by utilizing the neural networks which has been mentioned in [49]. [111][112] elaborated the outcomes of [49] into fuzzy polynomial equation. Neural networks have been utilized in order to extract the solution of dual fuzzy equations which has been illustrated in [110]. A matrix pattern associated to the neural learning has been quoted in [153]. Whatsoever, these techniques are not usual as they cannot resolve general fuzzy equations associated to neural networks. Also, they cannot generate the fuzzy coefficients straight away associated with neural networks [201][108].

In several FDEs, the coefficients are in the form of fuzzy numbers which are used to demonstrate the uncertainties [78]. The applications associated to FDEs are at par with nonlinear modeling as well as control [110]. FDEs with different attributes utilize fuzzy variables for illustrating the uncertainties. The investigation on the solutions of FDE are incorporated with chaotic analysis, quantum system as well as several engineering constraints, viz civil engineering and modeling actuators. The general idea in concerned with fuzzy derivative was initially laid down by [64]. After that it is elaborated in [75]. The linear first-order equation is the most generalized FDE. In [43] the analytical solution is extracted via generalizing the differentiability. The first order FDE along with periodic boundary conditions is investigated and mentioned in [126], also the higher order linear FDE is discussed. In

[27], the analytical solutions related to the second order FDE are extracted. The analytical solutions associated to the third order liner FDE are demonstrated in [95]. Also, analytical techniques in order to solve n-th order linear FDE are suggested in [53][24].

Too much complexity is involved in solving nonlinear FDE. [195] investigated the basis solutions of nonlinear FDEs associated with generalized differentiability by utilizing interval-valued methodology. [163] utilized periodic boundary as well as Hukuhara differentiability for impulsive FDE. [82] proposed some appropriate criteria in order to fuzzify the crisp solutions. [135] utilized two-point fuzzy boundary value in concerned with FDE. [94] illustrated homotopy analysis methodology at par with for FDE. Whatsoever, all of the above mentioned analytical methodologies associated to the solutions of FDEs are very much complex, notably for nonlinear FDEs.

Numerical solutions of FDEs have been elaborately mentioned by many researchers in current days. An iterative methodology for the numerical solutions related to the first-order FDE is suggested in [192]. Laplace transform has been utilized for second-order FDE in [18]. The elaboration of classical fuzzy set theory in [101] results in obtaining of numerical solution related to FDE. The predictor-corrector technique is implemented in [23]. [139][205][162] portrays the methodology of Euler numerical technique in order to resolve FDE. To be mentioned few numerical approaches, such as Nystrom method [125], Taylor method [5] and Runge-Kutta method [173] can also be implemented for resolving FDEs. Whatsoever, the approximation precision of these numerical methods are generally not very much [137].

The solution of FDE is uniformly continuous, also it is inside compact sets [52]. Neural networks is a superior approximator for estimating the solutions of FDEs. [15] illustrated that the solution of ODE can be estimated with the help of neural network. [84] demonstrated the variations in the midst of the exact solution and approximation solutions associated to ODEs. [146][217] implemented neural approximations at par with ODEs to dynamic systems. [147] applied B-splines neural network for obtaining the approximated solutions related to nonlinear ODEs. [132] implemented dynamics neural networks for the approximation of the first-order ODE. There are very limited number of works related to FDE. [77] proposed a static neural network in order to resolve FDE. As the structure of the neural network is not appropriate for FDE, the precision of the approximation is not of quality level.

Exact (closed-form) solution associated to differential equation implicates a superior task in the appropriate holding of qualitative natures related to several proceedings as well as events at par with various fields in concerned to natural science. Exact solutions allow researchers to lay down design as well as to initiate experiments by establishing justifiable natural provisions for determination of these parameters or functions. Whatsoever, extracting the exact solutions related to the PDEs apart from very particular or non complex cases is too complex. Investigation of previous research opens up the fact that several methodologies have been laid down for resolving some types of PDEs. The Homotopy analysis methodology has been incorporated to obtain the solution of linear as well as nonlinear higher dimensional initial boundary value problems with variable coefficients which has been mentioned in [109]. The Homotopy perturbation methodology in order to solve PDEs at par with variable coefficients is employed in [117]. In [33, 113] the PDEs have been solved by utilizing two-dimensional differential transformation techniques. In [209] the modified technique at par with non complex equation has been incorporated in order to find the exact analytical solutions associated to nonlinear PDEs. In [107] Adomain decomposition method is applied in order to solve Burgers-Fisher equation. Moghimi et al. [151] utilized a novel iteration technique in order to carry out generalized Burger-Fisher as well as Burger equation. In [214] an iteration technique in order to solve both linear as well as nonlinear wave equations has been researched.

Other researchers have also mentioned some numerical solutions related to PDEs. The spreadsheet program has been utilized in order to generate the numerical solution of the hyperbolic equation which is mentioned in [124] . Some experiments have been conducted in order to generate solution which is in the form of an array that holds the value of the solution at chosen group of points [128]. Additional group of investigators implemented finite element technique which has been variedly used in the area of mechanics for solving few specific PDEs [102]. There prevail other numerical approach namely finite difference technique [193]. An explicit monotone difference technique in order to estimate the entropy solutions related to degenerate parabolic equation is introduced in [79]. In [57] a Taylor polynomial estimation in order to extract the solution of hyperbolic PDEs associated to constant coefficients has been laid down. In [114] the spectral collocation technique is illustrated for obtaining the solution related to the generalized Burgers-Fisher equation. The investigation revealed by the researchers related to double non-traveling wave solutions associated to two systems of nonlinear PDEs has been mentioned in [92]. Colzani et al. [67] have researched the radial solutions of the wave equation in concerned to the Euclidean space. In [80] the application of convolution quadrature was revealed in relation to the time-domain boundary integral formulation associated to the wave equation at par with non-zero initial conditions. Higdon [99] investigated on the finite difference equation and came to the conclusion that the discrete forms of the one-way wave equation are compatible at par with the analytical forms which ascertain suitable absorption at certain nonzero angles of incidence. A linear wave equation in concerned to a boundary damping term was investigated in [142]. The Cauchy problem at par with the semilinear wave equation incorporated in the Schwarzschild metric (3+1)-dimensional spacetime was illustrated in [62]. The results of feedback control in refer to the wave equation has been illustrated in [90], whereas the open loop control in concerned to the wave equation has been demonstrated in [88, 129]. Whatsoever, there is complexity involved with the above mentioned methodologies.

Since the solutions of the PDE are uniformly continuous and also the safety problems in connection to the viable sets are mostly compact, neural networks are superiorly selected candidates for the estimation of the feasibility problems [69]. The capacity in concerned to the function approximation which is embedded to the neural networks gives accurate and differentiable solutions in a closed analytic form [96]. In [72] some appropriate theoretical results on the basis of asymptotic manner of finite neural networks, considering the issue of fixed boundary conditions has been validated. The methodology for optimization associated to the training of multidimensional neural network as well as carrying out its simulation was mentioned in [130]. A feed-forward neural network is suggested in order to resolve an elliptic PDE in connection to 2D in [74]. Other methodology for solving a class of first-order PDEs on the basis of multilayer neural networks is demonstrated in [98]. The investigators of [152] laid down an unsupervised neural network to resolve the nonlinear schrodinger equation. In

[30] a two dimensional wave equation as well as the vibration problem in relation to a lateral beam having both fixed ends up to five decimal digits accuracy incorporated in a neural approach has been resolved. In [198] by employing a feed forward neural network, controlled heat problem at par with three decimal digits accuracy has been solved. Whatsoever, the advancement in researches related to the approximation abilities of networks will display that artificial neural networks can be numerically precise and foreseeable as extraordinary computational methodologies. The progress in training and analysis of existing architectures can be carried out using artificial neural networks which are termed as numerical tools.

2.2 Fuzzy equations and dual fuzzy equations

2.2.1 Fuzzy neural network method

In [51] neural network has been employed for solving fuzzy linear equation

$$AX = C \tag{2.1}$$

where A, B and X are considered to be triangular fuzzy numbers. Taking into account certain values of A and C, (2.1) generates no solution for X [56]. The training of neural network in order to solve (2.1) was mentioned by the researchers in [51], considering that zero is not at par with the support of A. The investigation was carried out considering neural network solutions termed to be Y and X^*. When there is no restrictions in concerned to the weights of the network, then the neural network output will be Y. The non existence of relationship between Y and X was proved and validated by utilizing computer analysis. X^* is the solution of the neural network, taking into consideration that the certain sign restrictions are set on the weights. X^* is illustrated to be an approximation which is named as a new solution of fuzzy equations. It has been displayed by using $X \leq X^*$.

The evolutionary algorithm as well as neural network in combination have been utilized for solving fuzzy equation which has been mentioned in [54] as follows

$$AX \oplus B = C \tag{2.2}$$

where A, B, C and X are termed as triangular fuzzy numbers. The first solution X_c related to (2.2) is stated to be the classical solution that utilizes α-cut and interval arithmetic for obtaining X_c.

Example 2.1 *Assume* $[A] = (1, 2, 3)$*,* $[B] = (-3, -2, -1)$ *and* $[C] = (3, 4, 5)$*. Employing the intervals into the fuzzy equation generates*

$$\begin{aligned}(1+\alpha)\underline{X}_c^{\alpha} + (-3+\alpha) &= (3+\alpha)\\ (3-\alpha)\overline{X}_c^{\alpha} + (-1-\alpha) &= (5-\alpha)\end{aligned}$$

here $[X_c]^{\alpha} = (\underline{X_c}^{\alpha}, \overline{X_c}^{\alpha})$*. It can be extracted*

$$\begin{aligned}\underline{X}_c^{\alpha} &= \tfrac{6}{1+\alpha}\\ \overline{X}_c^{\alpha} &= \tfrac{6}{3-\alpha}\end{aligned}$$

However $[\underline{X}_c^{\alpha}, \overline{X}_c^{\alpha}]$ *does not state a fuzzy number as because* $\underline{X}_c^{\alpha}(\overline{X}_c^{\alpha})$ *is a decreasing (increasing) function of* α*. Occasionally* X_c *prevails and sometimes wont exist.*

By the fuzzification of the crisp solution $(c-b)/a, a \neq 0$, we extract the other solution. $(C-B)/A$ represents the fuzzified solution, taking into assumption that zero is not at par with the support of A. For the evaluation of the fuzzified solution, two approaches have been suggested. The primary approach generates the solution X_e by utilizing the extension principle as well as the secondary approach generates the solution X_I by the means of α-cut and interval arithmetic. X_e can be achieved as mentioned below

$$X_e = \min\{\Pi(a, b, c) | (c-b)/a = x\}$$

where $\Pi(a, b, c) = \min\{A(a), B(b), C(c)\}$. For obtaining α-cut of X_e the process is described as follows

$$\begin{aligned}\underline{X}_e^{\alpha} &= \min\{\tfrac{c-b}{a} | a \in [A]^{\alpha}, b \in [B]^{\alpha}, c \in [C]^{\alpha}\}\\ \overline{X}_e^{\alpha} &= \max\{\tfrac{c-b}{a} | a \in [A]^{\alpha}, b \in [B]^{\alpha}, c \in [C]^{\alpha}\}\end{aligned}$$

where $[X_e]^{\alpha} = (\underline{X_e}^{\alpha}, \overline{X_e}^{\alpha})$. The solution X_I can be calculated as follows

$$[X_I]^{\alpha} = ([C]^{\alpha} - [B]^{\alpha})/[A]^{\alpha}$$

The original fuzzy equation maybe or may not be solved by $X_e(X_I)$. Taking into account some fuzzy equations, X_e are mathematically very much complex to extract, so in [54] an evolutionary algorithm has been implemented for estimating their α-cuts. The mentioned paper can be normalized for interacting with the fuzzy problems, evolutionary algorithms as well as neural nets. There have been disadvantages incorporated in the method which has been mentioned in [54]. The method is exclusively meant for the symmetric fuzzy numbers, in addition it computes just the upper bound as well as the lower bound of the fuzzy numbers avoiding the center part.

An architecture related to fuzzy neural network is suggested in order to obtain a real root at par with fuzzy polynomials which is illustrated in the form mentioned below [11]

$$A_1x + ... + A_nx^n = A_0 \tag{2.3}$$

where $x \in R$ as well as $A_0, A_1, ..., A_n \in E$. A learning algorithm associated with the cost function in order to tweak the crisp weights has been suggested. The methodology mentioned in [11] has drawbacks. It was solely capable of extracting a crisp solution of fuzzy polynomials, and this neural network cannot extract a fuzzy solution.

In [111] the researchers obtained the approximate solution related to the following fuzzy polynomial having degree n

$$A_1x + ... + A_nx^n = A_0 \tag{2.4}$$

where $A_0, A_1, ..., A_n, x \in E$. They laid down two types of neural networks for approximating the solution related to (2.4), namely feedforward (static) as well as recurrent (dynamic) models. The corresponding algorithm related to both neural networks is based on the least mean square. The difference between two neural networks states that dynamic neural network is superiorly robust than static neural network. The technique which is illustrated in [111] is sufficient to find approximate solution at par with special case of fuzzy equation, not generalized case.

The general fuzzy equation to be mentioned as dual fuzzy equation [213] has been illustrated in [110]. Normal fuzzy equations posses fuzzy numbers solely on one side of the equation. Whatsoever, dual fuzzy equations posses fuzzy numbers on both sides of the equation. As because it is not possible to move the fuzzy numbers in between the sides of the equation [119], dual fuzzy equations are superiorly generalized and complex. In [110] the existence of the solutions related to the dual fuzzy equations is analyzed, which is incorporated with the controllability problem associated to fuzzy control [65]. Afterward two kinds of neural networks for approximation of the solutions related to dual fuzzy equations have been demonstrated namely static and dynamic models.

2.2.2 Ranking method

The ranking methodology was primarily laid down by Delgado et al [71]. In [183] the researcher has obtained the real roots of polynomial equation which has been demonstrated as follows

$$C_1x + C_2x^2 + ... + C_nx^n = C_0$$

where $x \in R$ as well as $C_0, C_1, ..., C_n$ are taken to be fuzzy numbers. Fuzzy polynomial equation is converted to system of crisp polynomial equations. This conversion takes place with ranking method on the basis of three parameters Value, Ambiguity as well as Fuzziness. The obtained system of crisp polynomial equations is resolved numerically.

In [165] the conceptual content of a ranking method is suggested in order to extract the real roots associated to a dual fuzzy polynomial equation which has been illustrated as follows

$$A_1x \oplus A_2x^2 \oplus A_nx^n = B_1x \oplus B_2x^2 \oplus B_nx^n + d$$

where $x \in R$ as well as $A_1, ..., A_n, B_1, ..., B_n, d$ are denoted as fuzzy numbers. The dual fuzzy polynomial equations is converted to the system associated to the crisp dual polynomial equations. This conversion is carried out by utilizing ranking methodology on the basis of three parameters namely Value, Ambiguity and Fuzziness.

In [166] the real roots corresponding to the polynomial equation to be mentioned as $A_1x + A_2x^2 + ... + A_nx^n = A_0$ is obtained by utilizing the ranking method considering fuzzy numbers, where $x \in R$ as well as $A_0, A_1, ..., A_n$ are denoted as fuzzy numbers. In the quoted paper, the ranking methodology is utilized for real roots associated with dual polynomial equations as mentioned below

$$A_1x + A_2x^2 + ... + A_nx^n = B_1x + B_2x^2 + ... + B_nx^n + d$$

where $x \in R$, $A_1, ..., A_n, B_1, ..., B_n$ as well as d are considered as fuzzy numbers.

In [167] the ranking technique is implemented in order to obtain the real roots of an interval type-2 dual fuzzy polynomial equation $A_1x + A_2x^2 + ... + A_nx^n = B_1x + B_2x^2 + ... + B_nx^n + d$, where $x \in R$, the coefficients $A_1, ..., A_n, B_1, ..., B_n$ as well as d are termed as interval type-2 fuzzy numbers. Type-2 dual fuzzy polynomial equation is converted into a system at par with crisp type-2 dual fuzzy polynomial equation. The mentioned conversion is done by ranking method associated with fuzzy numbers on the basis of three parameters viz value, ambiguity and fuzziness.

It was revealed that solutions in correspond to three parameters such as Value, Ambiguity and Fuzziness are not sufficient to generate solutions. Henceforth in [168], a novel ranking methodology is suggested in order to eradicate the intrinsic weakness. The novel ranking methodology which is incorporated with four parameters is then implemented in the interval type-2 fuzzy polynomials, covering the interval type-2 of fuzzy polynomial equation, dual fuzzy polynomial equations as well as system of fuzzy polynomials. The effectiveness of the novel ranking methodology is numerically considered in the triangular fuzzy numbers as well as the trapezoidal fuzzy numbers.

2.2.3 Newton method, steepest descent method, Broyden's method and genetic algorithm method

In 1669, Isaac Newton introduced a novel algorithm [161] for solving a polynomial equation which was demonstrated on the basis of an example as $y^3 - 2y - 5 = 0$. To obtain a precise root of the mentioned equation, initially a starting value should be assumed, where $y \approx 2$. By assuming $y = 2+p$ and substituting it into the original equation, the following is obtained as $p^3 + 6p^2 + 10p - 1 = 0$. As p is presumed to be minute, $p^3 + 6p^2$ is neglected in comparison with $10p-1$, also the previous equation generates $p \approx 0.1$, so a superior approximation of the root is $y \approx 2.1$. The repetition of this process is feasible and $p = 0.1 + q$ is extracted, also the substitution deliver $q^3+6.3q^2+11.23q+0.061 = 0$, henceforth $q \approx -0.061/11.23 = -0.0054...$, so a novel approximation of the root is $y \approx 2.0946$. It is the requirement to repeat the process till the expected number of digits is achieved. In his methodology, Newton did not distinctly utilize the hypothesis of derivative but only applied it on polynomial equations.

In [8] the Newton's methodology is proposed in association with fuzzy nonlinear equations in lieu of standard analytical methodologies, as they are not appropriate throughout. The primary intention is to extract a solution for fuzzy nonlinear equation $F(x) = c$. Primarily the cited researchers have mentioned fuzzy nonlinear equation in parametric form as illustrated below

$$\begin{cases} \underline{F}(\underline{x}, \overline{x}, \alpha) = \underline{c}(\alpha) \\ \overline{F}(\underline{x}, \overline{x}, \alpha) = \overline{c}(\alpha) \end{cases}$$

so they resolved it by utilizing Newton's methodology.

In [25] the investigators have found the solution of

$$A_1 x \oplus A_2 x^2 \oplus ... \oplus A_n x^n = A_0$$

where $A_i, X^j \in E$ for $i = 1, ..., n$ $j = 0, 1, ..., n$. The fuzzy quantities are demonstrated in parametric form. The primary initiative is based on the conversion of the polynomial fuzzy coefficients into parametric form, thereby implementing Newton's technique on each limit. In the final phase, for finding the root, which is considered to be fuzzy number, the α-level sets of fuzzy coefficients on each limits are computed numerically.

In [3] some effective numerical algorithms in order to solve nonlinear equation $f(x) = 0$ on the basis of Newton–Raphson methodology is demonstrated. The modified Adomian decomposition methodology is implemented for developing the numerical algorithms.

In [93] the iterative methods are illustrated to obtain a simple root δ, i.e. $f(\delta) = 0$ as well as $f'(\delta) \neq 0$, of a nonlinear equation $f(x) = 0$. The authors have been mentioned the construction of some higher-order rectifications of Newton's method in order to resolve nonlinear equations that maximize the convergence order of prevailing iterative methodologies by one, two or three units. The mentioned construction can be implemented to any iteration formula as well as per iteration. The resulting methodologies sum up only one additional function evaluation in order to maximize the order. This enables the computational effectivity superior. This scheme can be endlessly employed for improving any prevailing iteration formula.

The privilege of the Newton's methodology is due to the convergency speed, once an adequately precise approximation is known. A drawback of this methodology is that a precise initial approximation in concerned with the solution is required to validate convergency.

In [10], a numerical solution associated with fuzzy nonlinear equation $F(x) = 0$ is suggested using steepest descent technique, where the fuzzy quantities are demonstrated in parametric form. The equation is represented by parametric form as mentioned below

$$\begin{cases} \underline{F}(\underline{x}, \overline{x}, \alpha) = 0 \\ \overline{F}(\underline{x}, \overline{x}, \alpha) = 0 \end{cases}$$

The function $G : R^2 \to R$ is stated by

$$G(\underline{x}, \overline{x}) = [\underline{F}(\underline{x}, \overline{x}, \alpha), \overline{F}(\underline{x}, \overline{x}, \alpha)]^2$$

The technique of steepest descent characterizes a local minimum considering two-variable function G. The technique of steepest descent is instinctively stated as:

1. Finding out G at an initial approximation $X_0^\alpha = (\underline{x}_0^\alpha, \overline{x}_0^\alpha)$.
2. Determine a direction from $X_0^\alpha = (\underline{x}_0^\alpha, \overline{x}_0^\alpha)$ which causes a decrease in the value of G.
3. Shift a suitable amount in this direction and consider the new value $X_1^\alpha = (\underline{x}_1^\alpha, \overline{x}_1^\alpha)$.
4. Repeat sequence 1 via 3 along with X_0^α replaced by X_1^α.

The steepest descent technique approaches only linearly to the solution, but in general it will approach even for weak initial approximations [59]. Even though steepest descent technique does not need a superior initial value, its drawback is due to its low convergency speed.

The drawbacks of Newton's technique is due to the calculation and inversion of the Jacobian matrix $J(x)$ at each iteration. The rapid convergence of Newton's technique is possible when an appropriate initial value is achieved. However, it is difficult to extract this kind of value, also this technique is relatively expensive to implement [60]. The steepest descent technique is not relied on a superior initial value. Its flaw is related to the low speed of convergency. Broyden's methodology is suggested to resolve this kind of equation. Broyden's methodology is leaned towards superlinearly convergency. This methodology is selected since it is superiorly alternative in comparison to Newton's technique, also it minimizes the amount of computation at each iteration without remarkably demeaning the speed of convergency. It substitutes the matrix A_{k-1} whose inverse is directly evaluated at each iteration in place of the Jacobian matrix J, also it minimizes the arithmetic operation $O(n^3)$ to $O(n^2)$ [60]. In lieu of utilizing standard analytical methods, such as Buckley and Qu methods, which are not appropriate for resolving a system of fuzzy nonlinear equations taking into consideration that the coefficient is fuzzy number, Broyden's technique is suggested for resolving fuzzy nonlinear equations. In [181], Broyden's technique is implemented in order to solve fuzzy nonlinear equations. Initially, fuzzy nonlinear equations are displayed in parametric form, also they are resolved by utilizing the Broyden's technique. The suggested eight steps algorithm results in the solution having maximum error which is less than 10^{-5}.

A genetic algorithms methodology for resolving the linear and quadratic fuzzy equations $Ax = B$ as well as $Ax^2 + Bx = C$, where A, B, C and x are considered to be fuzzy numbers is mentioned in [143]. The methodology based on the genetic algorithms primarily begins with a set of random fuzzy solutions. After that in each generation of genetic algorithms, the solution candidates converge to the superior fuzzy solution. In the suggested methodology the final attained solution is not only restricted to fuzzy triangular but also it can be a fuzzy number. In the mentioned methodology, in order to obtain the best fuzzy solution associated to a fuzzy equation, initially a solution is required to be converted to its chromosome demonstration. The solution candidates related to the fuzzy equations are transformed to their level sets demonstration which are computable by genetic algorithms. Figure 2.1 displayes a level set structure associated with the chromosome in concerned to a potential solution.

A genetic algorithm for resolving the fuzzy equation $P(x) = y$ is demonstrated in [58], where x and y are considered to be k-sampled real fuzzy numbers, also P is taken to be a

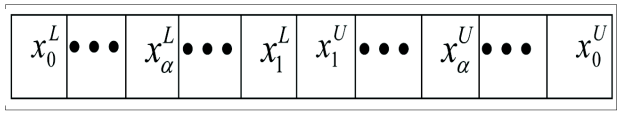

Figure 2.1: Chromosome representation of a potential solution x for fuzzy equation

fuzzy function relying on x. The motive is to obtain a suitable value associated with the fuzzy argument x in such a manner that the calculated value of the polynomial, $P(x)$, is very much adjacent to the supplied target value y. The presented genetic algorithm utilized a distinct demonstration of the fuzzy numbers which permits the implementation of simple genetic operators. The algorithm is self sufficient for finding multiple solutions associated with the fuzzy equations. Regrettably, no method has been utilized for an identical problem involved in the area of neural networks that can be taken possession of. Because of the distinct discrete criteria of the fuzzy arithmetic, the single realistic approach for resolving this problem is to design a committed genetic algorithm [149].

In [134] genetic algorithms are implemented for solving fuzzy equations without stating membership functions related to fuzzy numbers, also it has not used the extension principle as well as interval arithmetic, α-cut operations and a penalty technique in order to constraint violations. An important matter for using genetic algorithms in order to extract a better solution associated with the problem is the parameter settings that includes the probability of crossover, the probability of mutation as well as the number of generations. The fuzzy conception related to the genetic algorithm scheme is contrasting, but generates superior solutions in comparison with classical fuzzy techniques.

2.2.4 exponent to production method

In [26] the exponent to production technique is illustrated in order to generate an analytical and approximated solution related to fully fuzzy quadratic equation (FFQE) $F(X) = D$, where $F(X) = AX^2 + BX + C$. In the mentioned technique, the 2nd exponent of a fuzzy number is undergone a conversion into product of two fuzzy numbers which is in parametric form. By utilizing this sort of conversion, an analytical as well as approximated solution associated to a FFQE is extracted. The optimum spreads are revealed in order to minimize maximum error. In exponent to production technique, initially the 1-cut solution of FFQE at par with the real root is found out and so undetermined manipulated unsymmetrical spreads are subjected to the core point. In this case λ and μ have been extracted as optimum values that develops the superior spreads. This technique resolves some problems which are barred of analytical solution, also it is an benefit of exponent to production technique. Since numerical techniques require initial assumption for continuation, so the mentioned technique generates this requirement. In addition, in this technique the complexities do not rely on the sign of the coefficients as well as variable.

2.2.5 Adomian decomposition method

In [2] the standard Adomian decomposition is implemented on simple iteration technique in order to resolve the equation $f(x) = 0$, where $f(x)$ is a nonlinear function, also it has proved the convergency related to the series solution. Initially the nonlinear equation is transformed into canonical form, after that the Adomian technique computes the solution which is at par with the series form. As practically all the terms associated with the series are not possible to determine, hence the estimation of the solution from the truncated series has been accomplished. Therefore, the convergency related to the truncated series is usually very rapid.

Babolian et al. [34] altered the standard Adomian technique mentioned in [2] in order to solve nonlinear equation $f(x) = 0$ for acquiring a sequence of approximations related to the solution, with approximate superlinear convergency. They have employed Cherruault's definition [66] and took into consideration the order of convergency related to the technique [35].

In [171] a potential numerical algorithm in order to solve fuzzy polynomial equations $\sum_{i=1}^{n} a_i x^i = c$ on the basis of Newton's technique is demonstrated, where x and c are considered to be fuzzy numbers, also all coefficients are taken to be fuzzy numbers. The modified Adomian decomposition methodology is implemented for the construction of the numerical algorithm. Primarily the fuzzy polynomials are illustrated in a parametric form and finally they have been resolved using Adomian decomposition technique.

In [208] the Shanks transformation is employed on the Adomian decomposition technique in order to resolve nonlinear equations so as to improvise the preciseness of the approximate solutions. The numerical results demonstrate that the the implementation of this technique in similar conditions generates more appropriate solutions in concerned to the nonlinear equations when compared with those extracted from the Adomian decomposition technique. The Shanks transform is an effective approach which can speed up the convergency rate of the series. Adomian generates the solution of infinite series generally converging to a precise solution.

2.2.6 Fuzzy linear regression model

Generally, there exist two techniques in fuzzy regression analysis namely linear programming based technique [176][185][203][204] and fuzzy least squares technique [184][73]. The primary technique is relied on diminishing fuzziness at par with optimal critera. The secondary technique utilizes least square errors at par with fitting criteria. As illustrated in [212], the benefit of primary technique is its simplicity associated to programming as well as calculation, whereas in the fuzzy least squares technique is its minimal degree of fuzziness in the midst of the observed and approximated values. Currently, the least of the total squares errors associated with the spread values are utilized as the fitting criteria as well as advanced mathematical programming methodology in such a manner that the predictability of the primary technique can be improvised and the calculation complication related to the secondary technique can be minimized [157].

Fuzzy linear regression was initially put up by Tanaka et al. [204]. The main intension was to diminish the total spread of the fuzzy parameters related to the support of the approximated values which enclose the support of the observed values at par with a certain α-level. Even though this concept was lately modified by Tanaka et al. [203], their model is termed to be very responsive to outliers. However, it can generate infinite solutions as well as the spread of the approximated values, since it becomes wider as more data are piled up in the model.

In [185] a fuzzy linear regression model is generated in the form of $Y_i = A_0 + A_1 x_i$ at par with fuzzy output as well as fuzzy parameters taking into consideration mathematical programming problem by utilizing three indices in concerned with equalities between fuzzy numbers. Three patterns of multiobjective programming problems in order to extract fuzzy linear regression models are laid down related to the three indices. A linear programming relied on interactive decision making method in order to extract the convenient solution at par with the decision making for formulating the multiobjective programming problems is stated. The technique implied in [185] can generates an infinite number of solutions via repeated observations. Therefore, the mentioned technique is able to generate crisp coefficients. By the repeated observations this technique results in redundant constraints. Hence, all observations cannot contribute to the computation associated with the model. In [185], fuzzy linear regression undergoes from crisp coefficients, redundant constraints as well as the possibility related to an infinite number of solutions. To deal with the possibility related to an infinite number of solutions, the approximation point of the centers at par with the fuzzy coefficients can be computed using the available data which is implemented into fuzzy linear regression algorithms. In [188] it is displayed that the least squares technique can be utilized as point approximation to center the fuzzy coefficients which is employed in the Tanaka techniques. Two highly advantages are linked to the use of point approximation to center the fuzzy coefficients. Initially, if the researcher selects point approximation which is distinctively defined, hence possibility of an infinite number of solutions is eradicated. Secondly, point approximations permit all data points to contribute information in the fuzzy linear algorithm. Hence, under repeated observations, the utilization of point approximations associated with the center of the fuzzy coefficients tackles some of the problems imparted by redundant constraints.

Nasrabadi et al. [158] utilized a multi-objective programming concept in order to lay down the linear regression coefficients $Y_i = A_0 + A_1 X_i + ... + A_n X_{in}$, $i = 1, ..., m$ where

$X_{ij} = (x_{ij}, r_{ij})$, $A = (a_j, \alpha_j)$ as well as $Y_i = (y_i, \beta_i)$ are considered to be symmetric fuzzy numbers. In the mentioned fuzzy regression work, powerful predictions are developed on the basis of the fuzzy number parameters.

In [154] the fuzzy linear regression $Y_i = A_0 \oplus A_1 x_{i1} \oplus A_2 x_{i2} \oplus ... \oplus A_n x_{in}$ as well as fuzzy polynomial regression $Y_i = A_{l0} \oplus \sum_{j=1}^{n} A_{lj} x_{ij} \oplus \sum_{j=1}^{n} \sum_{k=1}^{n} A_{ljk} x_{ij} x_{ik} \oplus ...$, where input units are taken to be crisp numbers and output unit is taken to be a fuzzy number are mentioned. The proposed technique is relied on neural network model in order to extract the estimation of the regression coefficients. More generalized pattern of fuzzy polynomial regression has been revealed in [170].

In [170] a polynomial fuzzy regression model at par with fuzzy independent variables as well as fuzzy parameters based on the following form is illustrated as

$$Y_i = A_{l0} \oplus \sum_{j=1}^{n} A_{lj} X_{ij} \oplus \sum_{j=1}^{n} \sum_{k=1}^{n} A_{ljk} X_{ij} X_{ik} \oplus ...$$

where i denotes the different observations, $X_{i1}, X_{i2}, ..., X_{in}$, are coefficients as well as Y_i are considered to be fuzzy numbers. A fuzzy neural network model is utilized for extraction of an estimate related to the fuzzy parameters along a statistical sense. This technique permits the development of nonlinear regression models along with general fuzzy number inputs, outputs and parameters. The suggested technique consists of numerable properties. Initially, it can use non-triangular fuzzy observations. Furthermore, the fuzzy neural network technique carries out perfect, with respect to the sum of squared errors as well as the accuracy in approximation.

In order to obtain a fuzzy polynomial interpolation having degree n, it is essential to extract n fuzzy coefficients. So as to find these coefficients, it is a requirement to resolve $2n \times 2n$ equations which is very complicated in terms of large values of n, also sometimes it is barred of a fuzzy solution, for more details refer [85][86]. In [141], an innovative estimation algorithm for fuzzy polynomial interpolation by utilizing Artificial Bee Colony algorithm for interpolating fuzzy data is demonstrated. It is assumed that $X = \{x_1, ..., x_n\}$ be a set of m distinct points associated with R, also $F = \{y_1, ..., y_n\}$ be the value of a triangular fuzzy function f at the point x_i, $i = 1, ..., n$. The below mentioned polynomial of m degree is taken into account

$$p_m(x) = \sum_{j=0}^{m} a_j x^j = \sum_{j=0}^{m} (\underline{a}_j(r), \overline{a}_j(r)) x^j$$

where a_j is a trapezoidal fuzzy number at par with parametric form $(\underline{a}_j(r), \overline{a}_j(r))$ for $j = 0, 1, ..., m$. The experimental data are considered to be $(x_1, y_1), (x_2, y_2), (x_n, y_n)$, $x_i \in X$ and $y_i \in F$ (given that $n > m + 1$).

2.2.7 Algebraic fuzzy equations

Some investigations have been carried out on algebraic fuzzy equations, but the existing methodologies calculate the roots of an algebraic fuzzy equation analytically, also there exist no analytical solution related to algebraic fuzzy equations having degree greater than 3 [50]. Thus by utilizing numerical methodologies in concerned to such equations is utter essential. In [31] the author presented an algebraic fuzzy equation having degree n including fuzzy coefficients as well as crisp variable which is stated by

$$a_n x^n + ... + a_1 x + a_0 = 0 \tag{2.5}$$

in which $0, a_0, a_1, ..., a_n \in E$ and $a_n \neq 0$. i.e., also $P_n(x) = 0$, where $P_n(x) = \sum_{j=0}^{n} a_j x^j$ is mentioned as a polynomial of degree n. In order to determine the roots of the stated algebraic fuzzy equation an algorithm on the basis of Gauss-Newton technique produces a series, that can converge under the condition that the modal value function includes a root, or else the series can diverge. In addition, suppose that the equation contains more than one root, the roots can be obtained via different initial vectors. In order to resolve the algebraic fuzzy equation, the root as well as fuzzy zero are assumed to be unknowns, therefore by a series they can be determined. In case that a fuzzy zero is provided, then only the root of algebraic fuzzy equation should be extracted.

Consider the equations mentioned below

$$F(X) - B = 0, \quad F(X) = B$$

where B is taken to be as an interval or fuzzy value, $F(X)$ is taken to be some interval or fuzzy function. The above displayed equations are not termed to be equivalent. However, the crucial problem is linked with the conventional interval or fuzzy extension of the usual equation, which results in the interval or fuzzy equation to be mentioned as $F(X) - B = 0$. An interval exists on the left hand side of this elongated equation, whereas a real valued zero exists on the right hand side. As it is not possible for the interval to be equal with the real value, this result is termed as "interval equation's right hand side problem". Minimal problems will outcome while dealing with interval or fuzzy equations as $F(X) = B$, but in many issues its roots are termed as inverted intervals, i.e., in such a manner that $\overline{x} < \underline{x}$. The detail analysis of this fact has been mentioned in [190].

The linear equation consider in [190] is mentioned as

$$ax = b \tag{2.6}$$

where its algebraically equivalent forms are denoted as

$$x = \frac{b}{a} \tag{2.7}$$

$$ax - b = 0 \tag{2.8}$$

considering a, b to be intervals. Suppose $[a] = (\underline{a}, \overline{a})$ as well as $[b] = (\underline{b}, \overline{b})$ be intervals., hence considering the case $[a] > 0$, $[b] > 0$, i.e., $\underline{a}, \overline{a} > 0$ and $\underline{b}, \overline{b} > 0$, the interval extension of (2.6) is given by $(\underline{a}, \overline{a})(\underline{x}, \overline{x}) = (\underline{b}, \overline{b})$. It can be portrayed as $(\underline{ax}, \overline{ax}) = (\underline{b}, \overline{b})$. It is quiet evident that the equivalence of the right as well as left hand sides of the mentioned equation is feasible only if $\underline{ax} = \underline{b}$ and $\overline{ax} = \overline{b}$, which are illustrated as

$$\underline{x} = \frac{\underline{b}}{\underline{a}}, \quad \overline{x} = \frac{\overline{b}}{\overline{a}} \tag{2.9}$$

Interval extension of (2.7) can be regarded as below

$$\underline{x} = \frac{\underline{b}}{\overline{a}}, \quad \overline{x} = \frac{\overline{b}}{\underline{a}} \tag{2.10}$$

The suggested methodology can be utilized solely in the uncomplicated cases of linear equations. Generally, the choosing of the suitable interval or fuzzy extensions in concerned to nonlinear case is a complicated task.

Example 2.2 *Assume* $[a] = (3, 4)$, $[b] = (1, 2)$. *Hence from (2.9) it can be extracted* $\underline{x} = 0.333$, $\overline{x} = 0.5$, *also from (2.10)* $\underline{x} = 0.25$, $\overline{x} = 0.666$.

In [12] the decomposition methodology has been utilized for quadratic, cubic as well as generalized higher-order polynomial equations and negative, or nonintegral powers and random algebraic equations. The algebraic equations can be dealt by using the decomposition methodology and it supplies a crucial methodology in order to calculate the roots of polynomial equations usually resulting a very fast convergence. This methodology generally converges towards a precise solution.

In [36] a novel algorithm on the basis of the Adomian methodology is demonstrated in order to resolve algebraic equations. This modernized algorithm computes the superior estimations related to the exact solution of algebraic equations, when compared with the standard Adomian methodology. A nonlinear equation is considered as follows

$$F(x) = 0 \tag{2.11}$$

that can be transformed to

$$x = F_0(x) + c_0 \tag{2.12}$$

where F_0 is taken to be a nonlinear function, also c_0 is a constant. The Adomian methodology calculates x as a series

$$x = \sum_{i=0}^{\infty} x_i \tag{2.13}$$

The decomposition of nonlinear function is illustrated below

$$F(x) = \sum_{i=0}^{\infty} A_i \tag{2.14}$$

where A_i's are considered to be Adomian polynomials stated by

$$A_n(x_0, ..., x_n) = (\frac{1}{n!})(\frac{d^n}{d\lambda^n})F(\sum x_i\lambda^i)|_{\lambda=0} \tag{2.15}$$

Substituting (2.13) as well as (2.14) to (2.12) results in

$$\sum_{i=0}^{\infty} x_i = \sum_{i=0}^{\infty} A_i + c_0$$

each term of the series $x = \sum_{i=0}^{\infty} x_i$, at par with the Adomian method, can be computed using the relations mentioned below

$$
\begin{aligned}
x_0 &= c_0 \\
x_1 &= A_0 \\
x_2 &= A_1 \\
&. \\
&. \\
&. \\
x_n &= A_{n-1}
\end{aligned}
$$

In calculation of x by utilizing any software, since n increases the number of terms in the expression related to A_n increases and this results in the dissemination of round off errors. Also, the factor $\frac{1}{n!}$ mentioned in the formula related to A_n makes it superiorly minute, hence its contribution to x is not taken into account, therefore, the primary few terms related to the series $\sum_{i=0}^{\infty} x_i$ state the preciseness of the estimated solution. By taking into consideration of this concept, [36] laid down a novel algorithm on the basis of the Adomian methodology in order to improvise the preciseness dramatically.

2.3 Fuzzy partial differential equations and fuzzy differential equations

2.3.1 Fuzzy neural network method

In [1] a technique is introduced in order to resolve PDEs with boundary and initial conditions by imparting neural networks. An evolutionary algorithm is employed for training the networks. The outcomes of implementing the methodology to a one-dimensional as well as a two-dimensional problem are highly superior and convincing.

Whatsoever the principle concept is that evolutionary algorithms uncover all regions of the solution space as well as exploit favorable areas via implementing recombination, mutation, selection and reinsertion operations to the individuals of a population. In [1] the researchers worked considering a single population. In [177] it is illustrated that single population evolutionary algorithms are strong and out performs on a broad variation of problems. Nevertheless outcomes are highly effective which are extracted while working with multiple subpopulations in lieu of just a single population.

In [130] a technique in order to resolve both ODEs and PDEs is presented and is dependent on the function approximation abilities of feedforward neural networks, which results in the development of solution presented in a differentiable and closed analytic form. This form applies a feedforward neural network as the general estimation element, that its parameters (weights and biases) are tweaked to diminish a suitable error function. In order to train the network, optimization methodologies have implemented, that need the calculation of the gradient error considering the network parameters. In the suggested methodology the model function is presented as the sum of two terms. The first term suffices the initial/boundary conditions, also does not includes adjustable parameters. The second term includes a feedforward neural network to be trained in order to suffice the differential equation. The implementation of a neural architecture sums up several attractive features to the technique:

1- The solution through artificial neural network is differentiable having closed analytic form which is easily utilized in any subsequent computation. Most other methodologies suggest a discrete solution (viz predictor-corrector or Runge–Kutta methodologies) or a solution of limited differentiability (viz finite elements).

2- The implementation of neural networks supplies a solution with highly superior generalized attributes. Compared results with the finite element methodology depicted in this work describe this point vividly.

3- The required number of model parameters is very much less than any other method and hence, compact solution models are extracted with low demand criteria on memory space.

4- The technique is simple and can be implemented to ODEs, systems of ODEs and also to PDEs stated on orthogonal box boundaries. However, the process is in advancement to rectify the case of irregular (arbitrarily shaped) boundaries.

5- The technique can be tested in hardware, utilizing neuroprocessors, and also it proposes the chance to handle real-time complex differential equation problems that occur in several engineering applications.

6- The technique can also be effectively imposed on parallel architectures.

This technique is simple and can be employed to both ODEs as well as PDEs by developing the suitable form of the trial solution. The technique displays superior generalization performance as the deviation at the test points is in no case major than the maximum deviation at the training points. This is in contrast with the finite element technique in the case that the deviation at the testing points is extremely higher in comparison with the deviation at the training points.

The case which requires to be inspected is at par with the sampling of the grid points which are utilized for training. In [130] the grid was developed in a general way by taking into account equidistant points. It is expected that superior outcomes can be extracted considering the case in which the grid density will differ while training at par with the corresponding error values. This signifies that it is feasible to consider more training points at regions where the error values are more. Also the suggested methodologies can easily employ to the domains of higher dimensions (three or more). Since the dimensionality increases, the number of training points increases. This leads to a grievous problem for techniques which take into account local functions around each grid point as the needed number of parameters becomes extensively higher and hence, both memory as well as calculation time requirements become severely high. If we consider the case of the neural technique, the number of training parameters stays almost firm since the problem dimensionality increases. The only effect on the calculation time evolves from the fact that each training pass needs the demonstration of more points, i.e., the training set becomes larger. This problem can be handled by taking into consideration either parallel implementations or implementations on a neuroprocessor which can be inscribed in a traditional machine and supply sufficiently superior execution times.

In [105] a modified technique is proposed in order to obtain the numerical solutions of fuzzy PDEs by utilizing fuzzy artificial neural networks. Utilizing improvised fuzzy neural network ensure that the training points get selected over an open interval without training the network in the range of first and end points. This novel technique is on the basis of substituting each x in the training set (where $x \in [a,b]$) by the polynomial $Q(x) = \epsilon(x+1)$ in such a manner that $Q(x) \in (a,b)$, by selecting an appropriate $\epsilon \in (0,1)$. Also, it can be suggested that the proposed methodology can deal efficiently all types of fuzzy PDEs as well as to supply precise estimation solution entirely for all domain and not only at the training set. Hence, one can utilize the interpolation methodologies (to be mentioned as curve fitting methodology) in order to obtain the estimated solution at points in the midst of the training points or at points outside the training set.

In [77] a new technique is proposed in order to solve FDEs having initial conditions on the basis of the utilization of feed-forward neural networks. Initially, the FDE is substituted by a system of ODEs. The trial solution related to the system is stated as an addition of two parts. The first phase suffices the initial condition, also does not have adjustable parameters. The second phase includes a feed-forward neural network having adjustable parameters (the weights). Therefore by development, the initial condition is sufficed, also the network is trained to suffice the differential equations. In this novel scheme, the inputs of the neural network are considered to be as the training points. This technique supplies solutions with superior generalization as well as high preciseness.

In [156] a technique for estimating solution of a second order FDE is suggested by utilizing fuzzy neural network on the basis of back-propagation-type learning algorithms. The employ of more simplifies network architectures causes the back-propagation-type learning algorithm highly complex. In [156] a notable simulation results from partially fuzzy neural network is demonstrated but the researchers have not elaborated their learning algorithm to neural network having more than three layers.

In [103] a novel technique on the basis of learning algorithm associated with fuzzy neural network as well as Taylor series is laid down for extracting numerical solution of FDEs. A fuzzy neural network on the basis of the semi-Taylor series (in concerned to the function e^x) for the first (and second) order FDE is utilized. It is possible to use the same approach for solving high order FDE as well as fuzzy PDE. A fuzzy trial solution related to the fuzzy initial value problem is presented as an addition of two parts. The primary phase suffices the fuzzy initial condition and it includes Taylor series, also contains no fuzzy adjustable parameters. The secondary phase includes a feed-forward fuzzy neural network having fuzzy adjustable parameters (the fuzzy weights). Therefore by development, the fuzzy primery condition is sufficed and the training of fuzzy network is carried out in order to suffice the FDE. The preciseness of this technique is relied on the Taylor series that is selected for the trial solution. This selection is not distinct, hence, the preciseness is different from one problem to another problem. The suggested technique gives more precise estimation. Superior outcome will be possible if more neurons or more training points are used. However, after resolving a FDE the solution is achievable at any arbitrary point in the training interval (even in the midst training points).

In [104] a hybrid scheme on the basis of the modified fuzzy neural network as well as optimization method is proposed for resolving FDEs. Utilizing modified fuzzy neural network leads to the fact that the training points should be chosen over an open interval without training the network in the range of first and end points. hence, the computing volume which includes calculation error is minimized. Actually, the training points that relied on the distance chosen for training neural network are transformed to similar points in the open interval by incorporating a novel strategy, so the network is trained in these similar field. The suggested model produces solutions with high preciseness. Modified fuzzy neural networks contain high ability in function estimation.

In [155] a new hybrid technique on the basis of learning algorithm of fuzzy neural network for extracting the solution of differential equation with fuzzy initial value is demonstrated. The model obtains the estimated solution of FDE inside of its domain for the close by neighborhood of the fuzzy initial point. One lack of fully fuzzy neural networks along with fuzzy connection weights is long calculation time. One more lack is that the learning algorithm is complex. In order to minimize the severity of the learning algorithm, in [155] a partially fuzzy neural network architecture is demonstrated by taking into account that the connection weights to output unit are fuzzy numbers whereas connection weights and biases to hidden units are real numbers. By using the other learning algorithms, different simulation outcomes can be achieved. For example, some global learning algorithms to be mentioned as genetic algorithms can train non-fuzzy connection weights more superiorly than the back-propagation-type learning algorithm for fuzzy mappings of triangular-shape fuzzy numbers having incrementing fuzziness. Also the utilization of more general network architectures causes the back-propagation-type learning algorithm to be more complex.

2.3.2 Euler method

In [139], the FDE is substituted by its parametric form. The classical Euler technique is implemented for resolving the novel system that contains two classical ODEs having initial conditions. The capacity of technique is demonstrated by resolving several linear as well as nonlinear first-order FDEs.

In [162], a linear first-order FDE by employing the strongly generalized differentiability approach is analyzed. This work is based on the Generalized Characterization Theorem in which the FDE is substituted with its equivalent systems and so for estimating the two fuzzy solutions, two ODE systems are resolved by using generalized Euler approximation technique that includes four classic ODEs having initial conditions. Furthermore, the error analysis of the generalized Euler technique that assures pointwise convergency is laid down. The significance of transforming a FDE into a system of ODEs is based on the fact that any numerical technique which are appropriate for ODEs can be employed.

In [16], a novel fuzzy version of Euler's methodology in order to solve differential equations having fuzzy initial values is suggested. The suggested methodology is relied on Zadeh's extension principle in order to reformulate the classical Euler's methodology, that considers the dependency problem which is generated in fuzzy setting. This problem is regularly avoided in the numerical methodologies which is included in the literature in order to resolve differential equations having fuzzy initial values. This paper has positive attributes when compared with the traditional fuzzy version of Euler's methodology. At par with the [48], taking into consideration non-dependency problem associated with fuzzy calculation will result in repetition of some numerical calculations. Therefore, there prevails expected errors, also in the last phase the errors may generate estimations which are broader in comparison with the correction. This is authentic as the initial results carried out in [17] have demonstrated that the solution of FDEs extracted by utilizing the suggested methodology in [139] has overestimation in calculation.

In [206] two improvised Euler type methodologies to be mentioned as Max-Min improved Euler methodology and average improved Euler methodology are suggested for extracting numerical solution of linear as well as nonlinear ODEs at par with fuzzy initial condition. In this paper all the possible blends of lower as well as upper bounds in concerned with the variable are considered and then resolved by the suggested methodologies. Also, an exact method is laid down.

In [205] the numerical solution associated with linear, non-linear as well as system of ODEs having fuzzy initial condition is researched. Two Euler type methodologies namely Max-Min Euler methodology and average Euler methodology are laid down for extracting numerical solution related to the FDEs. Several investigators in their works have considered the left and right bounds of the variables at par with the differential equations. In this research work, the investigators constructed the methodologies by taking into account all possible combinations of lower as well as upper bounds of the variable. The solution extracted by Max-Min Euler methodology very closely resembles with the outcomes extracted by [139] and exact solution.

2.3.3 Taylor method

In [150], differential transform technique is different from the conventional high-order Taylor series technique, that needs symbolic calculation of essential derivatives of the data function, also is computationally costly in concerned to higher order. In the mentioned paper, FDTM is suggested for resolving fuzzy PDE. A few examples were analyzed by utilizing FDTM, also the outcomes exhibits significant performance.

In [6], an approach on the basis of the 2nd Taylor technique is illustrated in order to resolve linear as well as nonlinear FDEs. The convergence order of the Euler technique in [139] is $O(h)$, whereas the convergence order in [6] is $O(h^2)$. The better solutions are extracted by [6].

In [5], a concept on the basis of the Taylor technique of order p is demonstrated thoroughly, also this is followed by a complete error analysis. The solutions have superior preciseness in this paper.

2.3.4 Differential transform method

In [150], a two-dimensional differential transform methodology of fixed grid size is employed to obtain approximate solutions related to fuzzy PDEs. Also an adaptive grid size mechanism on the basis of the fixed grid size methodology is laid down. The suggested methodologies generate the Taylor series expansion solution for the domain between any adjacent grid points. This methodology is effective for extracting exact and approximate solution of linear, nonlinear ordinary as well as for fuzzy PDEs. The differential transform methodology provides an analytical solution in the polynomial form. Differential transform methodology converts the PDEs as well as related initial conditions into a recurrence equation which ultimately results in the solution of a system related to algebraic equations as coefficients of a power series solution. It is varied from the conventional high-order Taylor series methodology, that needs symbolic calculation of the essential derivatives of the data functions, also is computationally costly for high order. The fuzzy differential transform methodology finds out the estimating solution by utilizing finite Taylor series. The fuzzy differential transform methodology is bared from evaluating the derivatives symbolically, whereas it computes the relative derivatives by utilizing an iteration technique stated by the fuzzy transformed equations extracted from the original equations.

In [28], an elongation of the differential transformation methodology in order to resolve the FDE is proposed. A different types of exact, approximate, and purely numerical methodologies are given to extract the solution related to a fuzzy initial value problem. The investigators have generally gone through the methodologies on the basis of the Hukuhara derivative. Whatsoever, in some situations this concept undergoes a severe drawback. The solution includes a property that dim $(x(t))$ is non-decreasing in t, i.e. the solution is irreversible in possibilistic terms. Hence, this explanation is not an appropriate generalization of the linked crisp case. It is considered that this problem is the outcome of fuzzification related to the the derivative employed in the development of the FDE. Additionally, most known methodologies of resolving FDEs are computationally intense, since they are trial-and-error in nature or require complex symbolic calculations. In [28], the authors have solved these difficulties by utilizing a more simple definition of the derivative at par with the fuzzy mappings, elaborating the class of differentiable fuzzy mappings which are laid down in [41][42], and then utilized differential transformation methodology for solving FDEs.

In [38], a generalization of differential transformation methodology in order to solve the fuzzy PDE by utilizing the strongly generalized differentiability approach is researched.

2.3.5 Nystr$\ddot{o}$m method

In [125], a numerical approach on the basis of Nystr$\ddot{o}$m technique for resolving fuzzy first-order initial value problem is suggested. Sufficient conditions for the validation of stability and convergency of the suggested algorithms are supplied. The illustrated technique is Convergent as well as stable.

In [191] the Nystr$\ddot{o}$m technique is constructed for estimating the solutions related to hybrid FDE initial value problems by utilizing the Seikkala derivative. The differential equations includes fuzzy valued functions as well as interaction having a discrete time controller termed as hybrid FDEs [175].

2.3.6 Predictor-corrector method

In [23], three numerical methodologies in order to resolve fuzzy ODEs are proposed. These methodologies are Adams–Bashforth, Adams–Moulton and predictor–corrector. Predictor–corrector is extracted by blending Adams–Bashforth as well as Adams–Moulton methodologies. The suggested methodologies are Convergent and stable. Considering the convergence order of the Euler methodology which is O(h) (as given in [139]), a higher order of convergency is achievable by utilizing the suggested methodologies in [23], to be mentioned that a predictor–corrector methodology of convergence order $O(h^m)$ is utilized where the Adams–Bashforth m-step methodology and Adams–Moulton $(m-1)$-step methodology are taken to be as predictor and corrector, respectively. By going with the ideas of [182], the suggested methodologies in [23] can resolve the stiff problems.

In [40], it has shown that the exact solutions demonstrated in [23], are not solutions associated with the FDEs and the correct exact solutions related to these problems is illustrated. Two characterization theorems are laid down for the solutions of FDEs that are employed for converting a FDE into a system of ODEs. The characterization theorems reveal the following investigation direction for FDEs. Convert the FDE into a system of ODEs, afterward resolve the system of ODEs, as it is feasible to go back to the original FDE. This scheme can be employed with other differentiability concepts which include more natural attributes in comparison with the Hukuhara derivative.

In [127] a numerical solution in concerned with hybrid FDE is researched. The improved predictor–corrector methodology is selected and altered in order to resolve the hybrid FDEs on the basis of the Hukuhara derivative. The symbolic systems associated with the computer to be mentioned as Maple and Mathematica are employed to carry out complex computations of algorithm. It is displayed that the solutions extracted using predictor–corrector methodology is more precise and well matched with the exact solutions.

2.3.7 Runge-Kutta method

In [173] an effective s-stage Runge-Kutta technique is employed for extracting the numerical solution of FDE. Runge–Kutta method is applied for a more generalized category of problems and a convergence definition as well as error definitions are given at par with FDEs theory. Convergency related to s-stage Runge–Kutta methods is analyzed. This technique when compared with developed Euler technique performs superior. Although Euler technique is suitable, it is embedded with the disadvantage. When the convergence of Euler technique is analyzing [139], the authors generally investigate on the convergence of the ODEs system which takes place while resolving numerically.

In [7] a numerical algorithm in order to resolve linear as well as nonlinear fuzzy ODEs on the basis of Seikkala's derivative of fuzzy process is investigated. A numerical methodology on the basis of four-stage order Runge–Kutta technique is stated and is carried on by a complete error analysis. Whatsoever, their work generalizes the same problems which is mentioned in [139], also relies totally on four-stage methodologies.

In [115] a numerical algorithm in order to solve linear as well as nonlinear fuzzy ODE on the basis of Seikkala derivative of fuzzy process is suggested. A numerical technique on the basis of the Runge-Kutta methodology of order five is elaborately investigated and this is carried on by going through an analysis of complete error. This technique with $O(h^5)$ outperforms than improved Euler's technique with $O(h^2)$.

In [115] a numerical solution for nth-order FDEs on the basis of Seikkala derivative having initial value problem is investigated. The Runge-Kutta Nystrom technique is employed for extracting the numerical solution of this problem, also the convergency as well as stability related to the method is validated. In this methodology the nth-order FDE is transformed to a fuzzy system that can be resolved by utilizing the Runge-Kutta Nystrom methodology.

A family of extended Runge-Kutta-like formulae are implemented in [89]. The formula reveals the utilization of first derivatives f'. This approach uses an elaborated Runge-Kutta methodology involving local truncation error of order 5 with only four evaluations of both f as well as f' to approximate the local error in an elaborated Runge-Kutta methodology having order four per step. The positive attribute of this method is that just four evaluations of both f as well as f' are needed per step, also arbitrary classical Runge-Kutta methodologies of orders 3 and 4 employed in combination require six evaluations of f per step. In [89] a numerical algorithm in order to resolve the fuzzy first order initial value problem on the basis of elaborated Runge-Kutta-like formulae of order 4 is implemented. In this paper the elaborated Runge-Kutta-like formula is employed for enhancing the order of preciseness related to the solutions by evaluating both f and f', instead of only evaluating f.

In [120] a numerical algorithm in order to solve FDEs on the basis of Seikkala's derivative including a fuzzy process is suggested. A numerical technique based on a Runge-Kutta Nystrom technique of order three is employed for solving the initial value problem, also it is illustrated that this methodology is superior in comparison with the Euler method by considering the convergence order of Euler methodology ($O(h)$) as well as Runge-Kutta Nystrom methodology ($O(h^3)$).

2.3.8 Finite difference method

Finite difference methodologies illustrate functions as discrete values across a grid, also estimate their derivatives as differences between points on the grid. In [81] a numerical methodology for solving the fuzzy heat equation is laid down. A difference approach is taken into account for the one dimensional heat equation. Additionally the necessary conditions for stability of the suggested approach is illustrated. The suggested difference methodology associated with the example mentioned in [81] is tested when the exact solution is known. In this example the Hausdorff distance between exact solution and estimated solution is extracted.

In [133] finite difference methodologies for resolving differential equations is proposed. Furthermore, finite difference estimations to higher order derivatives are suggested.

In [160] an implicit finite difference methodology for resolving fuzzy PDEs is discussed. The stability related to this methodology is demonstrated and resolved the parabolic equation with this concept.

In [13] a numerical methodology on the basis of the seikkala derivative for resolving fuzzy PDE is taken into account. Difference methodology for solving the FPDEs to be mentioned as fuzzy hyperbolic equation as well as fuzzy parabolic equation is illustrated, beside that the stability associated with this methodology is analyzed and conditions for stability are supplied. If all terms of FPDE belong to E, so the solutions of FPDE prevails, that are concluded from the numerical values. Examples demonstrate that the Hausdorff distance between exact solution and approximate solution is minute.

In [106] it has been stated how to induce boundary conditions into finite difference methodologies so the resulting estimations copy the identities between the differential operators of vector as well as tensor calculus. The scheme is valid for wide class of PDE, also is stated for Poisson's equation with Dirichlet, Neumann as well as Robin boundary conditions. These estimations retain the crucial properties of original differential problems. Particularly, the discrete estimation is symmetric and positive definite. The properties associated with the discrete operators make feasible to utilize efficient iteration methodologies in order to resolve system of linear equations.

In [22] difference methodologies based on the seikkala derivative for resolving linear and nonlinear fuzzy PDEs is considered. Fuzzy reachable set can be estimated by suggested methodologies including complete error analysis. The methodologies are demonstrated by resolving three types of FPDE namely finite difference Poisson equation (the difference methodology by utilizing the Taylor series), backward difference heat equation (Crank-Nicolson methodology) and finite difference wave equation.

In [123] an implicit finite difference methodology is discussed in order to solve fuzzy PDEs on the basis of the Siekkala derivative. Furthermore the stability of the methodology is laid down.

2.4 Conclusions

In this chapter, some of numerical methodologies are demonstrated as a solution of fuzzy equations, dual fuzzy equations, fuzzy PDEs as well as FDEs. This review illustrates that the real roots associated with fuzzy equation can be extracted with different algorithms. Whatsoever in few cases there exist no real roots in certain fuzzy equation. Solution of fuzzy polynomial by ranking methodology is proposed for solving fuzzy polynomial equation which convert to a crisp system of polynomial equations, therefore the system is easily solvable. For obtaining the real roots of system in a case that there is no exact solution, iteration methodologies can be utilized for estimating the solution. By modified Adomian decomposition methodology, the real roots can be extracted by laying down fuzzy polynomial in parametric form and solving it by Adomian decomposition methodology. Obtaining solution by Newton methodology demonstrates that the real roots of fuzzy equation can be extracted on initial step with high preciseness. Differently with fuzzy neural network, the real root of fuzzy equation can be obtained by laying down a learning algorithm. This review supplies an input for those showing interest in the field of fuzzy equations.

Chapter 3

Fuzzy control and modeling with fuzzy equations

In this chapter, we discuss more general fuzzy equations termed as dual fuzzy equations [213]. Normal fuzzy equations have fuzzy numbers only on one side of the equation. However, dual fuzzy equations have fuzzy numbers on both sides of the equation. Since the fuzzy numbers cannot be moved between the sides of the equation [119], dual fuzzy equations are more general and difficult. We first discuss the existence of the solutions of the dual fuzzy equations. It corresponds to controllability problem of the fuzzy control [65]. Then we provide two methods to approximate the solutions of the dual fuzzy equations. They are controller design process.

For modeling, we first transform the fuzzy equation into a neural network. Then we modify the normal gradient descent method to train the fuzzy coefficients. With this modification, we can apply normal neural modeling methods to uncertainty nonlinear system modeling with fuzzy equations. The approximation theory for crisp models are extended into fuzzy cases. The upper bounds of the modeling errors with fuzzy equations are estimated.

Some simulation results are provided to show performance and effectiveness of our fuzzy control and modeling design methods with neural networks.

3.1 Nonlinear system modeling with fuzzy equations

Consider the following unknown discrete-time nonlinear system

$$\bar{x}_{k+1} = \bar{f}\left[\bar{x}_k, u_k\right], \quad y_k = \bar{g}\left[\bar{x}_k\right] \tag{3.1}$$

where $u_k \in \mathbb{R}^u$ is the input vector, $\bar{x}_k \in \mathbb{R}^l$ is an internal state vector and $y_k \in \mathbb{R}^m$ is the output vector. $\bar{f}$ and $\bar{g}$ are general nonlinear smooth functions $\bar{f}, \bar{g} \in C^\infty$. Denote $Y_k = \left[y_{k+1}^T, y_k^T, \cdots\right]^T$ and $U_k = \left[u_{k+1}^T, u_k^T, \cdots\right]^T$. If $\frac{\partial Y}{\partial \bar{x}}$ is non-singular at $\bar{x} = 0$ and $U = 0$, then this leads to the following model

$$y_k = \Phi[y_{k-1}^T, y_{k-2}^T, \cdots u_k^T, u_{k-1}^T, \cdots] \tag{3.2}$$

where $\Phi(\cdot)$ is an unknown nonlinear difference equation representing the plant dynamics, u_k and y_k are measurable scalar input and output. The nonlinear system (3.2) is a NARMA model. We can also regard the input of the nonlinear system as

$$x_k = [y_{k-1}^T, y_{k-2}^T, \cdots u_k^T, u_{k-1}^T, \cdots]^T$$

and the output as y_k.

Many nonlinear systems as in (3.2) can be rewritten as the following linear-in-parameter model,

$$y_k = \sum_{j=0}^{n} a_j x_k^j \tag{3.3}$$

or

$$y_k = \sum_{i=1}^{n} a_i f_i\left(x_k\right) \tag{3.4}$$

or

$$y_k + \sum_{i=1}^{m} b_i g_i(x_k) = \sum_{i=1}^{n} a_i f_i\left(x_k\right) \tag{3.5}$$

where a_j, a_i and b_i are linear parameters, x_k^j, $f_i\left(x_k\right)$ and $g_i(x_k)$ are nonlinear functions. The variables of these functions are measurable input and output.

A famous example of this kind of model is the robot manipulator [194]

$$M(q)\ddot{q} + C(q,\dot{q})\dot{q} + B\dot{q} + g(q) = \tau \tag{3.6}$$

(3.6) can be rewritten as

$$\sum_{i=1}^{n} Y_i(q,\dot{q},\ddot{q})\theta_i = \tau \tag{3.7}$$

To identify or control the linear-in-parameter system (3.3), (3.4), (3.5) or (3.7) the normal least square or adaptive methods can be applied directly.

Here we consider the uncertain nonlinear systems, i.e., the parameter a_j, a_i, b_i and θ_i are not fixed (not crisp). They are uncertain in the sense of fuzzy logic. The uncertain nonlinear systems are modeled by linear-in-parameter models with fuzzy parameters. These models are called fuzzy equations. The special case of fuzzy equations is fuzzy polynomial interpolation. Before introducing fuzzy equations and fuzzy polynomial interpolation, we need the following definitions. The explanations for these definitions can be found in [220].

Definition 3.1 (fuzzy number) *A fuzzy number u is a function $u \in E : \Re \to [0,1]$, such that, 1) u is normal, (there exists $x_0 \in \Re$ such that $u(x_0) = 1$; 2) u is convex, $u(\lambda x + (1-\lambda)y) \geq \min\{u(x), u(y)\}$, $\forall x, y \in \Re, \forall \lambda \in [0,1]$; 3) u is upper semi-continuous on $\Re$, i.e., $u(x) \leq u(x_0) + \varepsilon$, $\forall x \in N(x_0)$, $\forall x_0 \in \Re,$, $\forall \varepsilon > 0$, $N(x_0)$ is a neighborhood; 4) The set $u^+ = \{x \in \Re, u(x) > 0\}$ is compact.*

We use membership functions to express the fuzzy number. The most popular membership functions are the triangular function

$$u(x) = F(a,b,c) = \begin{cases} \frac{x-a}{b-a} & a \leq x \leq b \\ \frac{c-x}{c-b} & b \leq x \leq c \end{cases} \tag{3.8}$$

otherwise $u(x) = 0$ and trapezoidal function

$$u(x) = F(a,b,c,d) = \begin{cases} \frac{x-a}{b-a} & a \leq x \leq b \\ \frac{d-x}{d-c} & c \leq x \leq d \\ 1 & b \leq x \leq c \end{cases} \tag{3.9}$$

otherwise $u(x) = 0$.

Similar with crisp number, the fuzzy number u has also four basic operations: $\oplus$, $\ominus$, $\odot$, and $\oslash$. They represent the operations: sum, subtract, multiply, and multiplied by a crisp number.

The dimension of x in the fuzzy number u depends on the membership function, for example (3.8) has three variables, (3.9) has four variables. In order to define consistency operations, we first apply $\alpha-$level operation to the fuzzy number.

Definition 3.2 (α-level) *The α-level of fuzzy number u is defined as*

$$[u]^\alpha = \{x \in \Re : u(x) \geq \alpha\} \tag{3.10}$$

where $0 < \alpha \leq 1$, $u \in E$.

So $[u]^0 = u^+ =\{x \in \Re, u(x) > 0\}$. Because $\alpha \in [0,1]$, $[u]^\alpha$ is bounded as $\underline{u}^\alpha \leq [u]^\alpha \leq \overline{u}^\alpha$. The α-level of u between $\underline{u}^\alpha$ and $\overline{u}^\alpha$ is defined as

$$[u]^\alpha = A\left(\underline{u}^\alpha, \overline{u}^\alpha\right) \tag{3.11}$$

Let $u, v \in E$, $\lambda \in \Re$, we define the following fuzzy operations. $\underline{u}^\alpha$ and $\overline{u}^\alpha$ are the function of α. We define $\underline{u}^\alpha = d_M(\alpha)$, $\overline{u}^\alpha = d_U(\alpha)$, $\alpha \in [0,1]$.

Definition 3.3 (Lipchitz constant) *[164] The Lipschitz constant H of a fuzzy number $u \in E$ is*

$$\begin{array}{c} |d_M(\alpha_1) - d_M(\alpha_2)| \leq H\,|\alpha_1 - \alpha_2| \\ \text{or } |d_U(\alpha_1) - d_U(\alpha_2)| \leq H\,|\alpha_1 - \alpha_2| \end{array} \tag{3.12}$$

Definition 3.4 (fuzzy operations) *[210] Sum,*

$$[u \oplus v]^\alpha = [u]^\alpha + [v]^\alpha = [\underline{u}^\alpha + \underline{v}^\alpha, \overline{u}^\alpha + \overline{v}^\alpha] \tag{3.13}$$

Subtract,

$$[u \ominus v]^\alpha = [u]^\alpha - [v]^\alpha = [\underline{u}^\alpha - \underline{v}^\alpha, \overline{u}^\alpha - \overline{v}^\alpha] \tag{3.14}$$

Multiply,

$$\underline{w}^{\alpha} \leq [u \odot v]^{\alpha} \leq \overline{w}^{\alpha} \text{ or } [u \odot v]^{\alpha} = A\left(\underline{w}^{\alpha}, \overline{w}^{\alpha}\right) \tag{3.15}$$

where $\underline{w}^{\alpha} = \underline{u}^{\alpha}\underline{v}^{1} + \underline{u}^{1}\underline{v}^{\alpha} - \underline{u}^{1}\underline{v}^{1}$, $\overline{w}^{\alpha} = \overline{u}^{\alpha}\overline{v}^{1} + \overline{u}^{1}\overline{v}^{\alpha} - \overline{u}^{1}\overline{v}^{1}$, $\alpha \in [0,1]$. *It is a cross product of twp fuzzy numbers.*

Multiplied by a crisp number: For arbitrary crisp real positive number τ,

$$\begin{aligned} &-\tau\underline{u}^{\alpha} \leq [u]^{\alpha} \oslash \tau \leq -\tau\overline{u}^{\alpha} \\ &\text{or } [u]^{\alpha} \oslash \tau = A\left(-\tau\underline{u}^{\alpha}, -\tau\overline{u}^{\alpha}\right) \end{aligned}$$

Obviously, we have the following properties: the scalar multiplication: $\alpha \in [0,1]$

$$[\lambda u]^{\alpha} = \lambda[u]^{\alpha} = \begin{cases} A\left(\lambda\underline{u}^{\alpha}, \lambda\overline{u}^{\alpha}\right) & \lambda \geq 0 \\ A\left(\lambda\overline{u}^{\alpha}, \lambda\underline{u}^{\alpha}\right) & \lambda < 0 \end{cases} \tag{3.16}$$

$$\ominus u = (-1)u, \quad u \in E$$

Definition 3.5 (dot product) *[37]The dot product of two fuzzy variables u and v is*

$$(u.v)^{\alpha} = A\begin{pmatrix} \min\{\underline{u}^{\alpha}\underline{v}^{\alpha}, \underline{u}^{\alpha}\overline{v}^{\alpha}, \overline{u}^{\alpha}\underline{v}^{\alpha}, \overline{u}^{\alpha}\overline{v}^{\alpha}\} \\ \max\{\underline{u}^{\alpha}\underline{v}^{\alpha}, \underline{u}^{\alpha}\overline{v}^{\alpha}, \overline{u}^{\alpha}\underline{v}^{\alpha}, \overline{u}^{\alpha}\overline{v}^{\alpha}\} \end{pmatrix}$$

Definition 3.6 (distance) *The distance between the fuzzy numbers u and v is*

$$d(u,v) = \sup_{0 \leq \alpha \leq 1}\left\{\max\left(|\underline{u}^{\alpha} - \underline{v}^{\alpha}|, |\overline{u}^{\alpha} - \overline{v}^{\alpha}|\right)\right\} \tag{3.17}$$

Definition 3.7 (absolute value) *[9]Absolute value of a triangular fuzzy number* $u(x) = F(a,b,c)$ *is*

$$|u(x)| = |a| + |b| + |c| \tag{3.18}$$

Definition 3.8 (positive) *A fuzzy number* $u \in E$ *is said to be positive if* $\underline{u}^{1} \geq 0$ *and negative if* $\overline{u}^{1} \leq 0$.

Clearly, If u is positive and v is negative then $u \odot v = \ominus(u \odot (\ominus v))$ is a negative fuzzy number. If u is negative and v is positive then $u \odot v = \ominus((\ominus u) \odot v)$ is a negative fuzzy number. If u and v are negative then $u \odot v = (\ominus u) \odot (\ominus v)$ is a positive fuzzy number.

If u is positive and v is negative:

$$(u \odot v)^\alpha = A \left(\begin{array}{c} \overline{u}^\alpha \underline{v}^1 + \overline{u}^1 \underline{v}^\alpha \\ -\overline{u}^1 \underline{v}^1, \underline{u}^\alpha \overline{v}^1 + \underline{u}^1 \overline{v}^\alpha - \underline{u}^1 \overline{v}^1 \end{array} \right)$$

If u is negative and v is positive:

$$(u \odot v)^\alpha = A \left(\begin{array}{c} \underline{u}^\alpha \overline{v}^1 + \underline{u}^1 \overline{v}^\alpha \\ -\underline{u}^1 \overline{v}^1, \overline{u}^\alpha \underline{v}^1 + \overline{u}^1 \underline{v}^\alpha - \overline{u}^1 \underline{v}^1 \end{array} \right)$$

If u and v are negative:

$$(u \odot v)^\alpha = A \left(\begin{array}{c} \overline{u}^\alpha \overline{v}^1 + \overline{u}^1 \overline{v}^\alpha \\ -\overline{u}^1 \overline{v}^1, \underline{u}^\alpha \underline{v}^1 + \underline{u}^1 \underline{v}^\alpha - \underline{u}^1 \underline{v}^1 \end{array} \right)$$

When the parameters in the linear-in-parameter model (3.3), (3.4) or (3.5) are fuzzy number, (3.3), (3.4) and (3.5) become fuzzy equations. For the uncertain nonlinear system (3.1), we use the following two types of fuzzy equations to model it

$$y_k = a_1 f_1(x_k) \oplus a_2 f_2(x_k) \oplus ... \oplus a_n f_n(x_k) \tag{3.19}$$

or

$$a_1 f_1(x_k) \oplus a_2 f_2(x_k) \oplus ... \oplus a_n f_n(x_k) = b_1 g_1(x_k) \oplus b_2 g_2(x_k) \oplus ... \oplus b_m g_m(x_k) \oplus y_k \tag{3.20}$$

Because a_i and b_i are fuzzy numbers, we use the fuzzy operation $\oplus$. (3.20) has more general form than (3.19), it is called dual fuzzy equation.

In a special case, $f_i(x_k)$ has polynomial form,

$$y_k = a_1 x_k \oplus ... \oplus a_n x_k^n \tag{3.21}$$

or

$$a_1 x_k \oplus ... \oplus a_n x_k^n = b_1 x_k \oplus ... \oplus b_n x_k^n \oplus y_k \tag{3.22}$$

(3.21) is called fuzzy polynomial and (3.22) is called dual fuzzy polynomial.

Modeling with fuzzy equation (or fuzzy polynomial) can be regarded as fuzzy interpolation. We use the the polynomial fuzzy equation (3.21) or dual polynomial fuzzy equation (3.22) to model a nonlinear function

$$z_k = f(x_k) \tag{3.23}$$

The object is to minimize error between the two output y_k and z_k. Since y_k is a fuzzy number and z_k is a crisp number, we use the maximum of all points as the modeling error

$$\max_k |y_k - z_k| = \max_k |y_k - f(x_k)| = \max_k |\beta_k| \tag{3.24}$$

where $y_k = F\left(a\left(k\right), b\left(k\right), c\left(k\right)\right)$, $\beta_k = F\left(\beta_1, \beta_2, \beta_3\right)$, which are defined in (3.8). From the definition of the absolute value of a triangular fuzzy number (3.18),

$$\begin{array}{l} \max_k |\beta_k| = \max_k \left[\begin{array}{c} |a\left(k\right) - f(x_k)| \\ +|b\left(k\right) - f(x_k)| + |c\left(k\right) - f(x_k)| \end{array} \right] \\ \beta_1 = \max_k |a\left(k\right) - f(x_k)| \\ \beta_2 = \max_k \left\{b\left(k\right) + f(x_k)\right\} \\ \beta_3 = \max_k \left\{c\left(k\right) + f(x_k)\right\} \end{array} \tag{3.25}$$

The modelling problem (3.24) is to find $a\left(k\right)$, $b\left(k\right)$, and $c\left(k\right)$, such that

$$\min_{a_k, b_k, c_k} \left\{ \max_k |\beta_k| \right\} = \min_{a_k, b_k, c_k} \left\{ \max_k |y_k - f(x_k)| \right\} \tag{3.26}$$

From (3.25)

$$\beta_1 \geq |a\left(k\right) - f(x_k)|, \quad \beta_2 \geq b\left(k\right) + f(x_k), \quad \beta_3 \geq c\left(k\right) + f(x_k)$$

(3.26) can be solved by the linear programming method regarded to fuzzy polynomial,

$$\left\{ \begin{array}{ll} & \min \beta_1 \\ \text{subject:} & \begin{array}{l} \beta_1 + \sum_{j=0}^{n} a_j x_k^j \geq f(x_k) \\ \beta_1 - \sum_{j=0}^{n} a_j x_k^j \geq -f(x_k) \end{array} \end{array} \right. \tag{3.27}$$

$$\begin{cases} \min \beta_2 \\ \text{subject:} \quad \begin{array}{l} \beta_2 - \sum_{j=0}^{n} \underline{a}_j x_k^j \geq f(x_k) \\ \beta_2 \geq 0 \end{array} \end{cases} \tag{3.28}$$

$$\begin{cases} \min \beta_3 \\ \text{subject:} \quad \begin{array}{l} \beta_3 - \sum_{j=0}^{n} \bar{a}_j x_k^j \geq f(x_k) \\ \beta_3 \geq 0 \end{array} \end{cases} \tag{3.29}$$

Also, (3.26) can be solved by the linear programming method regarded to dual fuzzy polynomial,

$$\begin{cases} \min \beta_1 \\ \text{subject:} \quad \begin{array}{l} \beta_1 + \sum_{j=0}^{n} a_j x_k^j \ominus \sum_{j=0}^{n} b_j x_k^j \geq f(x_k) \\ \beta_1 - \{\sum_{j=0}^{n} a_j x_k^j \ominus \sum_{j=0}^{n} b_j x_k^j\} \geq -f(x_k) \end{array} \end{cases} \tag{3.30}$$

$$\begin{cases} \min \beta_2 \\ \text{subject:} \quad \begin{array}{l} \beta_2 - \left[\sum_{j=0}^{n} \underline{a}_j x_k^j \ominus \sum_{j=0}^{n} \underline{b}_j x_k^j\right] \geq f(x_k) \\ \beta_2 \geq 0 \end{array} \end{cases} \tag{3.31}$$

$$\begin{cases} \min \beta_3 \\ \text{subject:} \quad \begin{array}{l} \beta_3 - \left[\sum_{j=0}^{n} \bar{a}_j x_k^j \ominus \sum_{j=0}^{n} \bar{b}_j x_k^j\right] \geq f(x_k) \\ \beta_3 \geq 0 \end{array} \end{cases} \tag{3.32}$$

where $\underline{a}_j$, $\underline{b}_j$, $\bar{a}_j$ and $\bar{b}_j$ are defined as in (3.11). In this way, the best approximation of $f(x_k)$ at point x_k is $y_k = F(a_k, b_k, c_k)$. The approximation error of β_k is minimized.

In this chapter, we use fuzzy equation (3.19) and dual fuzzy equation (3.20) to model the uncertain nonlinear system (3.1), such that the output of the plant y_k can follow desired output y_k^*,

$$\min_{a_k} \|y_k - y_k^*\| \tag{3.33}$$

This modeling object can be considered as: finding a_k for the following fuzzy equation

$$y_k^* = a_1 f_1(x_k) \oplus a_2 f_2(x_k) \oplus ... \oplus a_n f_n(x_k) \tag{3.34}$$

where $x_k = [y_{k-1}^T, y_{k-2}^T, \cdots u_k^T, u_{k-1}^T, \cdots]^T$.

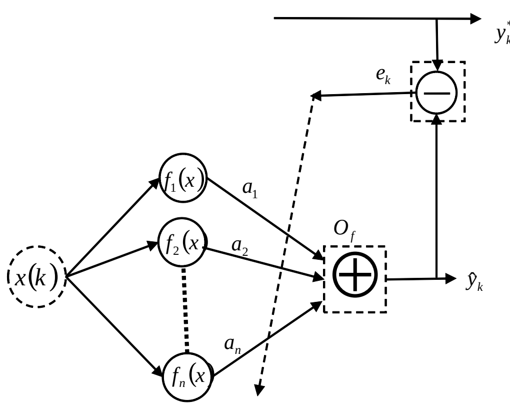

Figure 3.1: Fuzzy equation in the form of a neural network

The controller design process is to find u_k, such that the output of the plant y_k can follow desired output y_k^*, or the trajectory tracking error is minimized

$$\min_{u_k} \|y_k - y_k^*\| \tag{3.35}$$

This control object can be considered as: finding a solution u_k for the following dual fuzzy equation

$$a_1 f_1(x_k) \oplus a_2 f_2(x_k) \oplus ... \oplus a_n f_n(x_k) = b_1 g_1(x_k) \oplus b_2 g_2(x_k) \oplus ... \oplus b_m g_m(x_k) \oplus y_k^* \tag{3.36}$$

3.1.1 Fuzzy parameter estimation with neural networks

We design a neural network to represent the fuzzy equation (3.19), see Figure 3.1. The input to the neural network is $x(k)$, the output is the fuzzy number $\hat{y}_k$. The weights are a_i. The objective is to find suitable weight a_i such that the output of the neural network $\hat{y}_k$ converges to the desired output y_k^*.

In order to simplify the operation of the neural network, we use the triangular fuzzy number (3.8). The input fuzzy number x_k is first applied to α-level as in (3.10)

$$[x_k]^\alpha = A\left(\underline{x_k}^\alpha, \overline{x_k}^\alpha\right) \quad k = 0 \cdots N \tag{3.37}$$

Then we have

$$[O_j]^\alpha = A(f_j(\underline{x_k})^\alpha, f_j\overline{(x_k)}^a \quad j = 1 \cdots n \tag{3.38}$$

The output of the neural network is

$$\begin{array}{c}[\hat{y}_k]^\alpha = A\{\sum_{j\epsilon M} \underline{O_j}^\alpha \underline{a_j}^\alpha + \sum_{j\epsilon C} \overline{O_j}^\alpha \underline{a_j}^\alpha \\ +\underline{a_0}^\alpha, \sum_{j\epsilon M\,'} \overline{O_j}^\alpha \overline{a_j}^\alpha + \sum_{j\epsilon C\,'} \underline{O_j}^\alpha \overline{a_j}^\alpha + \overline{a_0}^\alpha\}\end{array} \tag{3.39}$$

where $M = \{j|\ \underline{a_j}^\alpha \geq 0\}$, $C = \{j|\ \underline{a_j}^\alpha < 0\}$, $M\,' = \{j|\ \overline{a_j}^\alpha \geq 0\}$, $C\,' = \{j|\ \overline{a_j}^\alpha < 0\}$.

In order to train the weights, we need to define a cost function for the fuzzy numbers. The training error is

$$e_k = y_k^* - \hat{y}_k$$

where $[y_k^*]^\alpha = A\left(\underline{y_k^*}^\alpha, \overline{y_k^*}^\alpha\right)$, $[\hat{y}_k]^\alpha = A\left(\underline{\hat{y}_k}^\alpha, \overline{\hat{y}_k}^\alpha\right)$, $[e_k]^\alpha = A\left(\underline{e_k}^\alpha, \overline{e_k}^\alpha\right)$. The cost function is defined as

$$\begin{array}{c}J_k = \underline{J}^\alpha + \overline{J}^\alpha \\ \underline{J}^\alpha = \frac{1}{2}\left(\underline{y_k^*}^\alpha - \underline{\hat{y}_k}^\alpha\right)^2 \\ \overline{J}^\alpha = \frac{1}{2}\left(\overline{y_k^*}^\alpha - \overline{\hat{y}_k}^\alpha\right)^2\end{array} \tag{3.40}$$

Obviously, $J_k \to 0$ means $[\hat{y}_k]^\alpha \to [y_k^*]^\alpha$.

Now we use gradient method to train the weight $a_j = \left[a_j^1, a_j^2, a_j^3\right]$ defined in (3.8), or a_j^r, $r = 1, 2, 3$. We calculate $\frac{\partial J_k}{\partial a_j^r}$ as

$$\frac{\partial J_k}{\partial a_j^r} = \frac{\partial \underline{J}^\alpha}{\partial a_j^r} + \frac{\partial \overline{J}^\alpha}{\partial a_j^r}$$

By the chain rule

$$\frac{\partial \underline{J}^\alpha}{\partial a_j^r} = \frac{\partial \underline{J}^\alpha}{\partial \underline{\hat{y}_k}^\alpha} \frac{\partial \underline{\hat{y}_k}^\alpha}{\partial \underline{a_j}^\alpha} \frac{\partial \underline{a_j}^\alpha}{\partial a_j^r} = \left(\underline{y_k^*}^\alpha - \underline{\hat{y}_k}^\alpha\right)\Gamma$$

where $j = 0$, and

$$\begin{array}{l}\frac{\partial \underline{J}^\alpha}{\partial a_j^r} = \frac{\partial \underline{J}^\alpha}{\partial \underline{\hat{y}_k}^\alpha} \frac{\partial \underline{\hat{y}_k}^\alpha}{\partial \underline{a_j}^\alpha} \frac{\partial \underline{a_j}^\alpha}{\partial a_j^r} \\ = \left(\underline{y_k^*}^\alpha - \underline{\hat{y}_k}^\alpha\right)\left\{\begin{array}{l}\underline{O_j}^\alpha\Gamma,\ \underline{a_j}^\alpha \geq 0 \\ \overline{O_j}^\alpha\Gamma,\ \underline{a_j}^\alpha < 0\end{array}\right.\end{array}$$

where $j = 1, ..., n$ and $\Gamma = \begin{cases} 1-\alpha & r=1 \\ \alpha & r=2 \\ 0 & r=3 \end{cases}$, also we have

$$\frac{\partial \overline{J}^{\alpha}}{\partial a_j^r} = \frac{\partial \overline{J}^{\alpha}}{\partial \overline{\hat{y}_k}^{\alpha}} \frac{\partial \overline{\hat{y}_k}^{\alpha}}{\partial \overline{a_j}^{\alpha}} \frac{\partial \overline{a_j}^{\alpha}}{\partial a_j^r} = \left(\overline{y_k^*}^{\alpha} - \overline{\hat{y}_k}^{\alpha} \right) \Gamma_1$$

where $j = 0$, and

$$\begin{array}{l} \frac{\partial \overline{J}^{\alpha}}{\partial a_j^r} = \frac{\partial \overline{J}^{\alpha}}{\partial \overline{\hat{y}_k}^{\alpha}} \frac{\partial \overline{\hat{y}_k}^{\alpha}}{\partial \overline{a_j}^{\alpha}} \frac{\partial \overline{a_j}^{\alpha}}{\partial a_j^r} \\ = \left(\overline{y_k^*}^{\alpha} - \overline{\hat{y}_k}^{\alpha} \right) \begin{cases} \overline{O_j}^{\alpha} \Gamma_1, & \overline{a_j}^{\alpha} \geq 0 \\ \underline{O_j}^{\alpha} \Gamma_1, & \overline{a_j}^{\alpha} < 0 \end{cases} \end{array}$$

where $j = 1, ..., n$ and $\Gamma_1 = \begin{cases} 0 & r=1 \\ \alpha & r=2 \\ 1-\alpha & r=3 \end{cases}$.

The coefficient a_j is updated as

$$a_j^r(k+1) = a_j^r(k) - \eta \frac{\partial J_k}{\partial a_j^r} \tag{3.41}$$

where $r = 1, 2, 3$, η is the training rate $\eta > 0$. In order to increase training process, we add a momentum term as

$$a_j^r(k+1) = a_j^r(k) - \eta \frac{\partial J_k}{\partial a_j^r} + \gamma \left[a_j^r(k) - a_j^r(k-1) \right] \tag{3.42}$$

where $\gamma > 0$.

3.1.2 Upper bounds of the modeling errors

In this part, we extend some well known approximation theories into fuzzy equation modeling. We first define the modelling error in the sense of fuzzy number.

Definition 3.9 *The distance between two fuzzy numbers, $u, v \in E$, is defined as the Hausdorff metric $d_H(u, v)$,*

$$d_H(u, v) = \max\{\sup_{x \in u} \inf_{y \in v} |x - y|, \sup_{y \in v} \inf_{x \in u} |x - y|\} \tag{3.43}$$

Lemma 3.1 *If $\theta \subset E$ is a compact set, then θ is uniformly support-bounded, i.e. there is a compact set $U \subset \Re$, such that $\forall u \in \theta$,*

$$Supp(u) \subset U.$$

Lemma 3.2 *Let $u, v \in E$, and $\alpha \in (0,1]$, $\lambda \in (0,+\infty]$, then we have: (i) if $f : \Re \to \Re$ is continuous, $[f(u)]^\alpha = f([u^\alpha])$ holds; (ii) if $f : \Re \to \Re$ is continuous, then $f(Supp(u)) = Supp(f(u))$.*

Proof. We need to only prove (ii) since (i) comes from [216]. At first, we demonstrate $\overline{f(A)} = f(\overline{A})$ for $A \subset \Re$. In fact, since $f(A) \subset f(\overline{A})$, and $f(\overline{A})$ is closed by the continuity of f, hence $\overline{f(A)} \subset f(\overline{A})$. On the other hand, for arbitrarily given $y \in f(\overline{A})$, there is a sequence $\{x_n | n \in N\} \subset \Re$, and a $x \in \Re$, such that $x_n \to x$ $(n \to +\infty)$, $y = f(x)$. The continuity of f implies $lim_{n\to+\infty} f(x_n) = f(x) = y$. But $f(x_n) \in f(A)$, so $y \in \overline{f(A)}$. Hence $f(\overline{A}) \subset \overline{f(A)}$. Thus $\overline{f(A)} = f(\overline{A})$.
Considering

$$\begin{array}{c} Supp(f(u)) = \overline{\{y \in \Re | f(u)(y) > 0\}} \\ f(Supp(u)) = f\left(\overline{\{x \in \Re | u(x) > 0\}}\right) \end{array}$$

we obtain the fact that

$$f(Supp(u)) = \overline{f(\{x \in \Re | u(x) > 0\})} = \overline{\{f(x) \in \Re | u(x) > 0\}}$$

holds. Since it may be easily proved that $\{y \in \Re | f(u)(y) > 0\} = \{f(x) | u(x) > 0\}$. Therefore,

$$Supp(f(u)) = f(Supp(u))$$

which implies the lemma. ■

Lemma 3.3 *Let $B \subset \Re$ be a compact set, and f, g be continuous on B, $h > 0$, moreover*

$$\forall x \in B, |f(x) - g(x)| < h \tag{3.44}$$

Then for each compact set $B_1 \subset B$, we have $|sup_{x\in B_1} f(x) - sup_{x \in B_1} g(x)| < h$.

Proof. Because of the facts that B_1 is a compact set and f, g are continuous on B_1, then there are $x_0 \in B_1$, $y_0 \in B_1$, such that

$$f(x_0) = \sup_{x \in B_1} f(x), \quad g(y_0) = \sup_{x \in B_1} g(x)$$

Supposing $|f(x_0) - g(y_{0)}| \geq h$, we have

$$f(x_0) - g(y_0) \leq -h, \quad or \quad f(x_0) - g(y_0) \geq h \tag{3.45}$$

In the first case (3.45), because $f(y_0) \leq f(x_0)$,

$$f(y_0) - g(y_0) \leq f(x_0) - g(y_0) \leq -h \Rightarrow |f(y_0) - g(y_0)| \geq h$$

holds, which contradicts (3.44). In the second case (3.45), since $g(x_0) \leq g(y_0)$, we obtain

$$f(x_0) - g(x_0) \geq f(x_0) - g(y_0) \geq h \Rightarrow |f(x_0) - g(x_0)| \geq h$$

which also contradicts (3.62). Therefore, (3.44) is not true, hence $-h < f(x_0) - g(y_0) < h$, so $|f(x_0) - g(x_0)| < h$, i.e. $|\sup_{x \in B_1} f(x) - \sup_{x \in B_1} g(x)| < h$. The proof is completed. ■

Theorem 3.1 *Let $f : \Re \rightarrow \Re$ be a continuous function, then for each compact set $\theta \subset E_0$ (the set of all the bounded fuzzy set), and $\psi > 0$, there are $n \in N$, and $a_0, a_i \in E_0$, $i = 1, 2, ..., n$, such that*

$$\forall x \in \theta \text{ and } \forall \tilde{x} \in \Re, \quad d(f(\tilde{x}), \sum_{i=1}^{n} f_i(x) a_i + a_0) < \psi \tag{3.46}$$

where ψ is a finite number.

Proof. The proof of Theorem can be followed from the below results. ■

If the function $f : \Re \rightarrow \Re$, we can extend f by the extension principle to the fuzzy function which is also written as $f : E_0 \rightarrow E$ as follows:

$$\forall u \in E_0, \quad f(u)(y) = \bigvee_{f(x)=y} \{u(x)\} \quad y \in \Re \tag{3.47}$$

f is called the extended function. Moreover, $cc(\Re)$ stands for the set of bounded closed intervals of $\Re$. Obviously

$$u \in E_0 \Longrightarrow \forall \alpha \in (0,1], \quad [u]^\alpha \in cc(\Re) \tag{3.48}$$

Moreover

$$Supp(u) \in cc(\Re) \tag{3.49}$$

So from now on, we suppose

$$Supp(u) = [s_1(u), s_2(u)] \tag{3.50}$$

Theorem 3.2 *Let $f : \Re \to \Re$ be a continuous function, then for each compact set $\theta \subset E_0$, $\varrho > 0$ and arbitrary $\varepsilon > 0$, there are $n \in N$, and $a_0, a_i \in E_0$, $i = 1, 2, ..., n$, such that*

$$\forall x \in \theta, \quad d(f(x), \sum_{i=1}^{n} f_i(x) a_i + a_0) < \varrho \tag{3.51}$$

where ϱ is a finite number. The lower and the upper limits of the α-level set of fuzzy function diminish to ϱ, but the center goes to ε.

Proof. Because $\theta \subset E_0$ is a compact set, hence by Lemma 3.4, we let $U \subset \Re$ be the compact set corresponding to θ. $\forall \varepsilon > 0$, by the conclusions in [69], there are $n \in N$, and $a_0, a_i \in \Re$, $i = 1, 2, ..., n$, such that

$$\forall x \in U, \quad |f(x) - \sum_{i=1}^{n} f_i(x) a_i + a_0| < \varepsilon \tag{3.52}$$

holds. Let $g(x) = \sum_{i=1}^{n} f_i(x) a_i + a_0, x \in \Re$, then

$$\forall x \in U, \quad |f(x) - g(x)| < \varepsilon \tag{3.53}$$

By Theorem 3.5, we imply (3.51) holds. ■

Theorem 3.3 *Supposing $\theta \subset E_0$ is compact, U the corresponding compact set of θ, and $f, g : \Re \to \Re$ are the continuous functions which satisfy the condition that for given $h > 0$,*

$$\forall x \in U, \quad |f(x) - g(x)| < h \tag{3.54}$$

holds. Then $\forall u \in \theta$, $d(f(u) - g(u)) \le h$.

Proof. Let $u \in E$ and $\alpha \in (0,1]$. Because f, g are continuous, hence $[f(u)]^\alpha = f([u^\alpha])$, $[g(u)]^\alpha = g([u^\alpha])$ holds by Lemma 3.5. Therefore, we obtain the following facts by the conclusions from [178],

$$d_H([f(u)]^\alpha - [g(u)]^\alpha) = d_H(f([u^\alpha]) - g([u^\alpha])) = \sup_{|p|=1}\{|s(p, f([u^\alpha])) - s(p, g([u^\alpha]))|\} \tag{3.55}$$

Because for $p \in \Re$: $|p| = 1$, we have

$$\begin{aligned} |s(p, f([u^\alpha]) - s(p, g([u^\alpha]))| &= |\sup\{py|y \in f([u^\alpha])\} - \sup\{py|y \in g([u^\alpha])\}| \\ &= |\sup\{pf(x)|x \in [u]^\alpha\} - \sup\{pg(x)|x \in [u]^\alpha\}| \end{aligned} \tag{3.56}$$

holds. And considering the conditions in the theorem, we obtain

$$\forall x \in [u]^\alpha, \quad |pf(x) - pg(x)| = |f(x) - g(x)| < h \tag{3.57}$$

Therefore, by (3.55), (3.56) and Lemma 3.6, the following

$$\forall \alpha \in (0,1], \quad d_H([f(u)]^\alpha, [g(u)]^\alpha) < h \Rightarrow d(f(u), g(u)) = \sup_{\alpha \in (0,1]}\{d_H([f(u)]^\alpha, [g(u)]^\alpha)\} \leq h \tag{3.58}$$

holds, which proves the theorem. ■

Arbitrary given $u \in E$, and $\alpha \in (0,1]$. Because f, g are continuous, hence $[f(u)]^\alpha = f([u^\alpha])$, $[g(u)]^\alpha = g([u^\alpha])$ holds by Lemma 3.5. Therefore, we obtain the following facts by the conclusions from [178],

$$d_H([f(u)]^\alpha - [g(u)]^\alpha) = d_H(f([u^\alpha]) - g([u^\alpha])) = \sup_{|p|=1}\{|s(p, f([u^\alpha])) - s(p, g([u^\alpha]))|\} \tag{3.59}$$

Because for $p \in \Re$: $|p| = 1$, we have

$$\begin{aligned} |s(p, f([u^\alpha]) - s(p, g([u^\alpha]))| &= |\sup\{py|y \in f([u^\alpha])\} - \sup\{py|y \in g([u^\alpha])\}| \\ &= |\sup\{pf(x)|x \in [u]^\alpha\} - \sup\{pg(x)|x \in [u]^\alpha\}| \end{aligned} \tag{3.60}$$

holds. And considering the conditions in the theorem, we obtain

$$\forall x \in [u]^\alpha, \quad |pf(x) - pg(x)| = |f(x) - g(x)| < h \tag{3.61}$$

Therefore, by (3.55), (3.56) and Lemma 3.6, the following

$$\forall\alpha \in (0,1], \quad d_H([f(u)]^\alpha, [g(u)]^\alpha) < h \Rightarrow d(f(u), g(u)) = \sup_{\alpha\in(0,1]} \{d_H([f(u)]^\alpha, [g(u)]^\alpha)\} \leq h \tag{3.62}$$

holds, which proves the theorem.

3.2 Fuzzy controller design

There are not analytical solution for the dual fuzzy equation (3.36). Here, we use neural networks to approximate the solution (control). In order to use neural networks to approximate the solution of the dual fuzzy equation (3.36), we first need to transform it into normal fuzzy equation as (3.19).

Generally, the inverse element for an arbitrary fuzzy number $u \in E$ does not exist , *i.e.*, there is not $v \in E$, such that

$$u \oplus v = 0$$

In other word,

$$u \oplus (\ominus u) \neq 0$$

So (3.36) cannot be

$$\begin{gathered} a_1 f_1(x_k) \oplus ... \oplus a_n f_n(x_k) \ominus b_1 g_1(x_k) \ominus ... \ominus b_m g_m(x_k) = y_k^* \\ [a_1 \ominus b_1]\, f_1(x_k) \oplus [a_2 \ominus b_2]\, f_2(x_k) \oplus ... = y_k^* \end{gathered}$$

Here we use the $\oslash$ operation. We add $\oplus b_i g_i(x)$, and apply $\oslash\tau$ on the both sides of (3.36)

$$\begin{aligned} & a_1 f_1(x_k) \oplus ... \oplus a_n f_n(x_k) \oplus \{[b_1 g_1(x) \oplus ... \oplus b_m g_m(x)] \oslash \tau\} \\ & = b_1 g_1(x_k) \oplus ... \oplus b_m g_m(x_k) \oplus \{[b_1 g_1(x) \oplus ... \oplus b_m g_m(x)] \oslash \tau\} \oplus y_k^* \end{aligned} \tag{3.63}$$

When $\tau = 1$, by the definition of $\oslash$, (3.63) is

$$a_1 f_1(x) \oplus ... \oplus a_n f_n(x) \ominus b_1 g_1(x) \ominus ... \ominus b_m g_m(x) = y_k^* \tag{3.64}$$

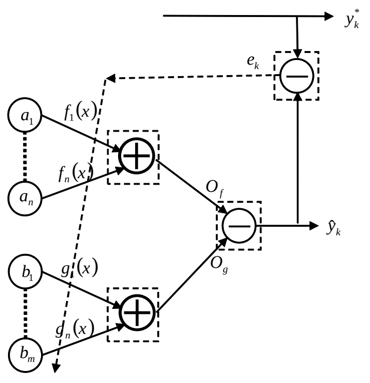

Figure 3.2: Dual fuzzy equation in the form of neural network (NN)

We design a neural network to represent the fuzzy equation (3.64), see Figure 3.2. The input to the neural network is the fuzzy numbers a_i and b_i, the output of the fuzzy number is y_k. The weights are $f_i(x)$ and $g_j(x)$.

The objective is to find suitable weight x (solution) such that the output of the neural network $\hat{y}_k$ converges to the desired output y_k^*. In the control point of view, we want to find a controller u_k which is a function of x, such that the output of the plant (3.1) y_k (crisp value) approximate the fuzzy number y_k^*.

In order to simplify the operation of the neural network as in Figure 3.2, we use the triangular fuzzy number (3.8). The input fuzzy numbers a_i and b_i are first applied to α-level as in (3.10)

$$\begin{array}{ll} [a_i]^\alpha = A\left(\underline{a}_i^\alpha, \overline{a}_i^\alpha\right) & i = 1 \cdots n \\ [b_j]^\alpha = A\left(\underline{b}_i^\alpha, \overline{b}_i^\alpha\right) & j = 1 \cdots m \end{array} \tag{3.65}$$

Then they are multiplied by the weights $f_i(x)$ and $g_j(x)$, and summarized according to (3.13)

$$\begin{array}{l} [O_f]^\alpha = A\left(\begin{array}{c} \sum_{i \epsilon M_f} f_i(x)\, \underline{a_i}^\alpha + \sum_{i \epsilon C_f} f_i(x)\, \overline{a_i}^\alpha, \\ \sum_{i \epsilon C_f} f_i(x)\, \overline{a_i}^\alpha, \sum_{i \epsilon M_f} f_i(x)\, \underline{a_i}^\alpha \end{array}\right) \\ [O_g]^\alpha = A\left(\begin{array}{c} \sum_{j \epsilon M_g} g_j(x)\, \underline{b_j}^\alpha + \sum_{j \epsilon C_g} g_j(x)\, \overline{b_j}^\alpha, \\ \sum_{j \epsilon C_g} g_j(x)\, \overline{b_j}^\alpha, \sum_{j \epsilon M_g} g_j(x)\, \underline{b_j}^\alpha \end{array}\right) \end{array} \tag{3.66}$$

where $M_f = \{i|f_i(x) \geq 0\}$, $C_f = \{i|f_i(x) < 0\}$, $M_g = \{j|g_j(x) \geq 0\}$, $C_g = \{j|g_j(x) < 0\}$.

The output of the neural network is

$$[\hat{y}_k]^\alpha = A\left(\underline{O_f}^\alpha - \underline{O_g}^\alpha, \overline{O_f}^\alpha - \overline{O_g}^\alpha\right) \tag{3.67}$$

In order to train the weights, we need to define a cost function for the fuzzy numbers. The training error is

$$e_k = y_k^* \ominus \hat{y}_k$$

where $[y_k^*]^\alpha = A\left(\underline{y_k^*}^\alpha, \overline{y_k^*}^\alpha\right)$, $[\hat{y}_k]^\alpha = A\left(\underline{\hat{y}_k}^\alpha, \overline{\hat{y}_k}^\alpha\right)$, $[e_k]^\alpha = A\left(\underline{e_k}^\alpha, \overline{e_k}^\alpha\right)$. The cost function is defined as

$$\begin{aligned} J_k &= \underline{J}^\alpha + \overline{J}^\alpha \\ \underline{J}^\alpha &= \tfrac{1}{2}\left(\underline{y_k^*}^\alpha - \underline{\hat{y}_k}^\alpha\right)^2 \\ \overline{J}^\alpha &= \tfrac{1}{2}\left(\overline{y_k^*}^\alpha - \overline{\hat{y}_k}^\alpha\right)^2 \end{aligned} \tag{3.68}$$

Obviously, $J_k \to 0$ means $[\hat{y}_k]^\alpha \to [y_k^*]^\alpha$.

Remark 3.1 *A main advantage of the least mean square index (3.68) is that it has a self-correcting feature which permits to operate for arbitrarily long period without deviating from its constraints. The corresponding gradient algorithm is susceptible to cumulative round off errors and is suitable for long runs without an additional error-correction procedure. It is more robust in statistics, identification and signal processing [199].*

Now we use gradient method to train the weights $f_i(x)$ and $g_j(x)$. The solution x_0 is the functions of $f_i(x)$ and $g_j(x)$. We calculate $\frac{\partial J_k}{\partial x_0}$ as

$$\frac{\partial J_k}{\partial x_0} = \frac{\partial \underline{J}^\alpha}{\partial x_0} + \frac{\partial \overline{J}^\alpha}{\partial x_0}$$

By the chain rule

$$\begin{aligned} \frac{\partial \underline{J}^\alpha}{\partial x_0} &= \frac{\partial \underline{J}^\alpha}{\partial \underline{\hat{y}_k}^\alpha} \frac{\partial \underline{\hat{y}_k}^\alpha}{\partial \underline{O_f}^\alpha} \sum \frac{\partial \underline{O_f}^\alpha}{\partial f_i(x)} \frac{\partial f_i(x)}{\partial x_0} + \frac{\partial \underline{e}^\alpha}{\partial \underline{\hat{y}_k}^\alpha} \frac{\partial \underline{\hat{y}_k}^\alpha}{\partial \underline{O_g}^\alpha} \sum \frac{\partial \underline{O_g}^\alpha}{\partial g_j(x)} \frac{\partial g_j(x)}{\partial x_0} \\ \frac{\partial \overline{J}^\alpha}{\partial x_0} &= \frac{\partial \underline{J}^\alpha}{\partial \underline{\overline{\hat{y}_k}}^\alpha} \frac{\partial \underline{\overline{\overline{\hat{y}_k}}}^\alpha}{\partial \overline{\overline{O_f}}^\alpha} \sum \frac{\partial \overline{O_f}^\alpha}{\partial f_i(x)} \frac{\partial f_i(x)}{\partial x_0} + \frac{\partial \overline{e_k}^\alpha}{\partial \overline{\hat{y}_k}^\alpha} \frac{\partial \overline{\hat{y}_k}^\alpha}{\partial \overline{O_f}^\alpha} \sum \frac{\partial \overline{O_f}^\alpha}{\partial g_j(x)} \frac{\partial g_j(x)}{\partial x_0} \end{aligned}$$

If f_i' and g_j' are positive

$$\begin{aligned}\frac{\partial \underline{J}^{\alpha}}{\partial x_0} &= \textstyle\sum_{i=1}^{n} -\left(\underline{y_k^{*}}^{\alpha} - \underline{\hat{y}_k}^{\alpha}\right) \underline{a}_i^{\alpha} f_i' + \sum_{j=1}^{m}\left(\underline{y_k^{*}}^{\alpha} - \underline{\hat{y}_k}^{\alpha}\right) \underline{b}_j^{\alpha} g_j' \\ \frac{\partial \overline{J}^{\alpha}}{\partial x_0} &= \textstyle\sum_{i=1}^{n} -\left(\overline{y_k^{*}}^{\alpha} - \overline{\hat{y}_k}^{\alpha}\right) \overline{a_i}^{\alpha} f_i' + \sum_{j=1}^{m}\left(\overline{y_k^{*}}^{\alpha} - \overline{\hat{y}_k}^{\alpha}\right) \overline{b_j}^{\alpha} g_j'\end{aligned}$$

Otherwise

$$\begin{aligned}\frac{\partial \underline{J}^{\alpha}}{\partial x_0} &= \textstyle\sum_{i=1}^{n} -\left(\underline{y_k^{*}}^{\alpha} - \underline{\hat{y}_k}^{\alpha}\right) \overline{a_i}^{\alpha} f_i' + \sum_{j=1}^{m}\left(\underline{y_k^{*}}^{\alpha} - \underline{\hat{y}_k}^{\alpha}\right) \overline{b_j}^{\alpha} g_j' \\ \frac{\partial \overline{J}^{\alpha}}{\partial x_0} &= \textstyle\sum_{i=1}^{n} -\left(\overline{y_k^{*}}^{\alpha} - \overline{\hat{y}_k}^{\alpha}\right) \underline{a}_i^{\alpha} f_i' + \sum_{j=1}^{m}\left(\overline{y_k^{*}}^{\alpha} - \overline{\hat{y}_k}^{\alpha}\right) \underline{b}_j^{\alpha} g_j'\end{aligned}$$

The solution x_0 is updated as

$$x_0\left(k+1\right) = x_0\left(k\right) - \eta \frac{\partial J_k}{\partial x_0}$$

where η is the training rate $\eta > 0$. In order to increase training process, we add a momentum term as

$$x_0\left(k+1\right) = x_0\left(k\right) - \eta \frac{\partial J_k}{\partial x_0} + \gamma\left[x_0\left(k\right) - x_0\left(k-1\right)\right]$$

where $\gamma > 0$.

After x_0 is updated, it should be substituted to the weights $f_i\left(x_0\right)$ and $g_j\left(x_0\right)$.

The solution of the dual fuzzy equation (3.36) can be also approximated by another type of neural network, see Figure 3.3. Here the inputs are the nonlinear functions $f_i\left(x\right)$ and $g_j\left(x\right)$, the weights are the fuzzy number a_i and b_j. We use the training error e_k to update x.

The input is a crisp number $x\left(k\right)$. The nonlinear operations $f_i\left(x\right)$ and $g_j\left(x\right)$, O_f and O_g are the same as (3.66). The output of this neural network is the same as (3.67).

The different between the networks of Figure 3.2 (NN) and Figure 3.3 (FNN) are: FNN does not change weights, it is an autonomous system. NN is a standard neural network. FNN is more robust than NN, and we can use bigger training rate η in FNN.

3.2.1 Controllability of uncertain nonlinear systems via dual fuzzy equations

Since the control object is to find a u_k for the dual fuzzy equation (3.36), the controllability problem occurs if the dual fuzzy equation has solution. In order to show the existence of the solution of (3.36), we need the following lemmas

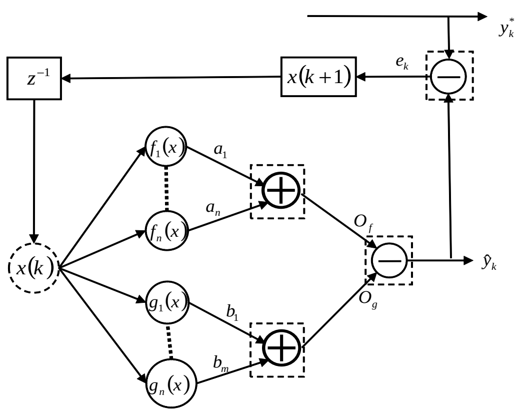

Figure 3.3: Dual fuzzy equation in the form of feedback neural network (FNN)

Lemma 3.4 *If the dual fuzzy equation (3.36) has a crisp solution u_k, then*

$$\left\{\cap_{j=1}^{n} domain\left[f_j\left(x\right)\right]\right\} \cap \left\{\cap_{j=1}^{m} domain\left[g_j\left(x\right)\right]\right\} \neq \phi \tag{3.69}$$

Proof. Let $u_0 \in \Re$ be a solution of (3.36), the dual fuzzy equation becomes

$$a_1 f_1(u_0) \oplus ... \oplus a_n f_n(u_0) = b_1 g_1(u_0) \oplus ... \oplus b_m g_m(u_0) \oplus y_k^*$$

Since $f_j(u_0)$ and $g_j(u_0)$ exist, $u_0 \in$domain$[f_j\left(x\right)]$, $u_0 \in$domain$[g_j\left(x\right)]$. Consequently, it can be concluded that $u_0 \in \cap_{j=1}^{n}$domain$[f_j\left(x\right)] = D_1$, and $u_0 \in \cap_{j=1}^{m}$domain$[g_j\left(x\right)] = D_2$. So there exists u_0, such that $u_0 \in D_1 \cap D_2 \neq \phi$. ■

Obviously, the necessary condition for the existence of the solution of (3.36) is (3.69).

Assume two fuzzy numbers m_0, $n_0 \in E$, $m_0 < n_0$. Define a set $K\left(x\right) = \{x \in E, m_0 \leq x \leq n_0\}$, and an operator $S : K \rightarrow K$, such that

$$S\left(m_0\right) \geq m_0, \quad S\left(n_0\right) \leq n_0 \tag{3.70}$$

here S is condensing and continuous, it is bounded as $S(z) < r(z)$, $z \subset K$ and $r(z) > 0$. $r(Z)$ can be regarded as the measure of z.

Lemma 3.5 *If we define $n_i = S(n_{i-1})$ and $m_i = S(m_{i-1})$, $i = 1, 2, ...$, the upper and lower bounds of S are $\bar{s}$ and $\underline{s}$, then*

$$\bar{s} = \lim_{i\to+\infty} n_i, \quad \underline{s} = \lim_{i\to+\infty} m_i, \tag{3.71}$$

and

$$m_0 \le m_1 \le ... \le m_n \le ... \le n_n \le ... \le n_1 \le n_0. \tag{3.72}$$

The proof of this lemma is directly, see [63].

If there exists a fixed point x_0 in K, the successive iterates $x_i = S(x_{i-1}), i = 1, 2, ...$ will converge to x_0, *i.e.*, the distance (3.17) $\lim_{i\to\infty} d(x_i, x_0) = 0$.

Theorem 3.4 *If the fuzzy numbers a_i and b_j $(i = 1 \cdots n,\ j = 1 \cdots m)$ in (3.36) satisfy the Lipschitz condition (3.12)*

$$\begin{array}{c} |d_M(a_i) - d_M(a_k)| \le H\,|a_i - a_k| \\ |d_U(a_i) - d_U(a_k)| \le H\,|a_i - a_k| \\ |d_M(b_i) - d_M(b_k)| \le H\,|b_i - b_k| \\ |d_U(b_i) - d_U(b_k)| \le H\,|b_i - b_k| \end{array} \tag{3.73}$$

where $k = 1 \cdots n$, d_M and d_U are defined in (3.12), the upper bounds of f_i and g_j are $|f_i| \le \bar{f}$, $|g_j| \le \bar{g}$, then the dual fuzzy equation (3.36) has a solution u which is in the following set

$$K_H = \left\{ \begin{array}{c} u \in E, |\overline{u}^{\alpha_1} - \underline{u}^{\alpha_2}| \\ \le \left(n\bar{f} + m\bar{g}\right) H\,|\alpha_1 - \alpha_2| \end{array} \right\} \tag{3.74}$$

Proof. Because the fuzzy numbers a_i and b_j in (3.36) are linear-in-parameter, from the definition (3.12) and the property (3.16)

$$d_M(\alpha) = a_{1M}(\alpha) f_1(x) \oplus ... \oplus a_{nM}(\alpha) f_n(x) \ominus b_{1M}(\alpha) g_1(x) \ominus ... \ominus b_{mM}(\alpha) g_m(x)$$

So

$$\begin{array}{l} |d_M(\alpha) - d_M(\varphi)| = |f_1(x)| \mid a_{1M}(\alpha) \ominus a_{1M}(\varphi) \mid \\ + \cdots + |f_n(x)| \, |a_{nM}(\alpha) \ominus a_{nM}(\varphi)| \\ + |g_1(x)| \, |b_{1M}(\alpha) \ominus b_{1M}(\varphi)| \\ + \cdots + |g_m(x)| \, |b_{mM}(\alpha) \ominus b_{mM}(\varphi)| \end{array} \tag{3.75}$$

By the Lipschitz condition (3.12), (3.75) is

$$|d_M(\alpha) - d_M(\varphi)| \leq \bar{f} H \textstyle\sum_{i=1}^n |\alpha - \varphi| + \bar{g} H \textstyle\sum_{i=1}^n |\alpha - \varphi| = \left(n\bar{f} + m\bar{g}\right) H \, |\alpha - \varphi|$$

Similarly, the upper bounds satisfy

$$|d_U(\alpha) - d_U(\varphi)| \leq \left(n\bar{f} + m\bar{g}\right) H \, |\alpha - \varphi|$$

Since the lower bound $|d_M(\alpha) - d_M(\varphi)| \geq 0$, by Lemma 3.2 the solution is in K_H which is defined in (3.74). ■

The following theorem uses linear the programming conditions (3.30)-(3.32) to show the controllability conditions of the dual polynomial fuzzy equation (3.22).

Lemma 3.6 *If the data number m and the order the polynomial n in (3.22) satisfy*

$$m \geq 2n + 1 \tag{3.76}$$

where $k = 1 \cdots m$, then the solutions of (3.31) and (3.32) are $\beta_2 = \beta_3 = 0$.

Proof. Because

$$\sum_{j=0}^{n} \underline{a}_j x_k^j \ominus \sum_{j=0}^{n} \underline{b}_j x_k^j \leq -f(x_k) \tag{3.77}$$

$i = 1, 2, ..., m$. We choose $2n + 1$ points for x_k, and the interpolating the dual polynomial

$$b\,(k) = \sum_{j=0}^{n} \underline{a}_j x_k^j \ominus \sum_{j=0}^{n} \underline{b}_j x_k^j \tag{3.78}$$

If $h = \max_k\{b\,(k) + f(x_k)\}$ and $h > 0$, then we can change the dual polynomial (3.22) into a new dual polynomial $b\,(k) - h$. This new dual polynomial satisfies (3.77). Because the feasible point of (3.31) $\beta_2 \geq 0$, it must be zero. Similar result can be obtained for (3.32). ■

Both $f(x_k)$ and x_k are crisp. If the data number is $k = 1 \cdots n$, there exists solution for the polynomial approximation [148]. Because $b\,(k)$ and $c\,(k)$, (3.30) has a solution.

Theorem 3.5 *If the data number is big enough as (3.76), and the dual polynomial fuzzy equation (3.22) satisfies*

$$D\left[h\left(x_{k1}, u_{k1}\right), h\left(x_{k2}, u_{k2}\right)\right] \leq lD\left[u_{k1}, u_{k2}\right] \tag{3.79}$$

where $0 < l < 1$, $h(\cdot)$ *represents a dual polynomial fuzzy equation,*

$$h\left(x_{k1}, u_{k1}\right) : a_1 x_{k1} \oplus \ldots \oplus a_n x_{k1}^n = b_1 x_{k1} \oplus \ldots \oplus b_n x_{k1}^n \oplus y_{k1} \tag{3.80}$$

$D[u, v]$ *is the Hausdorff distance [197],*

$$D[u, v] = \max\left\{\sup_{x\in u}\inf_{y\in v} d(x, y), \sup_{x\in v}\inf_{y\in u} d(x, y)\right\}$$

$d(x, y)$ *is the distance defined in (3.17), then (3.22) has a unique solution* u.

Proof. From Lemma 3.2 we know, there are solutions for (3.30)-(3.32), if there are many data which satisfy (3.76). Without loss of generality, we assume the solutions for (3.30)-(3.32) are at $x_k = 0$, which corresponds to u_0. (3.79) means $h(\cdot)$ in (3.80) is continuous. If we choose a $\delta > 0$ such that $D[y_k, u_0] \leq \delta$, then

$$D[h(x_k, u_0), u_0] \leq (1 - l)\delta$$

Here $h(0, u_0) = u_0$. Now we select x near 0, $x_k \in [0, c]$, $c > 0$, and define

$$\mathcal{C}_0 : \rho = \sup_{x_k\in[0,c]} D\left[y_{k_1}, y_{k_2}\right]$$

Let $\{y_{k_m}\}$ be a sequence in $\mathcal{C}_0$, for any $\varepsilon > 0$, we can find $N_0(\varepsilon)$ such that $\rho < \varepsilon$, $m, n \geq N_0$. So $y_{k_m} \longrightarrow y_k$ for $x_k \in [0, c]$. Furthermore

$$D[y_k, u_0] \leq D[y_k, y_{k_m}] + D[y_{k_m}, u_0] < \varepsilon + \delta \tag{3.81}$$

for all $x \in [0, c]$, $m \geq N_0(\varepsilon)$. Since $\varepsilon > 0$ is arbitrary small,

$$D[y_k, u_0] \leq \delta \tag{3.82}$$

for all $x \in [0, c]$. We now show that y_k is continuous at $x_0 = 0$. Given $\delta > 0$, there exists $\delta_1 > 0$ such that

$$D[y_k, u_0] \leq D[y_k, y_{k_m}] + D[y_{k_m}, u_0] \leq \varepsilon + \delta_1$$

for every $m \geq N_0(\varepsilon)$, by (3.82), whenever $|x - x_0| < \delta_1$, y_k is continuous at $x_0 = 0$. So (3.22) has a unique solution u_0. ■

The necessary condition for the controllability (existence of solution) of the dual fuzzy equation (3.36) is (3.69), the sufficient condition of the controllability is (3.73). For most of membership functions such as the triangular function (3.8) and the trapezoidal function (3.9), the Lipschitz condition (3.73) is satisfied. They are controllable.

3.3 Simulations

In this section, we use several real applications to show how to use the dual fuzzy equation to design fuzzy controller.

Example 3.1 (A chemistry process) *A chemical reaction is to use the poly ethylene (PE) and poly propylene (PP) to generate a desired substance (DS). If the cost of the material is defined as x, the cost PE is x and the cost of PP is x^2. The weights of PE and PP are uncertain, which satisfy the triangle function (3.8). We want to product two types DS. If we wish the cost between them are $F(3.5, 4, 5) = y^*$, what is the cost x ? The weights of PE are $F(2.5, 3, 3.25) = a_1$ and $F(0.75, 1, 1.25) = b_1$. The weights of PP are $F(1.75, 2, 2.5) = a_2$ and $(1.75, 2, 2.5) = b_2$. The above relation can be modeled by the following dual fuzzy equation*

$$\begin{aligned} &(2.5, 3, 3.25)x \oplus (1.75, 2, 2.5)x^2 \\ &= (0.75, 1, 1.25)x \oplus (1.75, 2, 2.5)x^2 \oplus (3.5, 4, 5) \end{aligned}$$

Here $f_1(x) = g_1(x) = x$, $f_2(x) = g_2(x) = x^2$. We use NN and FNN shown in Figure 3.2 and Figure 3.3 to approximate the solution x. The learning rates for them are the same $\eta = 0.02$. The results are shown in Table 1. The exact solution is $x_0 = 2$. The neural networks start from $x(0) = 4$. Both neural networks converge to the real solution. The error $|\hat{x} - x_0|$ between

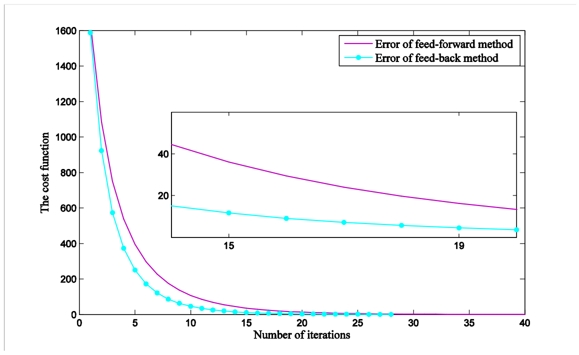

Figure 3.4: The error between the approximate solution and the exact solution

the approximate solution $\hat{x}$ and the exact solution x_0 is shown in Figure 3.4.

Table 1. Comparison results of two types of neural networks

k	$x(k)$ *with NN*	k	$x(k)$ *with FNN*
1	3.8377	1	3.7970
2	3.6105	2	3.3090
3	3.3435	3	2.9567
$\vdots$	$\vdots$	$\vdots$	$\vdots$
38	2.0053	26	2.0080
39	2.0044	27	2.0053
40	2.0036	28	2.0034

Example 3.2 (Heat source by insulating materials) *Heat source is in the center of the insulating materials. The thickness of the materials are not exact, which satisfy the trapezoidal function (3.9),*

$$A = F(0.12, 0.14, 0.15, 0.18) = a_1$$
$$B = F(0.08, 0.1, 0.2, 0.5) = a_2$$
$$C = F(0.09, 0.1, 0.2, 0.4) = b_1$$
$$D = F(0.02, 0.03, 0.05, 0.08) = b_2$$

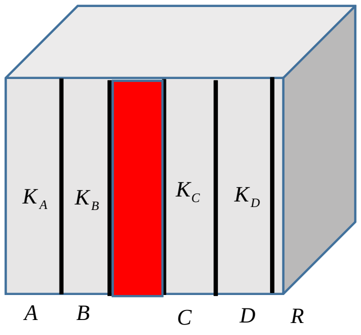

Figure 3.5: Heat source by insulating materials

see Figure 3.5. The conductivity coefficient of these materials are $K_A = e^x = f_1$, $K_B = x\sqrt{x} = f_2$, $K_C = x^2 = g_1$, $K_D = xsin(\frac{\Pi x}{8}) = g_2$, *here* x *is the elapsed time. The object of the example is to find the time when the thermal resistance at the right side arrives* $R = F(0.00415, 0.00428, 0.00569, 0.03187) = y^*$. *The thermal balance is [100]:*

$$\frac{A}{K_A} \oplus \frac{B}{K_B} = \frac{C}{K_C} \oplus \frac{D}{K_D} \oplus R$$

The exact solution is $x = 3$ *[100]. The maximum learning rate of NN as Figure 3.2 is* $\eta = 0.005$. *The maximum learning rate of FNN as Figure 3.3 is* $\eta = 0.1$. *The approximation results are shown in Table 2. FNN is faster and more robust than NN.*

Table 2. Comparison results of two types of neural networks

k	$x(k)$ *with NN*	k	$x(k)$ *with FNN*
1	0.6251	1	0.7250
2	1.0542	2	1.1060
3	1.3321	3	1.5042
⋮	⋮	⋮	⋮
39	2.9899	10	2.9931
40	2.9922	11	2.9959
41	2.9940	12	2.9974

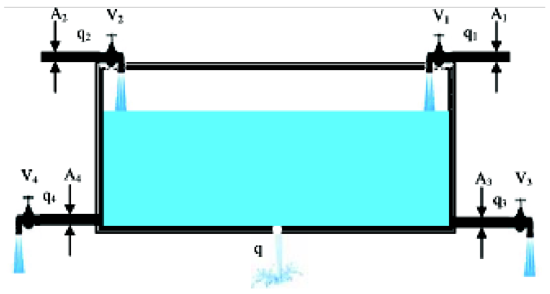

Figure 3.6: Water tank system

Example 3.3 (Water tank system) *The water tank system has two inlet valves q_1, q_2, and two outlet valves q_3, q_4, see Figure 3.6. The areas of the valves are uncertain as the triangle function (3.8), $A_1 = F(0.023, 0.025, 0.026)$, $A_2 = F(0.01, 0.02, 0.04)$, $A_3 = F(0.014, 0.015, 0.017)$, $A_4 = F(0.04, 0.06, 0.07)$. The velocities of the flow (controlled by the valves) are $f_1 = (\frac{x}{10})e^x$, $f_2 = xcos(\Pi x)$, $f_3 = cos(\frac{\Pi x}{8})$, $f_4 = \frac{x}{2}$. If we hope the outlet flow is $q = (4.090, 6.338, 36.402) = y^*$, what is the control variable x. The mass balance of the tank is [196]:*

$$\rho A_1 f_1 \oplus \rho A_2 f_2 = \rho A_3 f_3 \oplus \rho A_4 f_4 \oplus q$$

where ρ is the density of the water. The exact solution is $x_0 = 2$ [196]. We use $x(0) = 5$, $\eta = 0.001$, $\gamma = 0.001$ for both NN and FNN. The error $|\hat{x} - x_0|$ between the approximate solution $\hat{x}$ and the exact solution x_0 is shown in Figure 3.7. For this example, both NN and FNN work well.

Example 3.4 (Solid cylindrical rod) *The deformation of a solid cylindrical rod depends on the stiffness E, the forces on it F, the positions of the forces L, and the diameter of the rod d [201], see Figure 3.8. The positions are not exact, they satisfy the trapezoidal function (3.9). $L_1 = F(0.3, 0.4, 0.6, 0.7)$, $L_2 = F(0.5, 0.7, 0.8, 0.9)$, $L_3 = F(0.5, 0.7, 0.8, 0.9)$. The area of the rod is $A = \frac{\pi}{4}d^2$. The external forces are the function of x, $F_1 = x^7$, $F_2 = x^6\sqrt{x}$, $F_3 = e^{2x}$ [45]. We want the desired deformation at the point N be $N^* = F(0.000673, 0.000931, 0.001164, 0.001310)$ as in (3.9). what is the amount of control force,*

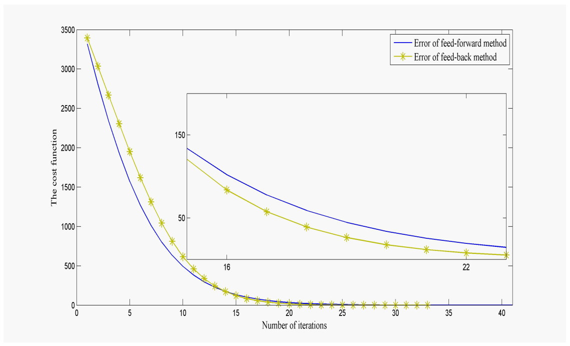

Figure 3.7: The error between the approximate solution and the exact solution

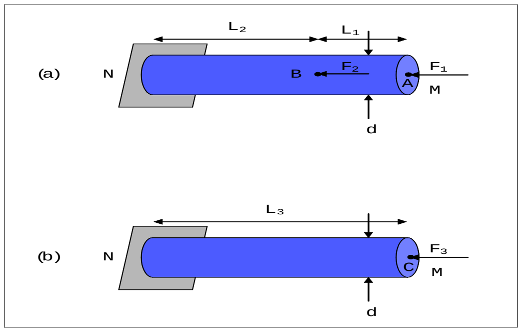

Figure 3.8: Two solid cylindrical rods

which should be applied? According to the tension relations [45]

$$\frac{L_1F_1}{AE} \oplus \frac{L_2(F_1+F_2)}{AE} = \frac{L_3F_3}{AE} \oplus N^*$$

where $d = 0.02$, $E = 70 \times 10^9$. *The exact solution is* $x = 4$.

We use $x(0) = 7$, $\eta = 0.002$, $\gamma = 0.002$ *for both NN and FNN. The error* $|\hat{x} - x_0|$ *between the approximate solution* $\hat{x}$ *and the exact solution* x_0 *is shown in Figure 3.9. For this example, both NN and FNN work well. FNN is little better than NN.*

Example 3.5 (Water Channel system) *The water in the pipe* d_1 *is divided into three pipes* d_2, d_3, d_4, *see Figure 3.10. The areas of the pipes are uncertain, they satisfy the*

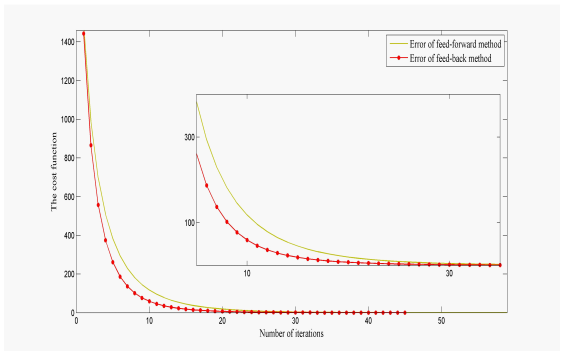

Figure 3.9: The error between the approximate solution and the exact solution

trapezoidal function (3.9). $A_1 = F(0.4, 0.6, 0.7, 0.8)$, $A_2 = F(0.05, 0.1, 0.2, 0.4)$, $A_3 = F(0.03, 0.08, 0.1, 0.2)$. *The water velocities in the pipes are controlled by the valves parameter* x, $v_1 = x^3$, $v_2 = \frac{e^x}{2}$, $v_3 = x$ *[196]. The control object is to let the flow in pipe* d_4, *which is*

$$Q = F(10.207861, 14.955723, 16.591446, 16.982892)$$

what is the valve control parameter x. *By mass balance*

$$A_1 v_1 = A_2 v_2 \oplus A_3 v_3 \oplus Q$$

The exact solution is $x = 3$ *[196]. The maximum learning rate of NN as Figure 3.2 is* $\eta = 0.001$. *The maximum learning rate of FNN as Figure 3.3 is* $\eta = 0.08$. *The approximation results are shown in Table 3. The error* $|\hat{x} - x_0|$ *between the approximate solution* $\hat{x}$ *and the*

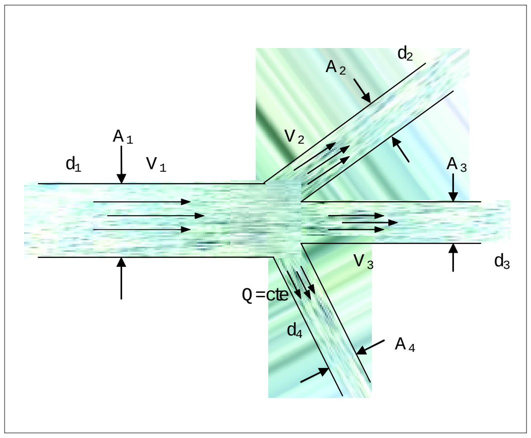

Figure 3.10: Water Channel system

exact solution x_0 is shown in Figure 3.11. FNN is faster and more robust than NN.

Table 3. Comparison results of two types of neural networks

k	$x(k)$ *with NN*	k	$x(k)$ *with FNN*
1	5.9024	1	5.9226
2	5.7361	2	5.5341
3	5.5321	3	5.1234
$\vdots$	$\vdots$	$\vdots$	$\vdots$
77	3.0599	21	3.0162
78	3.0322	22	3.0131
79	3.0110	23	3.0086

Example 3.6 (Water tanks system) *There are three tanks connected to a pipeline at a constant H, see Figure 3.12. We want to pump water from one tank to the other two tanks. This system satisfies the following relation*

$$H = A_0 \oplus A_1 Q \oplus A_2 Q^2 \oplus A_3 Q^3$$

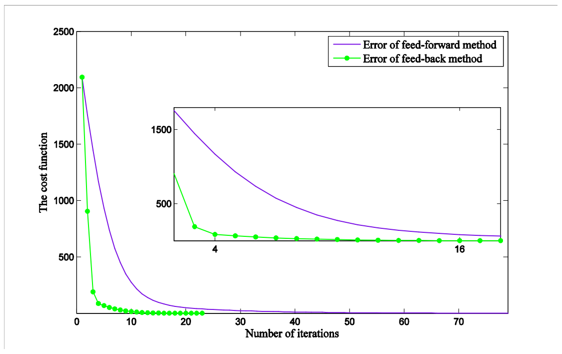

Figure 3.11: The error between the approximate solution and the exact solution

where Q is the quantity of flow, H is the height of the pipe, A_0, A_1, A_2, and A_3 are the characteristic coefficients of the the pump., they are

$$A_0 = (1, 5, 8), A_1 = (3, 7, 8), A_2 = (1, 2, 4), A_3 = (1, 3, 4)$$

We have the following 4 *real uncertain data*

$$Q = \{2, (2, 4, 5), (3, 5, 6, 7), (1, 2, 4)\}$$

where $(2, 4, 5)$ *and* $(1, 2, 4)$ *satisfy the triangle function (3.8),* $(3, 5, 6, 7)$ *is the trapezoidal function (3.9),* 2 *is a crisp number.*

$$H = \{(19, 51, 72), (19, 257, 648), (46, 465, 767, 1632), (6, 51, 360)\}$$

Now we use a neural network to approximate A_0, A_1, A_2, and A_3. The results are shown in

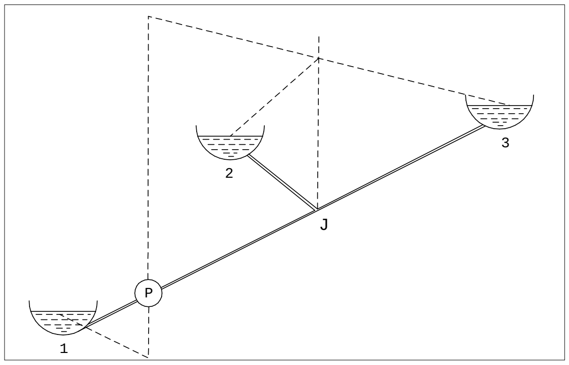

Figure 3.12: Pumping water from one tank to the other two tanks

Table 4.

Table 4. Neural network approximation for the coefficients

k	$\hat{A}_0$	$\hat{A}_1$
1	$(3.9, 7.9, 10.9)$	$(5.9, 9.8, 10.9)$
2	$(3.7, 7.7, 10.6)$	$(5.7, 9.6, 10.7)$
$\vdots$	$\vdots$	$\vdots$
75	$(1.0, 5.0, 8.0)$	$(3.0, 7.0, 8.0)$

k	$\hat{A}_2$	$\hat{A}_3$
1	$(2.91, 3.9, 5.9)$	$(3.9, 5.9, 6.9)$
2	$(2.7, 3.71, 5.8)$	$(3.7, 5.7, 6.8)$
$\vdots$	$\vdots$	$\vdots$
75	$(1.0, 2.0, 4.0)$	$(1.0, 3.0, 4.0)$

Example 3.7 (Heat source by insulating materials) *Heat source is in the left of the insulating materials, see Figure 3.13. The conductivity coefficient of these materials are* $K_A = e^x = f_1$, $K_B = x\sqrt{x} = f_2$, $K_C = x^2 = g_1$, $K_D = xsin(\frac{\Pi x}{8}) = g_2$, *here* x *is the elapsed time. The thermal balance is [100]:*

$$R = \frac{A}{K_A} \oplus \frac{B}{K_B} \oplus \frac{C}{K_C} \oplus \frac{D}{K_D}$$

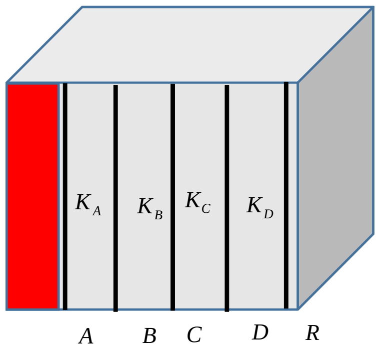

Figure 3.13: Heat source by insulating materials

Three types of variable x satisfy the triangle function (3.8) and the trapezoidal function (3.9)

$$x = \{(2, 3, 4), (3, 5, 6), (1, 3, 5, 7)\}$$

The corresponding data related to R are

$$R = \{(13, 43, 90), (24, 99, 180), (6, 43, 99, 237)\}$$

The parameters satisfy

$$A = (3, 4, 6), B = (1, 4, 5), C = (2, 3, 4)$$

The approximation results are shown in Table 5.

Table 5. Neural network approximation for the coefficients

t	A	B	C
1	$(5.8, 6.8, 8.8)$	$(3.9, 6.9, 7.92)$	$(3.9, 4.7, 5.9)$
2	$(5.7, 6.7, 8.6)$	$(3.8, 6.8, 7.8)$	$(3.8, 4.5, 5.8)$
3	$(5.6, 6.6, 8.5)$	$(3.6, 6.6, 7.7)$	$(3.6, 4.4, 5.6)$
$\vdots$	$\vdots$	$\vdots$	$\vdots$
95	$(3.0, 4.0, 6.0)$	$(1.0, 4.0, 5.0)$	$(2.0, 3.0, 4.0)$

3.4 Conclusions

In order to model uncertain nonlinear system, we use fuzzy equations and dual fuzzy equations, which are in the form of linear-in-parameter. The uncertainty is represented by fuzzy numbers. We first prove that these fuzzy models have solutions under certain conditions. These conditions are controllability of the fuzzy control algorithms. By some special fuzzy operations, we transform the dual fuzzy equations into two types of neural networks. We design modified gradient descent algorithms to train the neural networks, such that the solutions (fuzzy controllers) are estimated by the neural networks.

Since normal modeling methods cannot be applied for fuzzy number and fuzzy equation directly, we transform the fuzzy equation into a neural network. Then we modify the gradient descent method for fuzzy numbers updating, and propose a back-propagation learning rule for fuzzy equations. The upper bounds of the fuzzy modeling errors are proven. We successfully extend the approximation theory of crisp models to fuzzy equation model. The novel methods are validated with some benchmark examples.

Chapter 4

Modeling with fuzzy differential equation

In this chapter, a new model based on Bernstein neural network is used, which has good properties of Bernstein polynomial for FDE. The Bernstein polynomial has good uniform approximation ability for continuous functions [70]. Also it has innumerable drawing properties, homogeneous shape-sustaining approximation, bona fide estimation, and low boundary bias. A very important property of the Bernstein polynomial is that it generate a smooth estimation for equal distance knots [68]. This property is suitable for FDE and PDE approximation.

Two types of neural networks are used: static and dynamic models, to approximate the solutions of FDEs. These numerical methods use generalized differentiability of FDEs. The solutions of FDE is substituted into four ODEs. Then the corresponding Bernstein neural networks are applied.

For solving the strongly degenerate parabolic and Burgers-Fisher equations a model using neural networks is developed. A trial solution of this system is subdivided into two parts. The initial and boundary conditions compensate the first part that contains no adjustable parameters. The involvement of neural network containing adjustable parameters (weights and biases) concludes the second part. A training technique is implemented for the training of the network which requires the calculation of the error gradient in consideration of the network parameters. In order to maintain a continuation, a significantly modified methodology is illustrated for solving a wave equation which is modeled on the basis of two patterned Bernstein neural networks, static and dynamic models.

Some numerical examples are proposed to show the effectiveness of the approximation methods with the Bernstein neural networks.

4.1 Fuzzy modeling with fuzzy differential equation

In reference to fuzzy or the case concerned to interval arithmetic, equation $x = y \oplus z$ is not equivalent with the phase $z = x \ominus y = x \oplus (-1)y$ or to $y = x \ominus z = x \oplus (-1)z$ and this is the major factor for introducing the following Hukuhara difference (H-difference) [131].

Definition 4.1 (Hukuhara difference) *The definition of H-difference is suggested by* $x \ominus_H y = z \Longleftrightarrow x = y \oplus z$*; if* $x \ominus_H y$ *exists, its* α*-level is* $[x \ominus_H y]^\alpha = [\underline{x}^\alpha - \underline{y}^\alpha, \bar{x}^\alpha - \bar{y}^\alpha]$*. Precisely,* $x \ominus_H y = 0$ *but* $x \ominus y \neq 0$*.*

Definition 4.2 (Generalized Hukuhara difference) *[44] The generalized Hukuhara difference (gH-difference) associated to two fuzzy variables* x *and* y *is illustrated in a manner depicted below*

$$x \ominus_{gH} y = z \Longleftrightarrow \begin{cases} 1)\ x = y \oplus z \\ 2)\ y = x \oplus (-1)z \end{cases} \tag{4.1}$$

It is convenient to display that 1) *and* 2) *in combination are genuine if and only if* z *is a crisp number. With respect to* α*-level what we got are* $[x \ominus_{gH} y]^\alpha = [\min\{\underline{x}^\alpha - \underline{y}^\alpha, \bar{x}^\alpha - \bar{y}^\alpha\}, \max\{\underline{x}^\alpha - \underline{y}^\alpha, \bar{x}^\alpha - \bar{y}^\alpha\}]$ *and If* $x \ominus_{gH} y$ *and* $x \ominus_H y$ *subsist,* $x \ominus_H y = x \ominus_{gH} y$*. The circumstances for the inerrancy of* $z = x \ominus_{gH} y \in E$ *are*

$$\begin{aligned} 1) &\begin{cases} \underline{z}^\alpha = \underline{x}^\alpha - \underline{y}^\alpha \text{ and } \overline{z}^\alpha = \bar{x}^\alpha - \bar{y}^\alpha \\ \text{with } \underline{z}^\alpha \text{ increasing, } \overline{z}^\alpha \text{ decreasing, } \underline{z}^\alpha \leq \overline{z}^\alpha \end{cases} \\ 2) &\begin{cases} \underline{z}^\alpha = \bar{x}^\alpha - \bar{y}^\alpha \text{ and } \overline{z}^\alpha = \underline{x}^\alpha - \underline{y}^\alpha \\ \text{with } \underline{z}^\alpha \text{ increasing, } \overline{z}^\alpha \text{ decreasing, } \underline{z}^\alpha \leq \overline{z}^\alpha \end{cases} \end{aligned} \tag{4.2}$$

where $\forall\alpha \in [0,1]$

Definition 4.3 ($\alpha-$level of fuzzy function) *The $\alpha-$level of fuzzy-valued function $F : [0,a] \to E$ is*

$$F(x,\alpha) = [\underline{F}(x,\alpha), \overline{F}(x,\alpha)]$$

where $x \in E$, for each $\alpha \in [0,1]$.

With the definition of Generalized Hukuhara difference, the gH-derivative of F at x_0 is expressed as

$$\frac{d}{dt}F(x_0) = \lim_{h\to 0}\frac{1}{h}[F(x_0+h) \ominus_{gH} F(x_0)] \tag{4.3}$$

In (4.3), $F(x_0+h)$ and $F(x_0)$ exhibits similar style with x and y respectively included in (4.1).

We use the following FDE

$$\frac{d}{dt}x = f(x,t) \tag{4.4}$$

where x is the fuzzy variable $x \in E$, $\frac{d}{dt}x$ is the fuzzy derivative (see Hukuhara difference) of x, $f(x,t)$ is a fuzzy function. It is clear that the fuzzy function $f(x,u)$ is the mapping $f : [0,\zeta] \times E \to E$, where $\zeta \in \mathbb{R}$.

Let us consider the FDE (4.4) where $f : [0,\zeta] \times E \to E$. If we apply the $\alpha-$level (3.11) to $f(x,t)$ in (4.4), then we obtain two functions: $\underline{f}\,[t, \underline{x}(\zeta,\alpha), \bar{x}(\zeta,\alpha)]$ and $\overline{f}\,[t, \underline{x}(\zeta,\alpha), \bar{x}(\zeta,\alpha)]$.

The fuzzy differential equation (4.4) can be equivalent to the following four ODE

$$\begin{aligned} &1)\begin{cases} \frac{d}{dt}\underline{x} = \underline{f}\,[t, \underline{x}(\zeta,\alpha), \bar{x}(\zeta,\alpha)] \\ \frac{d}{dt}\bar{x} = \overline{f}\,[t, \underline{x}(\zeta,\alpha), \bar{x}(\zeta,\alpha)] \end{cases} \\ &2)\begin{cases} \frac{d}{dt}\underline{x} = \overline{f}\,[t, \underline{x}(\zeta,\alpha), \bar{x}(\zeta,\alpha)] \\ \frac{d}{dt}\bar{x} = \underline{f}\,[t, \underline{x}(\zeta,\alpha), \bar{x}(\zeta,\alpha)] \end{cases} \end{aligned} \tag{4.5}$$

if the following two conditions are satisfied [40]: $\underline{f}$ and $\overline{f}$ are "equivalent continuous", $\underline{f}$ and $\overline{f}$ satisfy the Lipchitz conditions

$$\begin{array}{c} \left|\overline{f}\left[t, \underline{x}(\zeta,\alpha_1), \bar{x}(\zeta,\alpha_1)\right] - \overline{f}\left[t, \underline{x}(\zeta,\alpha_2), \bar{x}(\zeta,\alpha_2)\right]\right| \\ \leq L_1 \left|\alpha_1 - \alpha_2\right| \\ \left|\underline{f}\left[t, \underline{x}(\zeta,\alpha_1), \bar{x}(\zeta,\alpha_1)\right] - \underline{f}\left[t, \underline{x}(\zeta,\alpha_2), \bar{x}(\zeta,\alpha_2)\right]\right| \\ \leq L_2 \left|\alpha_1 - \alpha_2\right| \end{array} \tag{4.6}$$

where L_1 and L_2 are positive constants.

4.2 Approximation of the solutions with Bernstein neural networks

In general, it is difficult to solve the four ODE in (4.5). Here, we use a special neural network, Bernstein neural network, to approximate the solutions of the FDE (4.4).

The two variables Bernstein series polynomial can be written as follow

$$\begin{array}{c} B(x_1, x_2) = \sum_{i=0}^{N} \sum_{j=0}^{M} \binom{N}{i} \binom{M}{j} \\ W_{i,j} x_{1i} (T - x_{1i})^{N-i} x_{2j} (1 - x_{2j})^{M-j} \end{array} \tag{4.7}$$

where $\binom{N}{i} = \frac{N!}{i!(N-i)!}$, $\binom{M}{j} = \frac{M!}{j!(M-j)!}$, $W_{i,j}$ is the coefficient. This polynomial can be regarded as neural network. It has two inputs: x_{1i} and x_{2j}, and one output y,

$$y = \sum_{i=0}^{N} \sum_{j=0}^{M} \lambda_i \gamma_j W_{i,j} x_{1i} (T - x_{1i})^{N-i} x_{2j} (1 - x_{2j})^{M-j} \tag{4.8}$$

where $\lambda_i = \binom{N}{i}$, $\gamma_j = \binom{M}{j}$.

Now we use the Bernstein neural network (4.8) to approximate the solution of the FDE (4.4). Since the solution of (4.4) can be written as four ODE as (4.5), we design the Bernstein neural network in the form of (4.5).

We assume x_1 is time interval t, x_2 is the α-level as in (3.11). The solution of (4.4) in the form of the Bernstein neural network is

$$\begin{array}{c} x_m(t,\alpha) = x_m(0,\alpha) \\ \oplus t \sum_{i=0}^{N} \sum_{j=0}^{M} \lambda_i \gamma_j W_{i,j} t_i (T-t_i)^{N-i} \alpha_j (1-\alpha_j)^{M-j} \end{array} \tag{4.9}$$

where $x_m(0,\alpha)$ is the initial condition of the solution.

We calculate the derivative of (4.8), then we obtain

$$\begin{array}{l} 1) \begin{cases} \frac{d}{dt}\underline{x}_m = C_1 + C_2 \\ \frac{d}{dt}\bar{x}_m = D_1 + D_2 \end{cases} \\ 2) \begin{cases} \frac{d}{dt}\underline{x}_m = C_1 + C_2 \\ \frac{d}{dt}\bar{x}_m = D_1 + D_2 \end{cases} \end{array} \tag{4.10}$$

where $t \in [0,T]$, $\alpha \in [0,1]$, $t_k = kh$, $h = \frac{T}{k}$, $k = 1, ..., N$, $\alpha_j = jh_1$, $h_1 = \frac{1}{M}$, $j = 1, ..., M$,

$$\begin{array}{l} C_1 = \sum_{i=0}^{N} \sum_{j=0}^{M} \lambda_i \gamma_j \underline{W}_{i,j} t_i (T-t_i)^{N-i} \alpha_j (1-\alpha_j)^{M-j} \\ D_1 = \sum_{i=0}^{N} \sum_{j=0}^{M} \lambda_i \gamma_j \bar{W}_{i,j} t_i (T-t_i)^{N-i} \alpha_j (1-\alpha_j)^{M-j} \\ C_2 = t_k \sum_{i=0}^{N} \sum_{j=0}^{M} \lambda_i \gamma_j \underline{W}_{i,j} [i t_{i-1,j} (T-t_i)^{N-i} \\ -(N-i)\, t_{i,j} (T-t_i)^{N-i-1}] \alpha_j^i (1-\alpha_j)^{M-j} \\ D_2 = t_k \sum_{i=0}^{N} \sum_{j=0}^{M} \lambda_i \gamma_j \bar{W}_{i,j} [i t_{i-1,j} (T-t_i)^{N-i} \\ -(N-i)\, t_{i,j} (T-t_i)^{N-i-1}] \alpha_j^i (1-\alpha_j)^{M-j} \end{array}$$

where $d = 0.02$, $E = 70 \times 10^9$. The exact solution is $x = 4$.

- input unit:

$$o_1^1 = t, \quad o_2^1 = \alpha$$

- the first hidden units:

$$\begin{array}{l} o_{1,i}^2 = f_i^1(o_1^1), \quad o_{2,i}^2 = f_i^2(o_1^1) \\ o_{3,j}^2 = g_j^1(o_2^1), \quad o_{4,j}^2 = g_j^2(o_2^1) \end{array}$$

- the second hidden units:

$$o_{1,i}^3 = o_{1,i}^2(o_{2,i}^2), \quad o_{2,j}^3 = o_{3,j}^2(o_{4,j}^2)$$

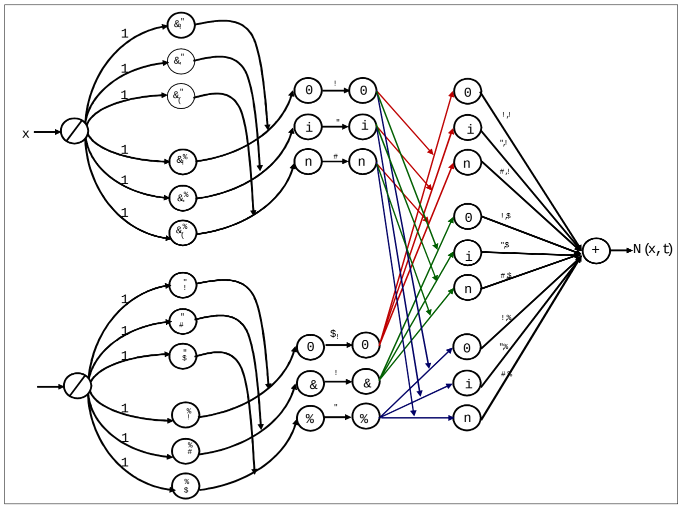

Figure 4.1: Static Bernstein neural network

- the third hidden units:

$$o^4_{1,i} = \lambda_i o^3_{1,i}, \quad o^4_{2,i'} = \gamma_j o^3_{2,j}$$

- the forth hidden units:

$$o^5_{i,j} = o^4_{1,i} o^4_{2,j}$$

- output unit:

$$N(t, \alpha) = \sum_{i=0}^{N} \sum_{j=0}^{M} (a_{i,j} o^5_{i,j})$$

Here $f^1_i = t^i$, $f^2_i = (T-t)^{N-i}$, $\lambda_i = \frac{1}{T^N}\binom{N}{i}$, $g^1_{i'} = \alpha^j$, $g^2_j = (1-\alpha)^{M-j}$, $\gamma_j = \binom{M}{j}$.
The training errors between (4.10) and (4.5) are defined as

$$\begin{aligned} &1)\begin{cases} \underline{e}_1 = C_1 + C_2 - \underline{f} \\ \bar{e}_1 = D_1 + D_2 - \bar{f} \end{cases} \\ &2)\begin{cases} \underline{e}_2 = C_1 + C_2 - \bar{f} \\ \bar{e}_2 = D_1 + D_2 - \underline{f} \end{cases} \end{aligned} \tag{4.11}$$

The standard back-propagation learning algorithm is utilized to update the weights with the above training errors

$$\begin{array}{c} \underline{W}_{i,j}\left(k+1\right)=\underline{W}_{i,j}\left(k\right)-\eta_1(\frac{\partial\underline{e}_1^2}{\partial\underline{W}_{i,j}}+\frac{\partial\bar{e}_1^2}{\partial\underline{W}_{i,j}}) \\ \bar{W}_{i,j}\left(k+1\right)=\bar{W}_{i,j}\left(k\right)-\eta_2(\frac{\partial\underline{e}_2^2}{\partial\bar{W}_{i,j}}+\frac{\partial\bar{e}_2^2}{\partial\bar{W}_{i,j}}) \end{array} \tag{4.12}$$

here η_1 and η_1 are positive learning rates. The momentum terms, $\gamma\Delta\underline{W}_{i,j}\left(k-1\right)$ and $\gamma\Delta\bar{W}_{i,j}\left(k-1\right)$, can be added to stabilized the training process.

The Bernstein neural network (4.9) shown in Figure 4.1 is feedforward (static) neural network. We can also use a recurrent (dynamic) neural network to approximate the solution of (4.4).

The dynamic Bernstein neural network is

$$\begin{cases} \frac{d}{dt}\underline{x}_m(t,\alpha)=\underline{P}(t,\alpha)A(\underline{x}_m(t,\alpha),\bar{x}_m(t,\alpha))+\underline{Q}(t,\alpha) \\ \frac{d}{dt}\bar{x}_m(t,\alpha)=\overline{P}(t,\alpha)A(\underline{x}_m(t,\alpha),\bar{x}_m(t,\alpha))+\overline{Q}(t,\alpha) \end{cases} \tag{4.13}$$

where $f(x,t)$ in (4.4) has the form of

$$f(t,x)=P(t)x+Q(t)$$

The structure of the dynamic Bernstein neural network is shown in Figure 4.2.

The training errors in (4.11) are changed into

$$\begin{array}{l} 1)\begin{cases} \underline{e}_1=C_1+C_2-\underline{P}A(\underline{x}_m,\bar{x}_m)-\underline{Q} \\ \bar{e}_1=D_1+D_2-\overline{P}A(\underline{x}_m,\bar{x}_m)-\overline{Q} \end{cases} \\ 2)\begin{cases} \underline{e}_2=C_1+C_2-\overline{P}A(\underline{x}_m,\bar{x}_m)-\overline{Q} \\ \bar{e}_2=D_1+D_2-\underline{P}A(\underline{x}_m,\bar{x}_m)-\underline{Q} \end{cases} \end{array} \tag{4.14}$$

The the training algorithm can be the same as the (4.12).

The leaning process of the dynamic Bernstein neural network (4.13) is faster then the static Bernstein neural network (4.9). However, the robustness of (4.9) is better than (4.13), because the weights of the dynamic Bernstein neural network are difficult to convergence.

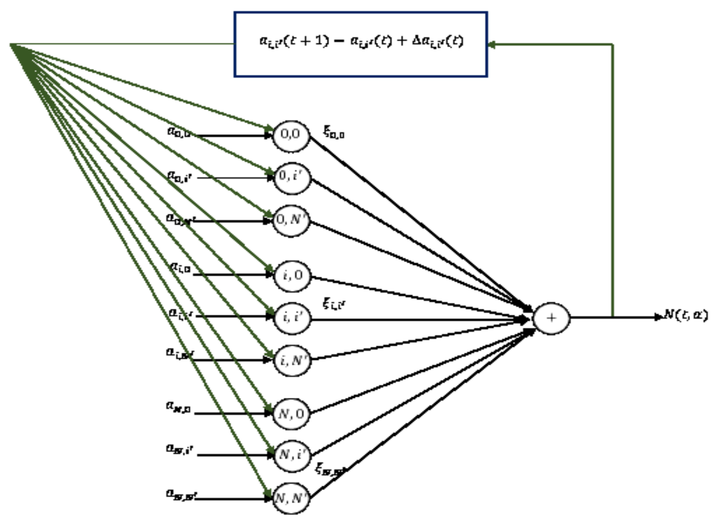

Figure 4.2: Dynamic Bernstein nerual network

4.3 Solutions with strongly degenerate parabolic and Burgers-Fisher equations

Definition 4.4 (Second-order nonlinear PDE) *The second-order singular nonlinear PDE can be portrayed using the equation below*

$$\frac{\partial^2 u(x,t)}{\partial t^2} + \frac{2}{t}\frac{\partial u(x,t)}{\partial t} = F(x, u(x,t), \frac{\partial u(x,t)}{\partial x}, \frac{\partial^2 u(x,t)}{\partial x^2}) \tag{4.15}$$

where t and x are independent variables, u is the dependent variable, and F is a nonlinear function of x, u, u_x and u_{xx}, also the initial conditions for the PDE (4.15) are as follows:

$$u(x,0) = f(x), \quad u_t(x,0) = g(x)$$

Definition 4.5 (Strongly degenerate parabolic equation) *The strongly degenerate parabolic equation has been explained as follows*

$$\frac{\partial u(x,t)}{\partial t} + \frac{\partial Q(u(x,t))}{\partial x} = \frac{\partial^2 A(u(x,t))}{\partial x^2}, \quad (x,t) \in \Pi_T := [0,1] \times (0,T), \quad T > 0 \tag{4.16}$$

consider the boundary conditions as:

$$u(x,0) = g_0(x), \quad u(0,t) = f_0(t), \quad u(1,t) = f_1(t)$$

where the integrated diffusion coefficient A is exhibited by

$$A(u) = \int_0^u a(s)ds, \quad a(u) \geq 0, \quad a \in L^\infty([0,1]) \cap L^1([0,1])$$

The function a is allowed to terminate on u-intervals of positive length, on which Eq. (4.16) degenerates to a first-order scalar conservation law. Hence, Eq. (4.16) is termed as strongly degenerate.

Definition 4.6 (Burgers-Fisher equation) *The generalized Burgers-Fisher equation is defined as*

$$\frac{\partial u(x,t)}{\partial t} + \alpha u^\sigma(x,t)\frac{\partial u(x,t)}{\partial x} - \frac{\partial^2 u(x,t)}{\partial x^2} = \beta u(x,t)(1 - u^\sigma(x,t)) \tag{4.17}$$

$$(x,t) \in \Pi_T := [0,1] \times [0,T], \quad T > 0$$

with initial and boundary conditions:

$$g_0(x) := u(x,0) = (\frac{1}{2} + \frac{1}{2}\tanh(\frac{-\alpha\sigma}{2(\sigma+1)}x))^{\frac{1}{\sigma}}, \quad 0 \leq x \leq 1$$

$$f_0(t) := u(0,t) = (\frac{1}{2} + \frac{1}{2}\tanh(\frac{-\alpha\sigma}{2(\sigma+1)}(-(\frac{\alpha}{\sigma+1} + \frac{\beta(\sigma+1)}{\alpha})t)))^{\frac{1}{\sigma}}, \quad 0 \leq x \leq 1, \quad t \geq 0$$

$$f_1(t) := u(1,t) = (\frac{1}{2} + \frac{1}{2}\tanh(\frac{-\alpha\sigma}{2(\sigma+1)}(1-(\frac{\alpha}{\sigma+1} + \frac{\beta(\sigma+1)}{\alpha})t)))^{\frac{1}{\sigma}}, \quad 0 \leq x \leq 1, \quad t \geq 0$$

where α, β, and σ are constants.

Here a three layer neural network with two input signals, m hidden neurons and one output signal is formulated in order to solve the strongly degenerate parabolic and Burgers-Fisher equations which depends on the function approximation capability of the neural network and repays the solution of differential equations in a closed analytic and differentiable form (see Figure 4.3). The input-output connection of each unit of the suggested neural network is written as:

• Input units:

$$o_1^1 = x \tag{4.18}$$

$$o_2^1 = t$$

• Hidden units:

$$o_j^2 = F(b_j + w_j^1 x + w_j^2 t) \quad j = 1, ..., m \tag{4.19}$$

• Output unit:

$$N(x,t) = \sum_{j=1}^{m}(W_j o_j^2) \tag{4.20}$$

Here, the most common function $F(r) = \frac{2}{1+e^{-2r}} - 1$ (tan-sigmoid function) which is termed to be a continuously differentiable nonlinear function, is accepted as an activation function

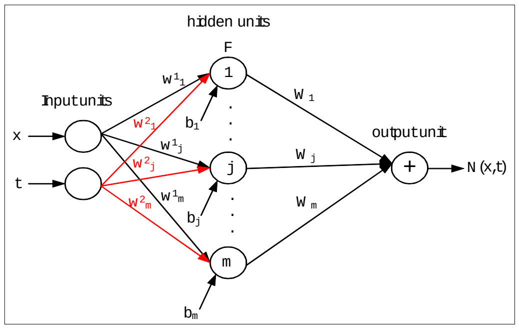

Figure 4.3: Neural network (NN) equivalent to strongly degenerate parabolic and Burgers-Fisher equations

of the hidden units. In order to set the boundary conditions, we select a trial solution which can be depicted as sum of the two terms as

$$u_m(x,t) = (1-x)f_0(t)+xf_1(t)+(1-t)\{g_0(x)-[(1-x)g_0(0)+xg_0(1)]\}+x(1-x)tN(x,t) \quad (4.21)$$

where

$$N(x,t) = \sum_{j=1}^{m}(W_jF(b_j + w_j^1x + w_j^2t)) \quad (4.22)$$

If we suppose that $0 \leq x \leq 1$, the rectangle $[0,1] \times [0,T]$ can be divided into nn' mesh points $(x_i,t_j) = ((i-1)h,(j-1)h')$, $h = \frac{1}{n-1}$, $h' = \frac{T}{n'-1}$, $(i = 1,...,n; j = 1,...,n')$. In the given problems, the substitution of approximate solution $u_m(x,t)$ instead of the unknown function will result the following least mean square error for the relation mentioned $(x,t) = (x_i,t_j)$ as follows:

$$E_{i,j} = \frac{1}{2}(M_{i,j})^2 \quad (4.23)$$

where for Burgers-Fisher equation

$$M_{i,j} = \frac{\partial u_m(x,t)}{\partial t}\Big|_{\substack{x=x_i\\ t=t_j}} + \alpha u_m^{\sigma}(x,t)\Big|_{\substack{x=x_i\\ t=t_j}} \frac{\partial u_m(x,t)}{\partial x}\Big|_{\substack{x=x_i\\ t=t_j}} - \frac{\partial^2 u_m(x,t)}{\partial x^2}\Big|_{\substack{x=x_i\\ t=t_j}}$$

$$-\beta u_m(x,t)|_{\substack{x=x_i\\t=t_j}} (1-u_m^{\sigma}(x,t)|_{\substack{x=x_i\\t=t_j}})$$

and for strongly degenerate parabolic equation

$$M_{i,j}=\frac{\partial u_m(x,t)}{\partial t}|_{\substack{x=x_i\\t=t_j}}+\frac{\partial Q(u_m(x,t))}{\partial x}|_{x=x_i,t=t_j}-\frac{\partial^2 A(u_m(x,t))}{\partial x^2}|_{\substack{x=x_i\\t=t_j}}$$

Generally the summed up error of the suggested neural network is exhibited as:

$$E=\sum_{i=1}^{n}\sum_{j=1}^{n'}E_{i,j}=\sum_{i,j}E_{i,j} \tag{4.24}$$

The proposed learning rule can be systematically extracted as minimizers of the referred error function. The above statement indicates that the main intention in the remaining part of this section is to train the proposed network architecture in order to complete this task. Consideration of above intention will result in the starting of derivation of a learning procedure that is considered to be a natural generalization of the Newton method in order to adjust network parameters (weights and biases). The Newton's rule is carried out on the basis of cost function $E_{i,j}$, to calculate the weight modifications W_q as [172]:

$$W_q(r+1)=W_q(r)-\mu(r)\frac{\partial E_{i,j}}{\partial W_q},\quad q=1,...,m \tag{4.25}$$

where μ is the training rate $\mu>0$. In order to increase training process, we add a momentum term as

$$W_q(r+1)=W_q(r)-\mu(r)\frac{\partial E_{i,j}}{\partial W_q}+\gamma[W_q(r)-W_q(r-1)]$$

where $\gamma>0$. The index r refers to the iteration number. Apart from that, $W_q(r+1)$ and $W_q(r)$ exhibit the upgraded and recent output weight values, respectively. Now, the explicit methodology of (4.25) can be potrayed as follows:

$$\begin{bmatrix} W_1\\ \vdots\\ W_m\end{bmatrix}_{r+1}=\begin{bmatrix} W_1\\ \vdots\\ W_m\end{bmatrix}_{r}-\frac{(\nabla E_{i,j}(W)_r)^T\nabla E_{i,j}(W)_r}{(\nabla E_{i,j}(W)_r)^T Q_r\nabla E_{i,j}(W)_r}\nabla E_{i,j}(W)_r+\gamma\begin{bmatrix} \Delta W_1\\ \vdots\\ \Delta W_m\end{bmatrix}_{r-1} \tag{4.26}$$

where

$$\nabla E_{i,j}(W) = (\frac{\partial E_{i,j}}{\partial W_1}, \cdots, \frac{\partial E_{i,j}}{\partial W_m})^T$$

and

$$Q = \begin{bmatrix} \frac{\partial^2 E_{i,j}}{\partial W_1^2} & \frac{\partial^2 E_{i,j}}{\partial W_1 \partial W_2} & \cdots & \frac{\partial^2 E_{i,j}}{\partial W_1 \partial W_m} \\ \frac{\partial^2 E_{i,j}}{\partial W_2 \partial W_1} & \frac{\partial^2 E_{i,j}}{\partial W_2^2} & \cdots & \frac{\partial^2 E_{i,j}}{\partial W_2 \partial W_m} \\ \cdots & \cdots & \cdots & \cdots \\ \frac{\partial^2 E_{i,j}}{\partial W_m \partial W_1} & \frac{\partial^2 E_{i,j}}{\partial W_m \partial W_2} & \cdots & \frac{\partial^2 E_{i,j}}{\partial W_m^2} \end{bmatrix} \tag{4.27}$$

are computed at the current mesh points (x_i, t_j). Here Q is the Hessian matrix with components $\frac{\partial^2 E_{i,j}}{\partial W_q \partial W_p}$ (for $q, p = 1, ..., m$). It is quite obvious that the convergence speed is in direct relation with the learning rate. In order to achieve optimal learning rate for rapid convergence of the learning optimization rule, the inverse of Hessian matrix Q of the error function $E_{i,j}$ is impinged at the current mesh points. The approximate Newton method illustrated above is well sufficient to scale the descent step in each step. The usage of chain rule for differentiation will result in the present partial derivative as mentioned below:

$$\frac{\partial E_{i,j}}{\partial W_q} = \frac{\partial E_{i,j}}{\partial M_{i,j}}.\frac{\partial M_{i,j}}{\partial W_q} = M_{i,j}.\frac{\partial M_{i,j}}{\partial W_q}$$

For Burgers-Fisher equation assume that

$$K_1(x,t) = \frac{\partial u(x,t)}{\partial t}, \quad K_2(x,t) = u^\sigma(x,t)\frac{\partial u(x,t)}{\partial x}$$

$$K_3(x,t) = \frac{\partial^2 u(x,t)}{\partial x^2}, \quad K_4(x,t) = u(x,t)(1 - u^\sigma(x,t))$$

then we have

$$\frac{\partial M_{i,j}}{\partial W_q} = \frac{\partial M_{i,j}}{\partial K_1(x_i,t_j)}.\frac{\partial K_1(x_i,t_j)}{\partial W_q} + \alpha\frac{\partial M_{i,j}}{\partial K_2(x_i,t_j)}.\frac{\partial K_2(x_i,t_j)}{\partial W_q}$$
$$-\frac{\partial M_{i,j}}{\partial K_3(x_i,t_j)}.\frac{\partial K_3(x_i,t_j)}{\partial W_q} - \beta\frac{\partial M_{i,j}}{\partial K_4(x_i,t_j)}.\frac{\partial K_4(x_i,t_j)}{\partial W_q}$$

For strongly degenerate parabolic equation that has been used in the stated methodology, $\frac{\partial M_{i,j}}{\partial W_q}$ can be accomplished in a same manner. When the above relation is substituted into (4.26), we will achieve the desired learning rule. The learning procedure stated above can also be widespread to the other network parameters (w_q^1, w_q^2 and b_q) in a similar way.

4.4 Solution with wave equation

Definition 4.7 (wave equation) *The Cauchy problem for the wave equation in one dimension can be stated as*

$$\frac{\partial^2 u(x,t)}{\partial t^2} + c^2 \frac{\partial^2 u(x,t)}{\partial x^2} = f(x,t), \quad (x,t) \in [0,a] \times [0,b] \tag{4.28}$$

with

$$u(x,0) = \phi(x), \quad u_t(x,0) = \psi(x)$$

where a and b are constants. In above equation the parameter c is called the speed of light.

We continue our discussion of solving PDEs with the help of two pattern of neural networks and the application of Bernstein polynomial. Consider the Cauchy problem (4.28), where the solution u depends on both spatial and temporal variables x and t respectively. The trial solution is written as:

$$u_m(x,t) = \phi(x) + t\psi(x) + t[B(x,t) - B(x,0) - \frac{\partial B(x,0)}{\partial t}]$$

where $B(x,t)$ is the bivariate Bernstein polynomial series of solution function $u(x,t)$, namely:

$$B(x,t) = \sum_{i=0}^{n} \sum_{j=0}^{m} \binom{n}{i} \binom{m}{j} \frac{x^i (a-x)^{n-i}}{a^n} \frac{t^j (b-t)^{m-j}}{b^m} q_{i,j}(x,t), \quad n,m \in N$$

$$B(x,t) = \sum_{i=0}^{n} \sum_{j=0}^{m} \beta_{i,j} x^i (a-x)^{n-i} t^j (b-t)^{m-j} q_{i,j}(x,t), \quad n,m \in N, \quad \beta_{i,j} = \frac{1}{a^n b^m} \binom{n}{i} \binom{m}{j} \tag{4.29}$$

where

$$\binom{n}{i} = \frac{n!}{i!(n-i)!}, \quad \binom{m}{j} = \frac{m!}{j!(m-j)!}$$

Consider the following relations:

$$\frac{\partial^2 u_m(x,t)}{\partial x^2} = \phi''(x) + t\psi''(x) + t[\frac{\partial^2 B(x,t)}{\partial x^2} - \frac{\partial^2 B(x,0)}{\partial x^2} - \frac{\partial^2 \partial B(x,0)}{\partial x^2 \partial t}]$$

and

$$\frac{\partial^2 u_m(x,t)}{\partial t^2} = 2\frac{\partial B(x,t)}{\partial t} + t\frac{\partial^2 B(x,t)}{\partial t^2}$$

Substituting the above relations in the origin problem (4.28) leads to the following differential equation:

$$2\frac{\partial B(x,t)}{\partial t} + t\frac{\partial^2 B(x,t)}{\partial t^2} + c^2(\phi''(x) + t\psi''(x) + t[\frac{\partial^2 B(x,t)}{\partial x^2} - \frac{\partial^2 B(x,0)}{\partial x^2} - \frac{\partial^2 \partial B(x,0)}{\partial x^2 \partial t}]) = f(x,t) \tag{4.30}$$

$$(x,t) \in [0,a] \times [0,b]$$

For simplicity the above relation can be rewritten as follows:

$$\sum_{i=0}^{n}\sum_{j=0}^{m} \xi_{i,j}(x,t) q_{i,j}(x,t) + \zeta(x,t) = f(x,t), \quad (x,t) \in [0,a] \times [0,b] \tag{4.31}$$

where

$$\begin{gathered}
\xi_{i,j}(x,t) = 2\beta_{i,j} x^i (a-x)^{n-i} (j t^{j-1} (b-t)^{m-j} - (m-j) t^j (b-t)^{m-j-1}) \\
+ t\beta_{i,j} x^i (a-x)^{n-i} (j(j-1) t^{j-2} (b-t)^{m-j} - 2j(m-j) t^{j-1} (b-t)^{m-j-1} \\
+ (m-j)(m-j-1) t^j (b-t)^{m-j-2}) + c^2 t \beta_{i,j} (i(i-1) x^{i-2} (a-x)^{n-i} \\
- 2i(n-i) x^{i-1} (a-x)^{n-i-1} + (n-i)(n-i-1) x^i (a-x)^{n-i-2}) t^j (b-t)^{m-j} \\
- c^2 t \beta_{i,j} (i(i-1) x^{i-2} (a-x)^{n-i} - 2i(n-i) x^{i-1} (a-x)^{n-i-1} \\
+ (n-i)(n-i-1) x^i (a-x)^{n-i-2}) (j t^{j-1} (b-t)^{m-j} - (m-j) t^j (b-t)^{m-j-1})
\end{gathered}$$

and

$$\zeta(x,t) = c^2(\phi''(x) + t\psi''(x) + t\frac{\partial^2 B(x,0)}{\partial x^2})$$

We design a neural network to represent the equation (4.29), see Figure 4.2.

In the above architecture the mathematical symbol is defined as:

$$\varphi_{i,j} = \sum_{i=0}^{n}\sum_{j=0}^{m} \binom{n}{i}\binom{m}{j} \frac{x^i (a-x)^{n-i}}{a^n} \frac{t^j (b-t)^{m-j}}{b^m}, \quad n,m \in N$$

The input-output relation of each unit in the proposed neural architecture can be summarized as follows:

• Input unit:

$$o_{i,j} = q_{i,j}, \quad i = 0, ..., n, \quad j = 0, ..., m$$

• Output unit:

$$N(x,t) = \varphi_{i,j} o_{i,j}$$

Now, a suitable numerical technique should be able to provide an appropriate tool for measuring and calculating the accuracy of obtained solution. Hence, to compare the exact solution with its obtained one, the least mean square error is used, which is stated as follows:

$$E_{i,j} = \frac{1}{2}(\sum_{i=0}^{n}\sum_{j=0}^{m} \xi_{i,j}(x,t) q_{i,j}(x,t) + \zeta(x,t) - f(x,t))^2$$

We use Newton's rule as described in (4.26) for adjusting the parameters such that the network error to be minimized over the space of weights setting. The initial parameter $q_{i,j}$ is selected randomly to begin the procedure. The described standard self learning mechanism works as follows:

$$q_{i,j}(r+1) = q_{i,j}(r) - \mu(r)\frac{\partial E_{i,j}}{\partial q_{i,j}}$$

where μ is the training rate $\mu > 0$. In order to increase training process, we add a momentum term as

$$q_{i,j}(r+1) = q_{i,j}(r) - \mu(r)\frac{\partial E_{i,j}}{\partial q_{i,j}} + \gamma[q_{i,j}(r) - q_{i,j}(r-1)]$$

where $\gamma > 0$. The index r refers to the iteration number.

Consider another type of neural network architecture shown in Figure 4.1. The input-output relation of each unit in the proposed neural architecture can be summarized as follows:

• Input unit:

$$o_1^1 = x$$

$$o_2^1 = t$$

• The first hidden units:

$$o_{1,i}^2 = f_i^1(o_1^1), \quad o_{2,i}^2 = f_i^2(o_1^1), \quad i = 0, ..., n$$

$$o^2_{3,j} = g^1_j(o^1_2), \quad o^2_{4,j} = g^2_j(o^1_2), \quad j = 0, ..., m$$

- The second hidden units:

$$o^3_{1,i} = o^2_{1,i}(o^2_{2,i}), \quad i = 0, ..., n$$

$$o^3_{2,j} = o^2_{3,j}(o^2_{4,j}), \quad j = 0, ..., m$$

- The third hidden units:

$$o^4_{1,i} = \lambda_i o^3_{1,i}, \quad i = 0, ..., n$$

$$o^4_{2,j} = \gamma_j o^3_{2,j}, \quad j = 0, ..., m$$

- The forth hidden units:

$$o^5_{i,j} = o^4_{1,i} o^4_{2,j}, \quad i = 0, ..., n, \quad j = 0, ..., m$$

- Output unit:

$$N(x,t) = \sum_{i=0}^{n} \sum_{j=0}^{m} (q_{i,j} o^5_{i,j})$$

In above relations we have:

$$f^1_i = x^i, \quad f^2_i = (a-x)^{n-i}, \quad \lambda_i = \frac{1}{a^n}\binom{n}{i}, \quad i = 0, ..., n$$

$$g^1_j = t^j, \quad g^2_j = (b-t)^{m-j}, \quad \gamma_j = \frac{1}{b^m}\binom{m}{j}, \quad j = 0, ..., m$$

4.5 Simulations

In this section, we use several real applications to show how to use the Bernstein neural networks to approximate the solutions of the FDEs.

Example 4.1 *The vibration mass which is shown in Figure 4.4 has a very simple model*

$$\frac{d}{dt}x(t) = \frac{k}{m}x(t) \tag{4.32}$$

where the spring constant $k = 1$, *the mass* m *are changeable in* $(0.75, 1.125)$, *so the position state* $x(t)$ *has some uncertainties, the ODE (4.32) can be formed into a FDE. It has the same form as (4.32), only* $x(t)$ *becomes a fuzzy variable. If the initial position is* $x(0) = (0.75 + 0.25\alpha, 1.125 - 0.125\alpha)$, $\alpha \in [0, 1]$, *then the exact solutions of the FDE (4.32) are [97]*

$$x(t, \alpha) = \left[(0.75 + 0.25\alpha)e^t, (1.125 - 0.125\alpha)e^t\right] \tag{4.33}$$

where $t \in [0, 1]$. *Now we use the static Bernstein neural network (4.9), noted as SNN, to approximate the solution (4.33)*

$$\begin{cases} \underline{x}_m(t, \alpha) = (0.75 + 0.25\alpha) \\ +t \sum_{i=0}^{N} \sum_{j=0}^{M} \lambda_i \gamma_j \underline{W}_{i,j} t_i (T - t_i)^{N-i} \alpha_j (1 - \alpha_j)^{M-j} \\ \bar{x}_m(t, \alpha) = (1.125 - 0.125\alpha) \\ +t \sum_{i=0}^{N} \sum_{j=0}^{M} \lambda_i \gamma_j \bar{W}_{i,j} t_i (T - t_i)^{N-i} \alpha_j (1 - \alpha_j)^{M-j} \end{cases}$$

We also use dynamic Bernstein neural network (4.13), noted as DNN, to approximated the solutions. The learning rates are $\eta = 0.01$, $\gamma = 0.01$ *To compare our results, we use the other two popular methods: Max-Min Euler method and Average Euler method [205]. The comparison results are shown in Table 1 and Table 2.*

Table 1. Solutions of different method

α	*Exact solution*	*SNN*	*DNN*	*Max-Min Euler*	*Average Euler*
0	[2.0387,3.0581]	[1.9713,3.0043]	[1.9901,3.0305]	[1.9453,2.9180]	[2.2441,2.6191]
0.1	[2.1067,3.0241]	[2.0302,2.9415]	[2.0591,2.9752]	[2.0102,2.8855]	[2.2791,2.6166]
0.2	[2.1746,2.9901]	[2.1059,2.9131]	[2.1283,2.9399]	[2.0750,2.8531]	[2.3140,2.6140]
0.3	[2.2426,2.9561]	[2.1618,2.8707]	[2.1901,2.8931]	[2.1398,2.8207]	[2.3490,2.6115]
0.4	[2.3105,2.9222]	[2.2307,2.8453]	[2.2601,2.8799]	[2.2047,2.7883]	[2.3840,2.6090]
0.5	[2.3785,2.8882]	[2.2984,2.8088]	[2.3288,2.8337]	[2.2695,2.7559]	[2.4189,2.6064]
0.6	[2.4465,2.8542]	[2.3631,2.7784]	[2.3904,2.7955]	[2.3344,2.7234]	[2.4539,2.6039]
0.7	[2.5144,2.8202]	[2.4292,2.7449]	[2.4555,2.7691]	[2.3992,2.6910]	[2.4888,2.6013]
0.8	[2.5824,2.7862]	[2.4895,2.7067]	[2.5101,2.7302]	[2.4641,2.6586]	[2.5238,2.5988]
0.9	[2.6503,2.7523]	[2.5564,2.6769]	[2.5821,2.7001]	[2.5289,2.6262]	[2.5588,2.5963]
1	[2.7183,2.7183]	[2.6199,2.6399]	[2.6414,2.6614]	[2.5937,2.5937]	[2.5937,2.5937]

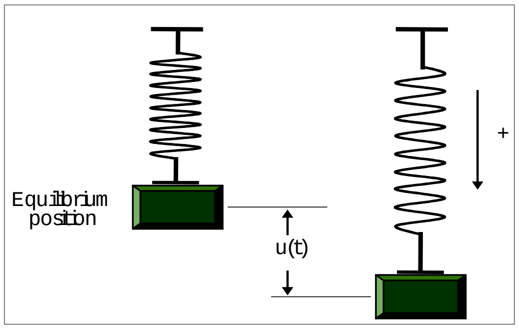

Figure 4.4: Vibration mass

Table 2. Approximation errors

α	*SNN*	*DNN*	*Max-Min Euler*	*Average Euler*
0	*[0.0601,0.1098]*	*[0.0207,0.0601]*	*[0.0934,0.1401]*	*[0.2054,0.4390]*
0.2	*[0.0658,0.1067]*	*[0.0241,0.0612]*	*[0.0996,0.1370]*	*[0.1394,0.3761]*
0.6	*[0.0798,0.1022]*	*[0.0322,0.0654]*	*[0.1121,0.1308]*	*[0.0074,0.2503]*
0.8	*[0.0791,0.0891]*	*[0.0328,0.0499]*	*[0.1183,0.1276]*	*[0.0586,0.1874]*
1	*[0.0921,0.0921]*	*[0.0534,0.0534]*	*[0.1246,0.1246]*	*[0.1246,0.1246]*

Example 4.2 *The heat treatment in welding can be modeled as [61]:*

$$\frac{d}{dt}x(t) = 3Ax^2(t) \tag{4.34}$$

where transfer area A *is uncertainty as* $A = (1+\alpha, 3-\alpha)$, $\alpha \in [0,1]$. *So (4.34) is a FDE. If the initial condition is* $x(0) = (0.5\sqrt{\alpha}, 0.2\sqrt{1-\alpha}+0.5)$, *the static Bernstein neural network (4.9) has the form of*

$$\begin{cases} \underline{x}_m(t,\alpha) = 0.5\sqrt{\alpha} \\ +t\sum_{i=0}^{N}\sum_{j=0}^{M}\lambda_i\gamma_j\underline{W}_{i,j}t_i(T-t_i)^{N-i}\alpha_j(1-\alpha_j)^{M-j} \\ \bar{x}_m(t,\alpha) = 0.2\sqrt{1-\alpha}+0.5 \\ +t\sum_{i=0}^{N}\sum_{j=0}^{M}\lambda_i\gamma_j\bar{W}_{i,j}t_i(T-t_i)^{N-i}\alpha_j(1-\alpha_j)^{M-j} \end{cases}$$

With the learning rates $\eta = 0.002$ *and* $\gamma = 0.002$, *the approximation results are shown in Table 3.*

Table 3. Bernstein neural network approximation

α	SNN	DNN
0	[0.0511,0.0754]	[0.0224,0.0381]
0.1	[0.0402,0.0623]	[0.0203,0.0362]
0.2	[0.0398,0.0588]	[0.0197,0.0374]
0.3	[0.0224,0.0312]	[0.0211,0.0321]
0.4	[0.0433,0.0613]	[0.0246,0.0462]
0.5	[0.0507,0.0631]	[0.0152,0.0258]
0.6	[0.0469,0.0726]	[0.0191,0.0392]
0.7	[0.0571,0.0778]	[0.0288,0.0452]
0.8	[0.0373,0.0509]	[0.0157,0.0362]
0.9	[0.0401,0.0635]	[0.0202,0.0408]
1	[0.0394,0.0394]	[0.0167,0.0167]

4.6 Conclusions

In this chapter, the solutions of the FDEs are approximated by two types of Bernstein neural networks. First the FDE is transformed into four ODEs with Hukuhara differentiability. Then neural models are constructed with the structure of ODEs. The modified backpropagation method for fuzzy variables is used for training the neural networks. The theory analysis and simulation results show that these new models, Bernstein neural networks, are effective to estimate the solutions of FDEs.

Also a methodology involving novel iterative technique considering neural networks is suggested to extract approximate solution for the second-order nonlinear PDEs. This perspective is designed to grant good approximation on the basis of learning technique which is associated with quasi-Newton rule. Furthermore to continue, a sophisticated methodology is provided in order to solve PDEs on the basis of the application of two patterned Bernstein neural networks, static and dynamic models.

Chapter 5

Uncertain nonlinear system control with Z-numbers

The decisions are carried out on the basis of knowledge. In order to make the decision fruitful, the knowledge acquired must be credible. Z-numbers connect to the reliability of knowledge [221]. Many fields related to the analysis of the decisions are actually use the ideas of Z-numbers. Z-numbers are much less complex to calculate compared with nonlinear system modeling methods. The Z-number is abundantly adequate number compared with the fuzzy number. Although Z-numbers are implemented in many literatures, from theoretical point of view this approach is not certified completely.

The Z-number is a novel idea that is subjected to a higher potential in order to illustrate the information of the human being and to use it in information processing [221]. Z-numbers can be regarded as to answer questions and carry out the decisions [121]. There are few structure based on the theoretical concept of Z-numbers [87]. [19] gave an inception which results in the extension of the Z-numbers. [122] proposed a theorem to transfer the Z-numbers to the usual fuzzy sets.

In this chapter, dual fuzzy equations [213] and FDEs are used to model the uncertain nonlinear systems, where the coefficients are Z-numbers. Then the existence of the solutions of the dual fuzzy equations and FDEs is discussed. It corresponds to controllability problem of the fuzzy control [65]. Two types of neural networks, feed-forward and feedback networks are used to approximate the solutions (control actions) of the dual fuzzy equation and FDE. Several real examples are employed to show the effectiveness of the fuzzy control design methods.

5.1 Nonlinear system modeling with dual fuzzy equations and Z-numbers

The following definitions will be used in this chapter.

Definition 5.1 (Z-numbers) *A Z-number has two components $Z = [A(x), p]$. The primary component $A(x)$ is termed as a restriction on a real-valued uncertain variable x. The secondary component p is a measure of reliability of A. p can be reliability, strength of belief, probability or possibility. When $A(x)$ is a fuzzy number and p is the probability distribution of x, the Z-number is defined as Z^+-number. When both $A(x)$ and p are fuzzy numbers, the Z-number is defined as Z^--number.*

The Z^+-number carries more information than the Z^--number. Here we use the definition of Z^+-number, i.e., $Z = [A, p]$, A is a fuzzy number, p is a probability distribution.

We use membership functions to express the fuzzy number. The most popular membership functions are the triangular function

$$\mu_A = F(a, b, c) = \begin{cases} \frac{x-a}{b-a} & a \le x \le b \\ \frac{c-x}{c-b} & b \le x \le c \end{cases} \text{ otherwise } \mu_A = 0 \tag{5.1}$$

and trapezoidal function

$$\mu_A = F(a, b, c, d) = \begin{cases} \frac{x-a}{b-a} & a \le x \le b \\ \frac{d-x}{d-c} & c \le x \le d \\ 1 & b \le x \le c \end{cases} \text{ otherwise } \mu_A = 0 \tag{5.2}$$

The probability measure is expressed as

$$P = \int_R \mu_A(x)p(x)dx \tag{5.3}$$

where p is the probability density of x, R is the restriction on p. For discrete Z-numbers

$$P(A) = \sum_{i=1}^{n} \mu_A(x_i)p(x_i) \tag{5.4}$$

Definition 5.2 (α-level of Z-numbers) *The α-level of the Z-number $Z = (A, P)$ is demonstrated as*

$$[Z]^\alpha = ([A]^\alpha, [p]^\alpha) \tag{5.5}$$

where $0 < \alpha \leq 1$. $[p]^\alpha$ is calculated by the Nguyen's theorem

$$[p]^\alpha = p([A]^\alpha) = p([\underline{A}^\alpha, \overline{A}^\alpha]) = \left[\underline{P}^\alpha, \overline{P}^\alpha\right]$$

where $p([A]^\alpha) = \{p(x)|x \in [A]^\alpha\}$. So $[Z]^\alpha$ can be expressed as the form α-level of a fuzzy number

$$[Z]^\alpha = \left(\underline{Z}^\alpha, \overline{Z}^\alpha\right) = \left(\left(\underline{A}^\alpha, \underline{P}^\alpha\right), \left(\overline{A}^\alpha, \overline{P}^\alpha\right)\right) \tag{5.6}$$

where $\underline{P}^\alpha = \underline{A}^\alpha p(\underline{x_i}^\alpha)$, $\overline{P}^\alpha = \overline{A}^\alpha p(\overline{x_i}^\alpha)$, $[x_i]^\alpha = (\underline{x_i}^\alpha, \overline{x_i}^\alpha)$.

Definition 5.3 (supremum metrics for Z-numbers) *[20]The supremum metrics in $\widehat{Z}^n$ is suggested as*

$$D(Z_1, Z_2) = d(A_1, A_2) + d(P_1, P_2) \tag{5.7}$$

in this case $d(\cdot, \cdot)$ is the supremum metrics considering fuzzy sets [131]. $(\widehat{Z}^n, D)$ is a complete metric space. $D(Z_1, Z_2)$ has been incorporated with the following possessions:

$$\begin{aligned}
&D(Z_1 + Z, Z_2 + Z) = D(Z_1, Z_2) \\
&D(Z_2, Z_1) = D(Z_1, Z_2) \\
&D(\lambda Z_1, \lambda Z_2) = |\lambda| D(Z_1, Z_2), \quad \lambda \in R \\
&D(Z_1, Z_2) \leq D(Z_1, Z) + D(Z, Z_2)
\end{aligned}$$

Similar with the fuzzy numbers [110], the Z-numbers are also incorporated with four primary operations: $\oplus$, $\ominus$, $\odot$ and $\oslash$. These operations are exhibited by: sum, subtract, multiply and division. The operations in this research work are different definitions with [222]. The α-level of Z-numbers is applied to simplify the operations.

Let us consider $Z_1 = (A_1, p_1)$ and $Z_2 = (A_2, p_2)$ to be two discrete Z-numbers illustrating the uncertain variables x_1 and x_2, $\sum_{k=1}^{n} p_1(x_{1k}) = 1$, $\sum_{k=1}^{n} p_2(x_{2k}) = 1$. The operations are defined

$$Z_{12} = Z_1 * Z_2 = (A_1 * A_2, p_1 * p_2)$$

where $* \in \{\oplus, \ominus, \odot, \oslash\}$.

The operations for the fuzzy numbers are defined as [110]

$$\begin{aligned}
&[A_1 \oplus A_2]^\alpha = [\underline{A_1}^\alpha + \underline{A_2}^\alpha, \overline{A_1}^\alpha + \overline{A_2}^\alpha] \\
&[A_1 \ominus A_2]^\alpha = [\underline{A_1}^\alpha - \underline{A_2}^\alpha, \overline{A_1}^\alpha - \overline{A_2}^\alpha] \\
&[A_1 \odot A_2]^\alpha = \left(\underline{A_1}^\alpha \underline{A_2}^\alpha + \underline{A_1}^\alpha \underline{A_2}^\alpha - \underline{A_1}^\alpha \underline{A_2}^\alpha, \overline{A_1}^\alpha \overline{A_2}^\alpha + \overline{A_1}^\alpha \overline{A_2}^\alpha - \overline{A_1}^\alpha \overline{A_2}^\alpha\right)
\end{aligned} \tag{5.8}$$

For all $p_1 * p_2$ operations, we use convolutions for the discrete probability distributions

$$p_1 * p_2 = \sum_i p_1(x_{1,i}) p_2(x_{2,(n-i)}) = p_{12}(x)$$

If A is a triangle function, the absolute value of the Z-number $Z = (A, p)$ is

$$|Z(x)| = (|a_1| + |b_1| + |c_1|, p(|a_2| + |b_2| + |c_2|)) \tag{5.9}$$

Also the above definitions satisfy the generalized Hukuhara difference [44]

$$Z_1 \ominus_{gH} Z_2 = Z_{12} \Longleftrightarrow \begin{cases} 1)\ Z_1 = Z_2 \oplus Z_{12} \\ 2)\ Z_2 = Z_1 \oplus (-1) Z_{12} \end{cases} \tag{5.10}$$

It is convenient to display that 1) and 2) in combination are genuine if and only if Z_{12} is a crisp number. With respect to α-level what we get is $[Z_1 \ominus_{gH} Z_2]^\alpha = [\min\{\underline{Z}_1^\alpha - \underline{Z}_2^\alpha, \overline{Z}_1^\alpha -$

$\overline{Z}_2^\alpha\}, \max\{\underline{Z}_1^\alpha - \underline{Z}_2^\alpha, \overline{Z}_1^\alpha - \overline{Z}_2^\alpha\}]$ and if $Z_1 \ominus_{gH} Z_2$ and $Z_1 \ominus_H Z_2$ subsist, $Z_1 \ominus_H Z_2 = Z_1 \ominus_{gH} Z_2$. The circumstances for the inerrancy of $Z_{12} = Z_1 \ominus_{gH} Z_2 \in E$ are

$$\begin{array}{l} 1) \left\{ \begin{array}{c} \underline{Z}_{12}^\alpha = \underline{Z}_1^\alpha - \underline{Z}_2^\alpha \text{ and } \overline{Z}_{12}^\alpha = \overline{Z}_1^\alpha - \overline{Z}_2^\alpha \\ \text{with } \underline{Z}_{12}^\alpha \text{ increasing, } \overline{Z}_{12}^\alpha \text{ decreasing, } \underline{Z}_{12}^\alpha \leq \overline{Z}_{12}^\alpha \end{array} \right. \\ 2) \left\{ \begin{array}{c} \underline{Z}_{12}^\alpha = \overline{Z}_1^\alpha - \overline{Z}_2^\alpha \text{ and } \overline{Z}_{12}^\alpha = \underline{Z}_1^\alpha - \underline{Z}_2^\alpha \\ \text{with } \underline{Z}_{12}^\alpha \text{ increasing, } \overline{Z}_{12}^\alpha \text{ decreasing, } \underline{Z}_{12}^\alpha \leq \overline{Z}_{12}^\alpha \end{array} \right. \end{array} \tag{5.11}$$

where $\forall \alpha \in [0, 1]$

Definition 5.4 ($\alpha-$level of Z-number valued function) *Let $\widetilde{Z}$ denotes the space of Z-numbers. The $\alpha-$level of Z-number valued function $F : [0, a] \to \widetilde{Z}$ is*

$$F(x, \alpha) = [\underline{F}(x, \alpha), \overline{F}(x, \alpha)]$$

where $x \in \widetilde{Z}$, for each $\alpha \in [0, 1]$.

With the definition of Generalized Hukuhara difference, the gH-derivative of F at x_0 is expressed as

$$\frac{d}{dt} F(x_0) = \lim_{h \to 0} \frac{1}{h} [F(x_0 + h) \ominus_{gH} F(x_0)] \tag{5.12}$$

In (5.12), $F(x_0 + h)$ and $F(x_0)$ exhibits similar style with Z_1 and Z_2 respectively included in (5.10).

If we apply the $\alpha-$level (5.5) to $f(t, x)$ in (5.45), then we obtain two Z-number valued functions: $\underline{f}\,[t, \underline{x}(\zeta, \alpha), \bar{x}(\zeta, \alpha)]$ and $\overline{f}\,[t, \underline{x}(\zeta, \alpha), \bar{x}(\zeta, \alpha)]$.

Now we use fuzzy equations (3.4) or (3.5) to model the uncertain nonlinear system (3.2). The parameters of the fuzzy equation (3.5) are in the form of Z-numbers,

$$y_k = a_1 \odot f_1(x_k) \oplus a_2 \odot f_2(x_k) \oplus \ldots \oplus a_n \odot f_n(x_k) \tag{5.13}$$

or

$$\begin{array}{l} a_1 \odot f_1(x_k) \oplus a_2 \odot f_2(x_k) \oplus \ldots \oplus a_n \odot f_n(x_k) \\ = b_1 \odot g_1(x_k) \oplus b_2 \odot g_2(x_k) \oplus \ldots \oplus b_m \odot g_m(x_k) \oplus y_k \end{array} \tag{5.14}$$

where a_i and b_i are Z-numbers.

Taking into consideration a particular case, $f_i(x_k)$ has polynomial pattern,

$$(a_1 \odot x_k) \oplus ... \oplus (a_n \odot x_k^n) = (b_1 \odot x_k) \oplus ... \oplus (b_n \odot x_k^n) \oplus y_k \tag{5.15}$$

(3.22) is termed as dual polynomial based on Z-number.

The object of the *modeling* is to minimize error between the two output y_k and z_k. As y_k is noted as a Z-number and z_k is considered to be crisp Z-number, hence we apply the minimum of every points as the modeling flaw

$$\begin{array}{l} \min_k |y_k - z_k| = \min_k |\beta_k| \\ y_k = \left((u_1(k), u_2(k), u_3(k)), p\left(v_1(k), v_2(k), v_3(k)\right)\right) \\ \beta_k = \left((\rho_1(k), \rho_2(k), \rho_3(k)), p\left(\varphi_1(k), \varphi_2(k), \varphi_3(k)\right)\right) \end{array} \tag{5.16}$$

By the definition of absolute value (5.9),

$$\begin{array}{l} \min_k |\beta_k| = \min_k[(|u_1(k) - f(x_k)| + |u_2(k) - f(x_k)| \\ +|u_3(k) - f(x_k)|), (|p(v_1(k)) - f(x_k)| + |p(v_2(k)) - f(x_k)| + |p(v_3(k)) - f(x_k)|)] \\ \rho_1(k) = \min_k |u_1(k) - f(x_k)|, \quad \rho_2(k) = \min_k |u_2(k) - f(x_k)|, \quad \rho_3(k) = \min_k |u_3(k) - f(x_k)| \\ p(\varphi_1(k)) = \min_k |p(v_1(k)) - f(x_k)|, \quad p(\varphi_2(k)) = \min_k |p(v_2(k)) - f(x_k)|, \\ p(\varphi_3(k)) = \min_k |p(v_3(k)) - f(x_k)| \end{array} \tag{5.17}$$

The modelling constraint (5.16) is to uncover $u_1(k)$, $u_2(k)$, $u_3(k)$, $p(v_1(k))$, $p(v_2(k))$ and $p(v_3(k))$ in such a manner

$$\min_{u_1(k),u_2(k),u_3(k),p(v_1(k)),p(v_2(k)),p(v_3(k))} \left\{\max_k |\beta_k|\right\} = \min_{u_1(k),u_2(k),u_3(k),p(v_1(k)),p(v_2(k)),p(v_3(k))} \left\{\max_k |y_k - f(x_k)|\right\} \tag{5.18}$$

Considering (5.17)

$$\begin{array}{l} \rho_1(k) \geq |u_1(k) - f(x_k)|, \quad \rho_2(k) \geq |u_2(k) - f(x_k)|, \quad \rho_3(k) \geq |u_3(k) - f(x_k)| \\ p(\varphi_1(k)) \geq |p(v_1(k)) - f(x_k)|, \quad p(\varphi_2(k)) \geq |p(v_2(k)) - f(x_k)|, \quad p(\varphi_3(k)) \geq |p(v_3(k)) - f(x_k)| \end{array}$$

(5.18) can be resolved by the application of linear programming methodology,

$$\begin{cases} \quad \min \rho_1(k) \\ \text{subject:} \begin{array}{l} \rho_1(k) + \{(\sum_{j=0}^{n} a_j \odot x_k^j) \ominus_{gH} (\sum_{j=0}^{n} b_j \odot x_k^j)\} \geq f(x_k) \\ \rho_1(k) - \{(\sum_{j=0}^{n} a_j \odot x_k^j) \ominus_{gH} (\sum_{j=0}^{n} b_j \odot x_k^j)\} \geq -f(x_k) \end{array} \\ \quad \min \varphi_1(k) \\ \text{subject:} \begin{array}{l} p(\varphi_1(k)) + \{(\sum_{j=0}^{n} a_j \odot x_k^j) \ominus_{gH} (\sum_{j=0}^{n} b_j \odot x_k^j)\} \geq f(x_k) \\ p(\varphi_1(k)) - \{(\sum_{j=0}^{n} a_j \odot x_k^j) \ominus_{gH} (\sum_{j=0}^{n} b_j \odot x_k^j)\} \geq -f(x_k) \end{array} \end{cases} \tag{5.19}$$

$$\begin{cases} \quad \min \rho_2(k) \\ \text{subject:} \begin{array}{l} \rho_2(k) - \left[\sum_{j=0}^{n} \underline{a}_j \underline{x}_k^j - \sum_{j=0}^{n} \underline{b}_j \underline{x}_k^j\right] \geq f(x_k) \\ \rho_2(k) \geq 0 \end{array} \\ \quad \min \varphi_2(k) \\ \text{subject:} \begin{array}{l} p(\varphi_2(k)) - \left[\sum_{j=0}^{n} \underline{a}_j \underline{x}_k^j - \sum_{j=0}^{n} \underline{b}_j \underline{x}_k^j\right] \geq f(x_k) \\ p(\varphi_2(k)) \geq 0 \end{array} \end{cases} \tag{5.20}$$

$$\begin{cases} \quad \min \rho_3(k) \\ \text{subject:} \begin{array}{l} \rho_3(k) - \left[\sum_{j=0}^{n} \bar{a}_j \bar{x}_k^j - \sum_{j=0}^{n} \bar{b}_j \bar{x}_k^j\right] \geq f(x_k) \\ \rho_3(k) \geq 0 \end{array} \\ \quad \min \varphi_3(k) \\ \text{subject:} \begin{array}{l} p(\varphi_3(k)) - \left[\sum_{j=0}^{n} \bar{a}_j \bar{x}_k^j - \sum_{j=0}^{n} \bar{b}_j \bar{x}_k^j\right] \geq f(x_k) \\ p(\varphi_3(k)) \geq 0 \end{array} \end{cases} \tag{5.21}$$

here $\underline{a}_j$, $\underline{b}_j$, $\underline{x}_k$, $\bar{a}_j$, $\bar{b}_j$ and $\bar{x}_k$ are explained as mentioned in (5.5). Henceforth, the superior way of approximating $f(x_k)$ at the juncture x_k is y_k. The lapse in approximating the error β_k is minimized.

The object of the *controller* design is to obtain u_k, such that the output of the plant y_k can follow a desired output y_k^*,

$$\min_{u_k} \|y_k - y_k^*\| \tag{5.22}$$

The control object can be regarded as: detect a solution u_k for the mentioned dual equation on the basis of Z-number

$$(a_1 \odot f_1(x_k)) \oplus (a_2 \odot f_2(x_k)) \oplus ... \oplus (a_n \odot f_n(x_k)) = (b_1 \odot g_1(x_k)) \oplus (b_2 \odot g_2(x_k)) \oplus ... \oplus (b_m \odot g_m(x_k)) \oplus y_k^* \tag{5.23}$$

here $x_k = [y_{k-1}^T, y_{k-2}^T, \cdots u_k^T, u_{k-1}^T, \cdots]^T$

5.2 Controllability of uncertain nonlinear systems via dual fuzzy equations and Z-numbers

As the main objective of control is the detection of a u_k of (5.14) based on Z-number, the controllability means the dual fuzzy equation (5.14) has solution.

We need the following lemmas.

Lemma 5.1 *If the coefficients of the dual equation (5.14) are Z-numbers, then the solution u_k satisfies*

$$\left\{ \cap_{j=1}^{n} domain\left[f_j\left(x\right)\right] \right\} \cap \left\{ \cap_{j=1}^{m} domain\left[g_j\left(x\right)\right] \right\} \neq \phi \tag{5.24}$$

Proof. Assume $u_0 \in \widehat{Z}$ is considered to be a solution of (5.14), the dual equation which relies on Z-numbers turns out to be

$$(a_1 \odot f_1(u_0)) \oplus ... \oplus (a_n \odot f_n(u_0)) = (b_1 \odot g_1(u_0)) \oplus ... \oplus (b_m \odot g_m(u_0)) \oplus y_k^*$$

As $f_j(u_0)$ and $g_j(u_0)$ prevail, $u_0 \in$domain$[f_j\left(x\right)]$, $u_0 \in$domain$[g_j\left(x\right)]$. Subsequently, it can be inferred that $u_0 \in \cap_{j=1}^{n}$domain$[f_j\left(x\right)] = D_1$, and $u_0 \in \cap_{j=1}^{m}$domain$[g_j\left(x\right)] = D_2$. Hence there prevails u_0, in such a manner $u_0 \in D_1 \cap D_2 \neq \phi$. ■

Let two Z-numbers $m_0, n_0 \in \widehat{Z}$, $m_0 < n_0$. We define a set $K\left(x\right) = \{x \in \widehat{Z}, m_0 \leq x \leq n_0\}$, and an operator $S : K \to K$ as

$$S\left(m_0\right) \geq m_0, \quad S\left(n_0\right) \leq n_0 \tag{5.25}$$

In this matter S is condensing and continuous, also it is bounded as $S(z) < r(z)$, $z \subset K$ and $r(z) > 0$. $r(z)$ can be considered as the evaluation of z.

Lemma 5.2 *We define $n_i = S(n_{i-1})$ and $m_i = S(m_{i-1})$, $i = 1, 2, ...$, and the upper and lower bounds of S are $\bar{s}$ and $\underline{s}$, then*

$$\bar{s} = \lim_{i \to +\infty} n_i, \quad \underline{s} = \lim_{i \to +\infty} m_i, \tag{5.26}$$

and

$$m_0 \le m_1 \le ... \le m_n \le ... \le n_n \le ... \le n_1 \le n_0. \tag{5.27}$$

Proof. As long as S is uprising, it is quite obvious from (5.25) that (5.27) prevails. In this case we verify that $\{m_i\}$ conjoins to some $\underline{s} \in \widehat{Z}$ and $S(\underline{s}) = \underline{s}$. The set $B = \{m_0, m_1, m_2, ...\}$ is enclosed and $B = S(B) \bigcup \{m_0\}$, thus, $r(B) = r(S(B))$ here $r(B)$ denotes the quantification of non-compactness of B. It is observed from S that $r(B) = 0$, i.e., B is a proportionally compact set. Thus, there prevails an outflow of $\{m_{n_k}\} \subset \{m_n\}$ in such a manner that $m_{n_k} \to \underline{s}$ for any $\underline{s} \in \widehat{Z}$ (take into consideration that $\widehat{Z}$ is complete). Distinctly, $m_n \le \underline{s} \le n_n$ $(n = 1, 2, ...)$. As in case $p > n_k$, according to the definition of supremum metrics for Z-numbers, it reveals that $D(\underline{s}, m_p) \le D(\underline{s}, m_{n_k})$. Hence, $m_p \to \underline{s}$ as $p \to \infty$. Considering limit $n \to \infty$ on either sides of the equality $m_n = S(m_{n-1})$, we find $\underline{s} = S(\underline{s})$, as a result S is continuous and K is closed.

Similarly, we can conclude that $\{n_n\}$ converges to some $\bar{s} \in \widehat{Z}$ and $S(\bar{s}) = \bar{s}$. So, we confirm that $\bar{s}$ and $\underline{s}$ are the maximal and minimal fixed point related to S in K, respectively. Assume $\widetilde{s} \in K$ and $S(\widetilde{s}) = \widetilde{s}$. As S is in the increase tend, it is obvious from $m_0 \le \widetilde{s} \le n_0$ that $S(m_0) \le S(\widetilde{s}) \le S(n_0)$, i.e., $m_1 \le \widetilde{s} \le n_1$. Utilizing the similar logic, we obtain $m_2 \le \widetilde{s} \le n_2$, and formally, $m_n \le \widetilde{s} \le n_n$ $(n = 1, 2, 3, ...)$. Here, considering limit $n \to \infty$, we extract $\underline{s} \le \widetilde{s} \le \bar{s}$.

The fixed point will result in x_0 inside K, the consecutive iterates $x_i = S(x_{i-1})$, $i = 1, 2, ...$ will result in convergency towards x_0, *i.e.*, the supremum matrix (5.7) $\lim_{i \to \infty} D(x_i, x_0) = 0$. ■

Theorem 5.1 *If a_i and b_j $(i = 1 \cdots n,\ j = 1 \cdots m)$ in (5.14) are Z-numbers, and they satisfy the Lipschitz condition*

$$\begin{array}{l} \left|(d_{M_1}(a_i), d_{M_2}(a_i)) - (d_{M_1}(a_k), d_{M_2}(a_k))\right| \le H\left|a_i(M_1) - a_k(M_1)\right| + H\left|a_i(M_2) - a_k(M_2)\right| \\ \left|(d_{U_1}(a_i), d_{U_2}(a_i)) - (d_{U_1}(a_k), d_{U_2}(a_k))\right| \le H\left|a_i(U_1) - a_k(U_1)\right| + H\left|a_i(U_2) - a_k(U_2)\right| \end{array} \tag{5.28}$$

the upper bounds of the functions f_i and g_j are $|f_i| \le \overline{f}$, $|g_j| \le \overline{g}$,

then the dual fuzzy equation (5.14) has a solution u in the following set

$$K_H = \left\{u \in \widetilde{Z}, \left|\overline{u}^{(\alpha_1,\beta_1)} - \underline{u}^{(\alpha_2,\beta_2)}\right| \le (n\overline{f} \oplus m\overline{g})(H\left|\alpha_1 - \alpha_2\right| + H\left|\beta_1 - \beta_2\right|)\right\} \tag{5.29}$$

Proof. Since a_i and b_j are the Z-numbers, and from the definition (5.28),

$$\begin{array}{c} d_M(\alpha,\beta) = ((a_{1M_1}(\alpha), a_{1M_2}(\beta)) \odot f_1(x)) \oplus ... \oplus ((a_{nM_1}(\alpha), a_{nM_2}(\beta)) \odot f_n(x)) \\ \ominus_{gH}((b_{1M_1}(\alpha), b_{1M_2}(\beta)) \odot g_1(x)) \ominus_{gH} ... \ominus_{gH} ((b_{mM_1}(\alpha), b_{mM_2}(\beta)) \odot g_m(x)) \end{array}$$

So

$$\begin{array}{c} \left|d_M(\alpha,\beta) - d_M(\varphi,\rho)\right| = (|f_1(x)| \odot \mid (a_{1M_1}(\alpha), a_{1M_2}(\beta)) \ominus_{gH} (a_{1M_1}(\varphi), a_{1M_2}(\rho)) \mid) \oplus \cdots \\ \oplus(|f_n(x)| \odot \mid (a_{nM_1}(\alpha), a_{nM_2}(\beta)) \ominus_{gH} (a_{nM_1}(\varphi), a_{nM_2}(\rho)) \mid) \\ \oplus(|g_1(x)| \odot \mid (b_{1M_1}(\alpha), b_{1M_2}(\beta)) \ominus_{gH} (b_{1M_1}(\varphi), b_{1M_2}(\rho)) \mid) \oplus \cdots \\ \oplus(|g_m(x)| \odot \mid (b_{mM_1}(\alpha), b_{mM_2}(\beta)) \ominus_{gH} (b_{mM_1}(\varphi), b_{mM_2}(\rho)) \mid) \end{array} \tag{5.30}$$

With respect to the Lipschitz condition (5.28), (5.30) is

$$\begin{array}{c} \left|d_M(\alpha,\beta) - d_M(\varphi,\rho)\right| \le \overline{f}(H\sum_{i=1}^n |\alpha - \varphi| + H\sum_{i=1}^n |\beta - \rho|) \oplus \overline{g}(H\sum_{i=1}^m |\alpha - \varphi| \\ +H\sum_{i=1}^m |\beta - \rho|) = \left(n\bar{f} \oplus m\bar{g}\right)(H\left|\alpha - \varphi\right| + H\left|\beta - \rho\right|) \end{array}$$

In the same manner, the upper limits suffice

$$\left|d_U(\alpha,\beta) - d_U(\varphi,\rho)\right| \le \left(n\bar{f} \oplus m\bar{g}\right)(H\left|\alpha - \varphi\right| + H\left|\beta - \rho\right|)$$

As the lower limit $\left|d_M(\alpha,\beta) - d_M(\varphi,\rho)\right| \ge 0$, with respect to Lemma 5.2 the solution contains in K_H and is defined as (5.29). ∎

Lemma 5.3 *Let us consider the data number to be m and also we suggest the order of the equation to be n in (5.15), also*

$$m \geq 2n+1 \tag{5.31}$$

where $k = 1 \cdots m$, hence the solutions of (5.20) and (5.21) are $\rho_2(k) = p(\varphi_2(k)) = \rho_3(k) = p(\varphi_3(k)) = 0$.

Proof. Since

$$\sum_{j=0}^{n} \underline{a}_j \underline{x}_k^j - \sum_{j=0}^{n} \underline{b}_j \underline{x}_k^j \leq -f(x_k), \quad i = 1, 2, ..., m. \tag{5.32}$$

Let us opt $2n+1$ points for x_k, and result is the interpolation of the dual polynomial based on Z-number

$$b\,(k) = \sum_{j=0}^{n} \underline{a}_j \underline{x}_k^j - \sum_{j=0}^{n} \underline{b}_j \underline{x}_k^j \tag{5.33}$$

Let $h = \max_k\{b\,(k) + f(x_k)\}$ and $h > 0$, as a result we can transform the dual polynomial based on Z-number (5.15) to the other form of new dual polynomial based on Z-number $b\,(k) - h$. This suggested recent dual polynomial based on Z-number suffices (5.32). Henceforth the presumable spot of (5.20) $\rho_2(k) \geq 0$ and $p(\varphi_2(k)) \geq 0$, it should be zero. In the similar manner, outcome can be extracted for (5.21). ■

The solutions x_k is Z-number. In case of $k = 1 \cdots n$, there should be a validated solution for the equation approximation [148]. Since $u_2\,(k)\,, v_2\,(k)\,, u_3\,(k)$ and $v_3\,(k)\,,$ (5.19) contains a solution.

Theorem 5.2 *If there are a big amount of data number (5.31), and the dual polynomial based on Z-number (5.15) satisfies*

$$D\left[h\left(x_{k1}, u_{k1}\right), h\left(x_{k2}, u_{k2}\right)\right] \leq lD\left[u_{k1}, u_{k2}\right] \quad 0 < l < 1 \tag{5.34}$$

where $h\,(\cdot)$ exhibits a dual polynomial based on Z-number,

$$h\left(x_{k1}, u_{k1}\right) : \left(a_1 \odot x_{k1}\right) \oplus ... \oplus \left(a_n \odot x_{k1}^n\right) = \left(b_1 \odot x_{k1}\right) \oplus ... \oplus \left(b_n \odot x_{k1}^n\right) \oplus y_{k1} \tag{5.35}$$

$D[u, v]$ *is the Hausdorff distance related to Z-numbers u and v,*

$$D[u,v] = \max\left\{ \sup_{(x_1,y_1)\in u} \inf_{(x_2,y_2)\in v} (d(x_1,x_2) + d(y_1,y_2)), \sup_{(x_1,y_1)\in v} \inf_{(x_2,y_2)\in u} (d(x_1,x_2) + d(y_1,y_2)) \right\}$$

$d(x, y)$ *is the supremum metrics considering fuzzy sets, then (5.15) contains a distinct solution u.*

Proof. The knowledge we extracted from lemma 5.2 states that, there exist solutions for (5.19)-(5.21), if it includes too much of data that satisfy (5.31). Neglecting deficit of generality, let we consider the solutions for (5.19)-(5.21) are at par with $x_k = 0$, which tends to u_0. (5.34) signifies $h(\cdot)$ in (5.35) is continuous. If we select a $\delta > 0$ in such a manner that $D[y_k, u_0] \leq \delta$, hence

$$D[h(x_k, u_0), u_0] \leq (1 - l)\delta$$

Considering $h(0, u_0) = u_0$. Taking into our account we choose x close to 0, $x_k \in [0, c]$, $c > 0$, and stated as

$$\mathcal{C}_0 : \rho = \sup_{x_k \in [0,c]} D[y_{k_1}, y_{k_2}]$$

Assume $\{y_{k_m}\}$ be a succession in $\mathcal{C}_0$, for any $\varepsilon > 0$, the computation can be done for $N_0(\varepsilon)$ in such a manner $\rho < \varepsilon$, $m, n \geq N_0$. Hence $y_{k_m} \longrightarrow y_k$ for $x_k \in [0, c]$. Henceforth

$$D[y_k, u_0] \leq D[y_k, y_{k_m}] + D[y_{k_m}, u_0] < \varepsilon + \delta \tag{5.36}$$

for all $x \in [0, c]$, $m \geq N_0(\varepsilon)$. As $\varepsilon > 0$ is randomly minute,

$$D[y_k, u_0] \leq \delta \tag{5.37}$$

for all $x \in [0, c]$. Now we validate that y_k is continuous at $x_0 = 0$. It is supplied $\delta > 0$, there resides $\delta_1 > 0$ in such a manner

$$D[y_k, u_0] \leq D[y_k, y_{k_m}] + D[y_{k_m}, u_0] \leq \varepsilon + \delta_1$$

for every $m \geq N_0(\varepsilon)$, by means of (5.37), while $|x - x_0| < \delta_1$, y_k is continuous at $x_0 = 0$. As a result (5.15) contains a distinct solution u_0. ■

The necessary circumstance in order to establish the controllability (existence of solution) related to the dual equation which is based on Z-number (5.23) is (5.24), the sufficient condition related to the controllability is (5.28). For majority of membership functions, such as triangular functions and the trapezoidal function, the Lipschitz condition (5.28) is contended. In this case it is considered to be controllable.

5.3 Fuzzy controller design with neural networks and dual fuzzy equations

It is not possible to acquire a solution based on analysis for (5.14). In this section, we utilize neural networks to approximate the solution (control). In order to fit the neural networks, (5.14) is written as

$$(a_1 \odot f_1(x)) \oplus ... \oplus (a_n \odot f_n(x)) \ominus_{gH} (b_1 \odot g_1(x)) \ominus_{gH} ... \ominus_{gH} (b_m \odot g_m(x)) = y_k^* \qquad (5.38)$$

We use two types of neural networks, feed-forward and feedback neural networks, to approximate the solutions of (5.38), see Figure 3.2 and Figure 3.3. The inputs to the neural network are the $Z-$numbers a_i and b_i, the outputs of the $Z-$number y_k. The weights are $f_i\left(x\right)$ and $g_j\left(x\right).$

The main idea is to detect appropriate weights of neural networks such that the output of the neural network $\hat{y}_k$, approaches the desired output y_k^*. In the control point of view, we want to find a controller u_k which is a function of x, such that the output of the plant (3.1) y_k (crisp value) approximate the$Z$-number y_k^*.

The input Z-numbers a_i and b_i are primarily implemented to α-level as (5.5)

$$\begin{array}{ll} [a_i]^\alpha = (\underline{a}^\alpha, \overline{a}^\alpha) & i = 1 \cdots n \\ [b_j]^\alpha = \left(\underline{b}^\alpha, \overline{b}^\alpha\right) & j = 1 \cdots m \end{array} \qquad (5.39)$$

After that, the operation involve multiplication by the Z-number weights $f_i\left(x\right)$ and $g_j\left(x\right)$

$$\begin{array}{l}
\left[O_f\right]^\alpha = \left(\sum_{i\epsilon M_f} \underline{f_i}^\alpha\left(x\right)\underline{a_i}^\alpha + \sum_{i\epsilon C_f} \underline{f_i}^\alpha\left(x\right)\overline{a_i}^\alpha, \sum_{i\epsilon M_f'} \overline{f_i}^\alpha\left(x\right)\overline{a_i}^\alpha, \sum_{i\epsilon C_f'} \overline{f_i}^\alpha\left(x\right)\underline{a_i}^\alpha\right) \\
\left[O_g\right]^\alpha = \left(\sum_{j\epsilon M_g} \underline{g_j}^\alpha\left(x\right)\underline{b_j}^\alpha + \sum_{j\epsilon C_g} \underline{g_j}^\alpha\left(x\right)\overline{b_j}^\alpha, \sum_{j\epsilon M_g'} \overline{g_j}^\alpha\left(x\right)\overline{b_j}^\alpha, \sum_{j\epsilon C_g'} \overline{g_j}^\alpha\left(x\right)\underline{b_j}^\alpha\right)
\end{array} \tag{5.40}$$

Here $M_f = \{i|\underline{f_i}^\alpha\left(x\right) \geq 0\}$, $C_f = \{i|\underline{f_i}^\alpha\left(x\right) < 0\}$, $M_f' = \{i|\overline{f_i}^\alpha\left(x\right) \geq 0\}$, $C_f' = \{i|\overline{f_i}^\alpha\left(x\right) < 0\}$, $M_g = \{j|\underline{g_j}^\alpha\left(x\right) \geq 0\}$, $C_g = \{j|\underline{g_j}^\alpha\left(x\right) < 0\}$, $M_g' = \{j|\overline{g_j}^\alpha\left(x\right) \geq 0\}$, $C_g' = \{j|\overline{g_j}^\alpha\left(x\right) < 0\}$.

The neural network output is

$$\left[\hat{y}_k\right]^\alpha = \left(\underline{O_f}^\alpha - \underline{O_g}^\alpha, \overline{O_f}^\alpha - \overline{O_g}^\alpha\right) \tag{5.41}$$

The training error is

$$e_k = y_k^* \ominus \hat{y}_k$$

here $\left[y_k^*\right]^\alpha = \left(\underline{y_k^*}^\alpha, \overline{y_k^*}^\alpha\right)$, $\left[\hat{y}_k\right]^\alpha = \left(\underline{\hat{y}_k}^\alpha, \overline{\hat{y}_k}^\alpha\right)$, $\left[e_k\right]^\alpha = \left(\underline{e_k}^\alpha, \overline{e_k}^\alpha\right)$.

In order to train the weights, we need a cost function related to the Z-numbers as

$$J_k = \underline{J}^\alpha + \overline{J}^\alpha, \quad \underline{J}^\alpha = \frac{1}{2}\left(\underline{y_k^*}^\alpha - \underline{\hat{y}_k}^\alpha\right)^2, \quad \overline{J}^\alpha = \frac{1}{2}\left(\overline{y_k^*}^\alpha - \overline{\hat{y}_k}^\alpha\right)^2 \tag{5.42}$$

It is quite obvious, $J_k \rightarrow 0$ when $\left[\hat{y}_k\right]^\alpha \rightarrow \left[y_k^*\right]^\alpha$.

The index (5.42) is least mean square. It has a self-correcting feature that makes it suitable to vast arbitrarily without shifting from its constraints. The gradient algorithm is subjected to cumulative series of errors.

The gradient technique is now been utilized to train the Z-number weights $f_i\left(x\right)$ and $g_j\left(x\right)$. The solution x_0 is the function of $f_i\left(x\right)$ and $g_j\left(x\right)$. We compute $\frac{\partial J_k}{\partial \underline{x_0}}$ and $\frac{\partial J_k}{\partial \overline{x_0}}$ as

$$\begin{array}{c}
\frac{\partial J_k}{\partial \underline{x_0}} = \frac{\partial \underline{J}^\alpha}{\partial \underline{x_0}} + \frac{\partial \overline{J}^\alpha}{\partial \underline{x_0}} \\
\frac{\partial J_k}{\partial \overline{x_0}} = \frac{\partial \underline{J}^\alpha}{\partial \overline{x_0}} + \frac{\partial \overline{J}^\alpha}{\partial \overline{x_0}}
\end{array} \tag{5.43}$$

According to the chain rule

$$\frac{\partial \underline{J}^\alpha}{\partial \underline{x_0}} = \frac{\partial \underline{J}^\alpha}{\partial \underline{\hat{y}_k}^\alpha}\frac{\partial \underline{\hat{y}_k}^\alpha}{\partial \underline{O_f}^\alpha}\frac{\partial \underline{O_f}^\alpha}{\partial \underline{f_i}^\alpha\left(x\right)}\frac{\partial \underline{f_i}^\alpha\left(x\right)}{\partial \underline{x_0}} - \frac{\partial \underline{J}^\alpha}{\partial \underline{\hat{y}_k}^\alpha}\frac{\partial \underline{\hat{y}_k}^\alpha}{\partial \underline{O_g}^\alpha}\frac{\partial \underline{O_g}^\alpha}{\partial \underline{g_j}^\alpha\left(x\right)}\frac{\partial \underline{g_j}^\alpha\left(x\right)}{\partial \underline{x_0}}$$

So

$$\frac{\partial J^{\alpha}}{\partial \underline{x_0}} = \sum_{i=1}^{n} -\left(\underline{y_k^{*\alpha}} - \underline{\hat{y}_k}^{\alpha}\right) \underline{a}_i^{\alpha} \underline{f_i}'^{\alpha} + \sum_{j=1}^{m} \left(\underline{y_k^{*\alpha}} - \underline{\hat{y}_k}^{\alpha}\right) \underline{b}_j^{\alpha} \underline{g_j}'^{\alpha}$$

Or

$$\frac{\partial J^{\alpha}}{\partial \underline{x_0}} = \sum_{i=1}^{n} -\left(\underline{y_k^{*\alpha}} - \underline{\hat{y}_k}^{\alpha}\right) \overline{a}_i^{\alpha} \underline{f_i}'^{\alpha} + \sum_{j=1}^{m} \left(\underline{y_k^{*\alpha}} - \underline{\hat{y}_k}^{\alpha}\right) \overline{b}_j^{\alpha} \underline{g_j}'^{\alpha}$$

$\frac{\partial J_k}{\partial \overline{x_0}}$ can be calculated the same as above.

The solution x_0 is upgraded as

$$\begin{aligned} \underline{x_0}(k+1) &= \underline{x_0}(k) - \eta \tfrac{\partial J_k}{\partial \underline{x_0}} \\ \overline{x_0}(k+1) &= \overline{x_0}(k) - \eta \tfrac{\partial J_k}{\partial \overline{x_0}} \end{aligned}$$

Here η is the rate of the training $\eta > 0$.

For the requirement of increasing the training methodology, the adding of the momentum term is mentioned as

$$\begin{aligned} \underline{x_0}(k+1) &= \underline{x_0}(k) - \eta \tfrac{\partial J_k}{\partial \underline{x_0}} + \gamma \left[\underline{x_0}(k) - \underline{x_0}(k-1)\right] \\ \overline{x_0}(k+1) &= \overline{x_0}(k) - \eta \tfrac{\partial J_k}{\partial \overline{x_0}} + \gamma \left[\overline{x_0}(k) - \overline{x_0}(k-1)\right] \end{aligned}$$

Here $\gamma > 0$. After the updating of x_0 , it is necessary to induce the weights $f_i(x_0)$ and $g_j(x_0)$.

The solution related to the dual equation is on the basis of Z-number (5.14) which can also be estimated by feedback neural network, as in Figure 3.3. In this form, the inputs are the Z-number functions of nonlinearity $f_i(x)$ and $g_j(x)$, the concerned weights are taken to be as Z-numbers a_i and b_j. The training error e_k has been utilized here in order to update x. Once the nonlinear operations $f_i(x)$ and $g_j(x)$ are performed, O_f and O_g are considered to be similar to (5.40). The output related to the neural network is similar to (5.41).

5.4 Nonlinear system modeling with fuzzy differential equations Z-numbers

Consider the following controlled unknown nonlinear system

$$\dot{x} = f_1(x_1, u, t) \tag{5.44}$$

where $f_1(x_1, u)$ is unknown vector function, $x_1 \in \Re^n$ is an internal state vector and $u \in \Re^m$ is the input vector.

Here, we use the following differential equation (FDE) to model the uncertain nonlinear system (5.44),

$$\frac{d}{dt}x = f(x, u) \tag{5.45}$$

where $x \in \Re^n$ is the Z-number variable, which corresponds to the state x_1 in (5.44), $f(t, x)$ is a Z-number vector function, which relates to $f_1(x_1, u)$, $\frac{d}{dt}x$ is the derivative associated to the Z-number variable. Here the uncertainties of the nonlinear system (5.44) are in the sense of Z-numbers.

The fuzzy differential equation (5.45) can be equivalent to the following four ODE

$$\begin{array}{l} 1) \left\{ \begin{array}{l} \frac{d}{dt}\underline{x} = \underline{f}\left[t, \underline{x}(\zeta, \alpha), \bar{x}(\zeta, \alpha)\right] \\ \frac{d}{dt}\bar{x} = \overline{f}\left[t, \underline{x}(\zeta, \alpha), \bar{x}(\zeta, \alpha)\right] \end{array} \right. \\ 2) \left\{ \begin{array}{l} \frac{d}{dt}\underline{x} = \overline{f}\left[t, \underline{x}(\zeta, \alpha), \bar{x}(\zeta, \alpha)\right] \\ \frac{d}{dt}\bar{x} = \underline{f}\left[t, \underline{x}(\zeta, \alpha), \bar{x}(\zeta, \alpha)\right] \end{array} \right. \end{array} \tag{5.46}$$

Here, we use the FDE (5.45) to model the uncertain nonlinear system (5.44), such that the output of the plant x can follow the plant output x_1,

$$\min_f \|x - x_1\| \tag{5.47}$$

This modeling object can be considered as: finding $\overline{f}$ and $\underline{f}$ in the fuzzy equations of (5.46) or finding the solutions of these models. It is impossible to obtain analytical solutions. We use neural networks to approximate them, see Figure 5.1.

In fact, the nonlinear system can be modeled by the neural network directly. However, this data-driven black box identification method does not use the model information.

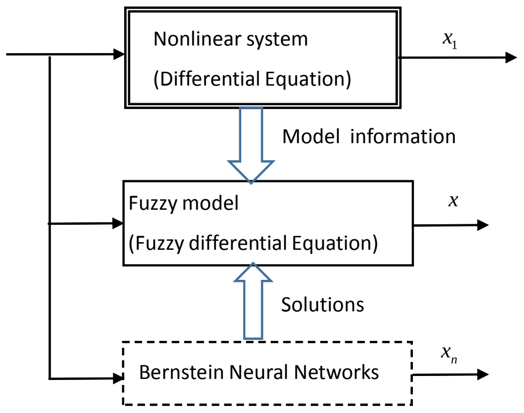

Figure 5.1: Nonlinear system modeling with fuzzy differential equation

5.5 Controllability of uncertain nonlinear systems via fuzzy differential equations and Z-numbers

Theorem 5.3 *If the Z-number function f and its derivative $\frac{\partial f}{\partial x}$ are on the rectangle $[-p,p]\times[-q,q]$, here $p,q\in\widetilde{Z}$, $\widetilde{Z}$ is the space of Z-numbers, then there exists an unique Z-number solution for the following FDE based on Z-numbers*

$$\frac{d}{dt}x = f(t,x), \quad x(t_0)=x_0 \tag{5.48}$$

for all $t\in(-b,b)$, $b\leq p$

Proof. We utilize Picard's iteration technique [46] to develop a sequence of Z-number functions $\varphi_n(t)$ as

$$\begin{aligned}\varphi_{n+1}(t) &= \varphi_0 \oplus \textstyle\int_0^t f(s,\varphi_n(s))ds \\ &= \varphi_0 \ominus_H (-1) \textstyle\int_0^t f(s,\varphi_n(s))ds\end{aligned}$$

We first validate that $\varphi_n(t)$ is continuous and prevail for all n. Obviously, if $\varphi_n(t)$ prevail then $\varphi_{n+1}(t)$ is also prevail as

$$\begin{aligned}\varphi_{n+1}(t) &= \varphi_0 \oplus \textstyle\int_0^t f(s,\varphi_n(s))ds \\ &= \varphi_0 \ominus_H (-1) \textstyle\int_0^t f(s,\varphi_n(s))ds\end{aligned}$$

Since f is continuous, so there exists $N \in E$ such that $|f(t,x)| \leq N$ for all $t \in [-p,p]$, as well as all $x \in [-q,q]$. If we set $t \in [-b,b]$ for $b \leq \min(q/N, p)$, then it is possible

$$\|\varphi_{n+1} \ominus \varphi_0\| = \| \int_0^t f(s, \varphi_n(s))ds\| \leq N|t| \leq Nb \leq q$$

This validates that $\varphi_{n+1}(t)$ acquires values in $[-q,q]$. Because

$$\varphi_n(t) = \sum_{k=1}^{n} (\varphi_n(t) \ominus \varphi_{n-1}(t))$$

for any $\gamma < 1$, we select $t \in (-b,b)$ such that $\mid \varphi_k(t) \ominus \varphi_{k-1}(t) \mid \leq \gamma^k$ for all k. This signifies that there exists $\gamma < 1$ [118]

$$\mid \varphi_k(t) \ominus \varphi_{k-1}(t) \mid \leq \gamma^k$$

From the mean value theorem [180],

$$\varphi_k(t) \ominus \varphi_{k-1}(t) = \int_0^t [f(s, \varphi_{k-1}(s)) \ominus f(s, \varphi_{k-2}(s))]ds$$

Applying the mean value theorem into the Z-number function $h(x) = f(s,x)$ in the two points $\varphi_{k-1}(s)$ and $\varphi_{k-2}(s)$,

$$h(\varphi_{k-1}(s)) \ominus h(\varphi_{k-2}(s)) = h'(\psi_k(s))(\varphi_{k-1}(s)) \ominus \varphi_{k-2}(s))$$

Taking into consideration $h'(x) = \frac{\partial f}{\partial x}$, we obtain

$$\varphi_k(t) \ominus \varphi_{k-1}(t) = \int_0^t \frac{\partial f}{\partial x}(s, \psi_k(s))(\varphi_{k-1}(s) \ominus \varphi_{k-2}(s))ds \tag{5.49}$$

Because $\mid \varphi_{k-1}(s) \ominus \varphi_{k-2}(s) \mid \leq \gamma^{k-1}$ for $s \leq t$ and $b < \gamma/N$, by substituting the above relation in (5.49) and utilizing the boundess of $\frac{\partial f}{\partial x}$,

$$\mid \varphi_k(t)) \ominus \varphi_{k-1}(t) \mid \leq \int_0^t N\gamma^{k-1}ds = Nt\gamma^{k-1} \leq Nb\gamma^{k-1}$$

In order to validate that x is continuous, it is necessary to show that for any given $\epsilon > 0$ there exists $\delta > 0$ in such a manner that $\mid t_2 - t_1 \mid < \delta$ implies $\mid \varphi(t_2) \ominus \varphi(t_1) \mid < \epsilon$. At par with the notation convenience, we suppose that $t_1 < t_2$. It follows that

$$\varphi(t_2) \ominus \varphi(t_1) = \lim_{n\to\infty} \varphi_n(t_2) \ominus \lim_{n\to\infty} \varphi_n(t_1)$$
$$= \lim_{n\to\infty}(\varphi_n(t_2) \ominus \varphi_n(t_1)) = \lim_{n\to\infty} \int_{t_1}^{t_2} f(s, \varphi_n(s))ds$$

There exists N in such a manner that $\mid f(s,x) \mid \leq N$. Hence

$$\mid \varphi(t_2) \ominus \varphi(t_1) \mid \leq \int_{t_1}^{t_2} N ds = N \mid t_2 - t_1 \mid \leq N\delta$$

henceforth by selecting $\delta < \epsilon/N$ it is observed that $\mid \varphi(t_2) \ominus \varphi(t_1) \mid < \epsilon$. So $\lim_{n\to\infty} \varphi_n(t)$ is prevail for all t.

Now we demonstrate that $\lim_{n\to\infty} \varphi_n(t)$ is continuous. Since

$$\begin{array}{c} \varphi(t) = \lim_{n\to\infty} \varphi_n(t) = \lim_{n\to\infty} \int_0^t f(s, \varphi_{n-1}(s))ds \\ = \int_0^t \lim_{n\to\infty} f(s, \varphi_{n-1}(s))ds = \int_0^t f(s, \lim_{n\to\infty} \varphi_{n-1}(s))ds \end{array}$$

where the last step (moving the limit inside the function) is at par with the concept that f is continuous in each variable. Hence it is clear that

$$\varphi(t) = \int_0^t f(s, \varphi(s))ds$$

because all functions are continuous,

$$\frac{d}{dt}\varphi = f(s, \varphi(t))$$

If there exists another solution $\phi(t)$,

$$\varphi(t) \ominus \phi(t) = \int_0^t (f(s, \varphi(t)) \ominus f(s, \phi(t)))ds$$

Since the two functions are different, there exists $\epsilon > 0$, $\mid \varphi(t) \ominus \phi(t) \mid > \epsilon$. We define

$$m = \max_{0 \leq t \leq b} \mid \varphi(t) \ominus \phi(t) \mid$$

N is the bound for $\frac{\partial f}{\partial x}$. Utilizing the mean value theorem,

$$\mid \varphi(t) \ominus \phi(t) \mid \leq \int_0^t N \mid \varphi(t) - \phi(t) \mid ds \leq N \mid t \mid m \leq Nbm$$

If we select $b < \epsilon/2mN$, it signifies that for all $t < b$, $\mid \varphi(t) - \phi(t) \mid < \epsilon/2$, that contracts the fact that the least difference is ϵ. So there exists an unique Z-number solution. ■

Theorem 5.4 *Assume the following FDE based on Z-numbers*

$$\frac{d}{dt}x = f(t,x) \tag{5.50}$$

here $f \in \bar{J}_{ab}$, $\bar{J}_{ab}$ is the set of linear strongly bounded operators, for every operators f there exists a function $\tau \in L([a,b];\widetilde{Z}_+)$ such that $|f(\nu)(t)| \leq \tau(t)\|\nu\|_G$, $t \in [a,b]$ and $\nu \in G([a,b];\widetilde{Z})$ and there prevails $f_0, f_1 \in \varphi_{ab}$, φ_{ab} is a set of linear operators $f \in \bar{J}_{ab}$ from the set $G([a,b];\widetilde{Z}_+)$ to the set $L([a,b];\widetilde{Z}_+)$, such that

$$\begin{array}{ll} |\underline{f}(t,\underline{\nu},\overline{\nu}) + \underline{f}_1(t,\underline{\nu},\overline{\nu})| \leq \underline{f}_0(t,|\underline{\nu}|,|\overline{\nu}|), & t \in [a,b] \\ |\overline{f}(t,\underline{\nu},\overline{\nu}) + \overline{f}_1(t,\underline{\nu},\overline{\nu})| \leq \overline{f}_0(t,|\underline{\nu}|,|\overline{\nu}|), & t \in [a,b] \end{array} \tag{5.51}$$

then (5.50) has an unique solution.

Proof. If x is a Z-number solution of (5.50) and $-\frac{1}{2}f_1 \in J_{ab}(a)$,

$$\frac{d}{dt}\beta = -\frac{1}{2}f_1(t,\beta) \oplus f_0(t,|x|) \oplus \frac{1}{2}f_1(t,|x|) \tag{5.52}$$

contains a unique Z-number solution β. Moreover as $f_0, f_1 \in \varphi_{ab}$

$$\begin{array}{ll} \underline{\beta}(t) \geq 0, & t \in [a,b] \\ \overline{\beta}(t) \geq 0, & t \in [a,b] \end{array} \tag{5.53}$$

According to (5.51) and the condition $f_1 \in \varphi_{ab}$, from (5.52) we have

$$\begin{array}{l} \frac{d}{dt}\underline{\beta} \geq -\frac{1}{2}\underline{f}_1(t,\underline{\beta},\overline{\beta}) + \underline{f}(t,\underline{x},\overline{x}) + \frac{1}{2}\underline{f}_1(t,\underline{x},\overline{x}) \\ \frac{d}{dt}\overline{\beta} \geq -\frac{1}{2}\overline{f}_1(t,\underline{\beta},\overline{\beta}) + \overline{f}(t,\underline{x},\overline{x}) + \frac{1}{2}\overline{f}_1(t,\underline{x},\overline{x}) \end{array}$$

thus $t \in [a,b]$

$$\begin{array}{l} \frac{d}{dt}(-\underline{\beta}) \leq -\frac{1}{2}\underline{f}_1(t,-\underline{\beta},-\overline{\beta}) + \underline{k}(t,\underline{x},\overline{x}) + \frac{1}{2}\underline{k}_1(t,\underline{x},\overline{x}) \\ \frac{d}{dt}(-\overline{\beta}) \leq -\frac{1}{2}\overline{f}_1(t,-\underline{\beta},-\overline{\beta}) + \overline{f}(t,\underline{x},\overline{x}) + \frac{1}{2}\overline{f}_1(t,\underline{x},\overline{x}) \end{array}$$

The last two inequalities is on account of the presumption $-\frac{1}{2}f_1 \in J_{ab}(a)$

$$\begin{array}{ll} |\underline{x}(t)| \leq \underline{\beta}(t) & t \in [a,b] \\ |\overline{x}(t)| \leq \overline{\beta}(t) & t \in [a,b] \end{array} \tag{5.54}$$

According to the (5.54) and the conditions $f_0, f_1 \in \varphi_{ab}$, (5.52) results in

$$\begin{array}{ll} \frac{d}{dt}\underline{\beta} \leq \underline{f}_0(t, \underline{\beta}, \overline{\beta}), & t \in [a,b] \\ \frac{d}{dt}\overline{\beta} \leq f_0(t, \underline{\beta}, \overline{\beta}), & t \in [a,b] \end{array}$$

As $f_0 \in J_{ab}(a)$, the last inequality with $\beta(a) = 0$ yields $\underline{\beta}(t) \leq 0$ and $\overline{\beta}(t) \leq 0$ for $t \in [a,b]$. (5.53) implies $\beta \equiv 0$. Thus based on (5.54) we have $x \equiv 0$. ■

5.6 Fuzzy controller design with neural networks and fuzzy differential equations

In general, it is difficult to solve the four equations (4.5) or (4.4). Here, we use a special neural network named Bernstein neural network to approximate the solutions of the FDE (4.4).

The Bernstein neural network uses the following Bernstein polynomial,

$$\begin{array}{c} B(x_1, x_2) = \sum_{i=0}^{N} \sum_{j=0}^{M} \binom{N}{i} \binom{M}{j} \\ W_{i,j} x_{1i} (T - x_{1i})^{N-i} x_{2j} (1 - x_{2j})^{M-j} \end{array} \tag{5.55}$$

where $\binom{N}{i} = \frac{N!}{i!(N-i)!}$, $\binom{M}{j} = \frac{M!}{j!(M-j)!}$, $W_{i,j}$ is the Z-number coefficient.

This two variable polynomial can be regarded as a neural network, which has two inputs x_{1i} and x_{2j} and one output y,

$$y = \sum_{i=0}^{N} \sum_{j=0}^{M} \lambda_i \gamma_j W_{i,j} x_{1i} (T - x_{1i})^{N-i} x_{2j} (1 - x_{2j})^{M-j} \tag{5.56}$$

where $\lambda_i = \binom{N}{i}$, $\gamma_j = \binom{M}{j}$.

Because the Bernstein neural network (5.56) has similar forms as (4.5), we use the Bernstein neural network (5.56) to approximate the solutions of four ODEs in (4.5).

If x_1 and x_2 in (5.55) are defined as: x_1 is time interval t, x_2 is the α-level , the solution of (4.4) in the form of the Bernstein neural network is

$$\begin{array}{c} x_m(t, \alpha) = x_m(0, \alpha) \\ \oplus t \sum_{i=0}^{N} \sum_{j=0}^{M} \lambda_i \gamma_j W_{i,j} t_i (T - t_i)^{N-i} \alpha_j (1 - \alpha_j)^{M-j} \end{array} \tag{5.57}$$

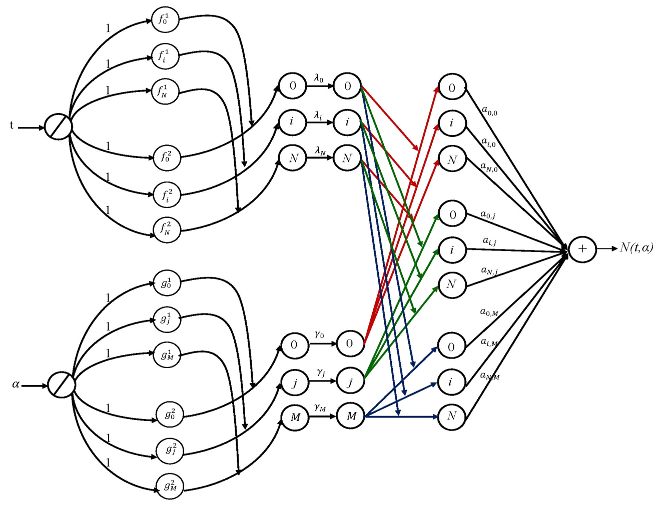

Figure 5.2: Static Bernstein neural network

where $x_m(0,\alpha)$ is the initial condition of the solution based on Z-number.

so the derivative of (5.56) is

$$\begin{array}{l} 1)\left\{\begin{array}{l} \frac{d}{dt}\underline{x}_m = C_1 + C_2 \\ \frac{d}{dt}\bar{x}_m = D_1 + D_2 \end{array}\right. \\ 2)\left\{\begin{array}{l} \frac{d}{dt}\underline{x}_m = C_1 + C_2 \\ \frac{d}{dt}\bar{x}_m = D_1 + D_2 \end{array}\right. \end{array} \tag{5.58}$$

where $t \in [0,T]$, $\alpha \in [0,1]$, $t_k = kh$, $h = \frac{T}{k}$, $k = 1,...,N$, $\alpha_j = jh_1$, $h_1 = \frac{1}{M}$, $j = 1,...,M$,

$$\begin{array}{l} C_1 = \sum_{i=0}^{N}\sum_{j=0}^{M}\lambda_i\gamma_j\underline{W}_{i,j}t_i(T-t_i)^{N-i}\alpha_j(1-\alpha_j)^{M-j} \\ D_1 = \sum_{i=0}^{N}\sum_{j=0}^{M}\lambda_i\gamma_j\overline{W}_{i,j}t_i(T-t_i)^{N-i}\alpha_j(1-\alpha_j)^{M-j} \\ C_2 = t_k\sum_{i=0}^{N}\sum_{j=0}^{M}\lambda_i\gamma_j\underline{W}_{i,j}[it_{i-1,j}(T-t_i)^{N-i} \\ -(N-i)\,t_{i,j}(T-t_i)^{N-i-1}]\alpha_j^i(1-\alpha_j)^{M-j} \\ D_2 = t_k\sum_{i=0}^{N}\sum_{j=0}^{M}\lambda_i\gamma_j\overline{W}_{i,j}[it_{i-1,j}(T-t_i)^{N-i} \\ -(N-i)\,t_{i,j}(T-t_i)^{N-i-1}]\alpha_j^i(1-\alpha_j)^{M-j} \end{array}$$

The above equations can be regarded as the neural network form, see Figure 5.2.

- input unit:

$$o_1^1 = t, \quad o_2^1 = \alpha$$

- the first hidden units:

$$o_{1,i}^2 = f_i^1(o_1^1), \quad o_{2,i}^2 = f_i^2(o_1^1)$$
$$o_{3,j}^2 = g_j^1(o_2^1), \quad o_{4,j}^2 = g_j^2(o_2^1)$$

- the second hidden units:

$$o_{1,i}^3 = o_{1,i}^2(o_{2,i}^2), \quad o_{2,j}^3 = o_{3,j}^2(o_{4,j}^2)$$

- the third hidden units:

$$o_{1,i}^4 = \lambda_i o_{1,i}^3, \quad o_{2,i'}^4 = \gamma_j o_{2,j}^3$$

- the forth hidden units:

$$o_{i,j}^5 = o_{1,i}^4 o_{2,j}^4$$

- output unit:

$$N(t,\alpha) = \sum_{i=0}^{N}\sum_{j=0}^{M}(a_{i,j}o_{i,j}^5)$$

where $f_i^1 = t^i$, $f_i^2 = (T-t)^{N-i}$, $\lambda_i = \frac{1}{T^N}\binom{N}{i}$, $g_j^1 = \alpha^j$, $g_j^2 = (1-\alpha)^{M-j}$, $\gamma_j = \binom{M}{j}$.
We define the training errors between (5.58) and (4.5) as

$$\begin{array}{l} 1)\begin{cases} \underline{e}_1 = C_1 + C_2 - \underline{f} \\ \bar{e}_1 = D_1 + D_2 - \bar{f} \end{cases} \\ 2)\begin{cases} \underline{e}_2 = C_1 + C_2 - \bar{f} \\ \bar{e}_2 = D_1 + D_2 - \underline{f} \end{cases} \end{array} \tag{5.59}$$

The standard back-propagation learning algorithm is utilized to update the weights with the above training errors

$$\begin{array}{l} \underline{W}_{i,j}(k+1) = \underline{W}_{i,j}(k) - \eta_1(\frac{\partial \underline{e}_1^2}{\partial \underline{W}_{i,j}} + \frac{\partial \bar{e}_1^2}{\partial \underline{W}_{i,j}}) \\ \overline{W}_{i,j}(k+1) = \overline{W}_{i,j}(k) - \eta_2(\frac{\partial \underline{e}_2^2}{\partial \overline{W}_{i,j}} + \frac{\partial \bar{e}_2^2}{\partial \overline{W}_{i,j}}) \end{array} \tag{5.60}$$

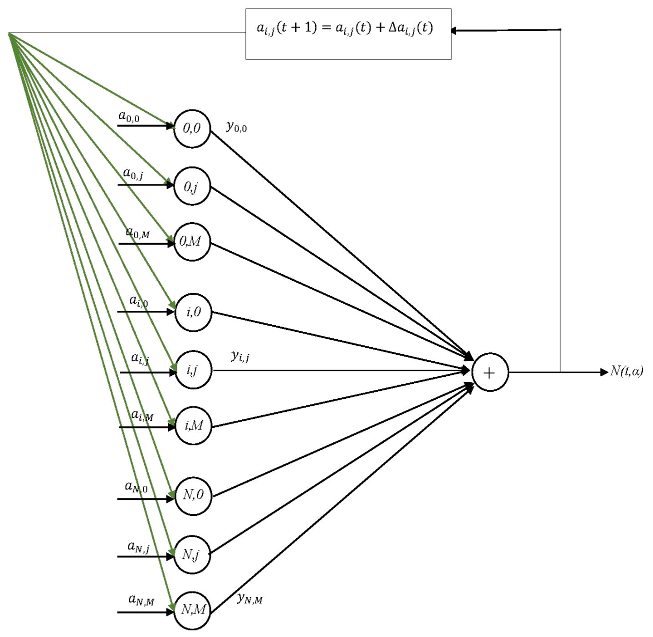

Figure 5.3: Dynamic Bernstein nerual network

where η_1 and η_2 are positive learning rates.

The momentum terms, $\gamma\Delta\underline{W}_{i,j}(k-1)$ and $\gamma\Delta\bar{W}_{i,j}(k-1)$ can be added to stabilize the training process. The above Bernstein neural network can be retended into a recurrent (dynamic) form, see Figure 5.3. The dynamic Bernstein neural network is

$$\left\{ \begin{array}{l} \frac{d}{dt}\underline{x}_m(t,\alpha) = \underline{P}(t,\alpha)A(\underline{x}_m(t,\alpha), \bar{x}_m(t,\alpha)) + \underline{Q}(t,\alpha) \\ \frac{d}{dt}\bar{x}_m(t,\alpha) = \overline{P}(t,\alpha)A(\underline{x}_m(t,\alpha), \bar{x}_m(t,\alpha)) + \overline{Q}(t,\alpha) \end{array} \right. \tag{5.61}$$

Obviously this dynamic network has the form of

$$f(t,x) = P(t)x + Q(t)$$

The training algorithm is similar as (5.60), only the training errors are changed as

$$\begin{array}{l} 1)\left\{\begin{array}{l} \underline{e}_1 = C_1 + C_2 - \underline{P}A(\underline{x}_m, \bar{x}_m) - \underline{Q} \\ \bar{e}_1 = D_1 + D_2 - \overline{P}A(\underline{x}_m, \bar{x}_m) - \overline{Q} \end{array}\right. \\ 2)\left\{\begin{array}{l} \underline{e}_2 = C_1 + C_2 - \overline{P}A(\underline{x}_m, \bar{x}_m) - \overline{Q} \\ \bar{e}_2 = D_1 + D_2 - \underline{P}A(\underline{x}_m, \bar{x}_m) - \underline{Q} \end{array}\right. \end{array} \tag{5.62}$$

5.7 Simulations

In this section, we use several real applications to show how to use the fuzzy equation and FDE with Z-number coefficients to design the fuzzy controller.

Example 5.1 (Chemical reaction) *The chemical reaction is to use the poly ethylene (PE) and poly propylene (PP) to generate a desired substance (DS). If the cost of the material is defined as x, the cost of PE is x and the cost of PP is x^2. The weights of PE and PP are uncertain, which satisfy the triangle function (5.1). We want to product two types DS. If we wish the cost between them are $[(360.5009, 400.5565, 421.3749), p(0.8, 0.9, 1)] = y^*$, what is the cost x ? The weights of PE are*

$$a_1 = [(2.7951, 3.35412, 3.9131), p(0.7, 0.8, 1)]$$
$$b_1 = [(1.5811, 2.1081, 2.6352), p(0.8, 0.9, 1)]$$

The PP weights are

$$a_2 = [(4.8107, 5.3452, 5.8797), p(0.7, 0.875, 1)]$$
$$b_2 = [(3.9131, 4.4721, 5.0311), p(0.6, 0.8, 1)]$$

The reaction can be modeled with the following fuzzy equation and Z-numbers

$$\begin{array}{c} [(2.7951, 3.35412, 3.9131), p(0.7, 0.8, 1)]x \oplus [(4.8107, 5.3452, 5.8797), p(0.7, 0.875, 1)]x^2 \\ = [(1.5811, 2.1081, 2.6352), p(0.8, 0.9, 1)]x \oplus [(3.9131, 4.4721, 5.0311), p(0.6, 0.8, 1)]x^2 \\ \oplus[(360.5009, 400.5565, 421.3749), p(0.8, 0.9, 1)] \end{array}$$

Here $f_1(x) = g_1(x) = x$, $f_2(x) = g_2(x) = x^2$.

The exact solution is $x^* = [(18.3712, 19.3919, 19.9022), p(0.8, 0.96, 1)]$. *We use feed-forward neural networks (NN) as Figure 3.2 and feedback neural networks (FNN) as Figure 3.3 to approximate the solution* x. *The learning rate is* $\eta = 0.02$. *The initial state is* $x(0) = [(22.6612, 23.7102, 24.2407), p(0.8, 0.9, 1)]$. *The approximation results are shown in Table 5.1. The modeling errors are shown in Figure 5.4.*

Table 5.1. Neural networks approximate the $Z-$ *numbers*

k	$x(k)$ *with NN*	k	$x(k)$ *with FNN*
1	$[(22.531, 23.684, 24.103), p(0.6, 0.8, 0.85)]$	1	$[(22.323, 23.487, 23.979), p(0.7, 0.8, 0.85)]$
2	$[(21.793, 22.837, 23.203), p(0.7, 0.8, 0.85)]$	2	$[(20.982, 22.133, 22.761), p(0.7, 0.85, 0.9)]$
$\vdots$	$\vdots$	$\vdots$	$\vdots$
35	$[(18.674, 19.717, 20.237), p(0.8, 0.92, 1)]$	18	$[(18.492, 19.514, 20.135), p(0.8, 0.92, 1)]$
36	$[(18.382, 19.401, 19.911), p(0.8, 0.96, 1)]$	19	$[(18.379, 19.397, 19.907), p(0.8, 0.96, 1)]$

We can see that both neural networks work well. We use the follows to transfer the $Z-$*numbers to fuzzy numbers,*

$$\alpha = \frac{\int x\pi_{\widetilde{P}}(x)dx}{\int \pi_{\widetilde{P}}(x)dx}$$

$Z = (\widetilde{A}, \widetilde{P}) = [(22.331, 23.384, 23.993), p(0.6, 0.8, 0.85)]$. *So* $\widetilde{Z}^{\alpha} = (22.331, 23.384, 23.993; 0.7)$, $\widetilde{Z}' = (\sqrt{0.7}\ 22.331, \sqrt{0.7}\ 23.384, \sqrt{0.7}\ 23.993)$. *The results of neural networks approximation for the fuzzy numbers are shown in Table 5.2.*

Table 5.2. Neural networks approximate the fuzzy numbers

k	$x(k)$ *with NN*	k	$x(k)$ *with FNN*
1	$(19.358, 20.349, 20.709)$	1	$(19.672, 20.698, 21.132)$
2	$(19.205, 20.125, 20.448)$	2	$(18.844, 19.878, 20.442)$
$\vdots$	$\vdots$	$\vdots$	$\vdots$
35	$(17.720, 18.710, 19.203)$	18	$(17.548, 18.517, 19.107)$
36	$(17.541, 18.513, 19.000)$	19	$(17.538, 18.509, 18.996)$

The $Z-$*numbers increase degree of reliability of the information. The comparison between the* Z*-number* $Z = [(18.382, 19.401, 19.911), p(0.8, 0.96, 1)]$ *and fuzzy number* $(17.541, 18.513, 19.000)$

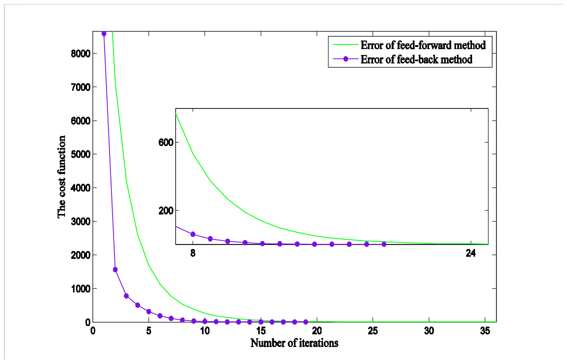

Figure 5.4: Approximation errors of the neural networks

for $k = 36$ is shown in Figure 5.5. We see that the Z-number incorporates with various information and the solution of the Z-number is more accurate. The membership function for the restriction in the Z-number is $\mu_{A_Z} = (18.382, 19.401, 19.911)$. It can be in probability form.

Example 5.2 (Heat source) *Heat source is in the center of the insulating materials, see Figure 3.5. The thickness of the materials are not exact, which satisfy the trapezoidal function (5.2),*

$$A = [(0.1317, 0.1536, 0.1646, 0.1975), p(0.7, 0.83, 0.9)] = a_1$$
$$B = [(0.0843, 0.1054, 0.2108, 0.5270), p(0.8, 0.9, 1)] = a_2$$
$$C = [(0.0964, 0.1072, 0.2144, 0.4288), p(0.7, 0.87, 0.9)] = b_1$$
$$D = [(0.0216, 0.0325, 0.0542, 0.0867), p(0.8, 0.85, 0.92)] = b_2$$

The coefficient associated with conductivity materials are $K_A = x = f_1$, $K_B = x\sqrt{x} = f_2$, $K_C = x^2 = g_1$, $K_D = \sqrt{x} = g_2$, where x is considered to be as the elapsed time. The control object is to find the time when the thermal resistance at the right side arrives $R = [(0.0162, 0.0293, 0.0424, 0.1241), p(0.75, 0.8, 0.9)] = y^*$. The thermal balance model is [100]:

$$\frac{A}{K_A} \oplus \frac{B}{K_B} = \frac{C}{K_C} \oplus \frac{D}{K_D} \oplus R$$

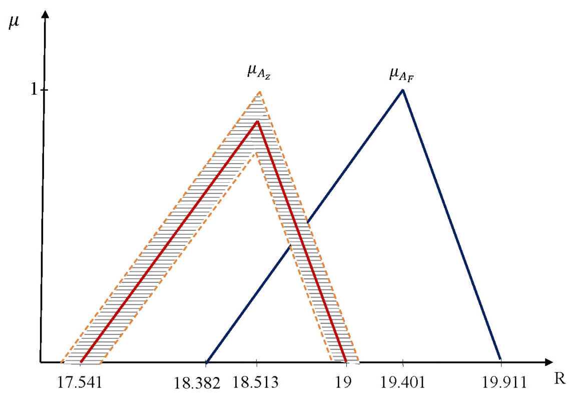

Figure 5.5: Z-number and fuzzy number

The exact solution is $x^* = [(2.0519, 3.0779, 4.1039, 6.1559), p(0.8, 0.95, 1)]$ *[100]. The learning rate is* $\eta = 0.1$ *(NN) and* $\eta = 0.005$ *(FNN). The neural networks approximation results are shown in Table 5.3 and Table 5.4.*

Table 5.3. Neural networks approximate the $Z-$ *numbers*

k	$x(k)$ *with NN*	k	$x(k)$ *with FNN*
1	$[(5.972, 6.983, 7.963, 9.982), p(0.6, 0.8, 0.85)]$	1	$[(5.989, 6.990, 7.979, 9.988), p(0.7, 0.85, 0.87)]$
2	$[(5.438, 6.383, 7.353, 9.302), p(0.75, 0.8, 0.9)]$	2	$[(5.378, 6.102, 7.123, 9.162), p(0.7, 0.85, 0.87)]$
$\vdots$	$\vdots$	$\vdots$	$\vdots$
53	$[(2.118, 3.170, 4.224, 6.333), p(0.8, 0.9, 1)]$	22	$[(2.089, 3.149, 4.146, 6.292), p(0.8, 0.96, 1)]$
54	$[(2.069, 3.087, 4.113, 6.175), p(0.8, 0.94, 1)]$	23	$[(2.059, 3.084, 4.113, 6.169), p(0.8, 0.94, 1)]$

Table 5.4. Neural networks approximate the fuzzy numbers

k	$x(k)$ *with NN*	k	$x(k)$ *with FNN*
1	$(5.131, 5.999, 6.841, 8.576)$	1	$(5.360, 6.255, 7.141, 8.939)$
2	$(4.934, 5.791, 6.671, 8.440)$	2	$(4.813, 5.461, 6.374, 8.199)$
$\vdots$	$\vdots$	$\vdots$	$\vdots$
53	$(2.009, 3.007, 4.007, 6.008)$	22	$(1.993, 3.004, 3.956, 6.004)$
54	$(1.966, 2.934, 3.915, 5.870)$	23	$(1.957, 2.931, 3.909, 5.864)$

Example 5.3 (Water channel system) *The water in the pipe d_1 is divided into three pipes d_2, d_3, d_4, see Figure 3.10. The areas of the pipes are uncertain, they satisfy the trapezoidal function (5.2),*

$$A_1 = [(0.421, 0.632, 0.737, 0.843), p(0.75, 0.9, 1)]$$
$$A_2 = [(0.052, 0.104, 0.209, 0.419), p(0.8, 0.91, 1)]$$
$$A_3 = [(0.031, 0.084, 0.105, 0.210), p(0.8, 0.9, 0.95)]$$

The water velocities in the pipes are controlled by the valves parameter x, $v_1 = x^3$, $v_2 = \frac{e^x}{2}$, $v_3 = x$ [196]. The control object is to let the flow in pipe d_4 which is

$$Q = [(11.478, 40.890, 93.332, 293.056), p(0.8, 0.87, 0.95)]$$

what is the valve control parameter x. By mass balance

$$A_1 v_1 = A_2 v_2 \oplus A_3 v_3 \oplus Q$$

The exact solution is demonstrated by $x = [(3.127, 4.170, 5.212, 7.298), p(0.8, 0.92, 1)]$ [196]. The learning rate of NN is $\eta = 0.08$. The neural networks approximation results are shown in Table 5.5 and Table 5.6.

Table 5.5. Neural networks approximate the $Z-$ numbers

k	$x(k)$ *with NN*	k	$x(k)$ *with FNN*
1	$[(5.751, 6.772, 7.741, 9.761), p(0.6, 0.8, 0.85)]$	1	$[(5.878, 6.881, 7.867, 9.877), p(0.7, 0.81, 0.85)]$
2	$[(5.327, 6.261, 7.131, 9.201), p(0.7, 0.8, 0.87)]$	2	$[(5.158, 6.004, 7.001, 9.002), p(0.7, 0.85, 0.9)]$
$\vdots$	$\vdots$	$\vdots$	$\vdots$
55	$[(3.142, 4.194, 5.229, 7.321), p(0.8, 0.9, 1)]$	20	$[(3.136, 4.185, 5.225, 7.312), p(0.85, 0.9, 1)]$
56	$[(3.136, 4.188, 5.222, 7.315), p(0.8, 0.93, 1)]$	21	$[(3.130, 4.178, 5.218, 7.305), p(0.8, 0.92, 1)]$

Table 5.6. Neural networks approximate the fuzzy numbers

k	$x\,(k)$ *with NN*	k	$x\,(k)$ *with FNN*
1	$(4.941, 5.818, 6.651, 8.386)$	1	$(5.188, 6.073, 6.944, 8.718)$
2	$(4.720, 5.548, 6.319, 8.153)$	2	$(4.632, 5.392, 6.287, 8.085)$
$\vdots$	$\vdots$	$\vdots$	$\vdots$
55	$(2.980, 3.978, 4.960, 6.945)$	20	$(2.975, 3.970, 4.956, 6.936)$
56	$(2.977, 3.976, 4.958, 6.945)$	21	$(2.970, 3.964, 4.951, 6.932)$

FNN is much faster and more robust compared with NN. After converting the $Z-$numbers to fuzzy numbers, it is possible to extract the fuzzy rules.

Example 5.4 *The heat treatment system in welding can be modeled as [61]:*

$$\frac{d}{dt}x(t) = 3Ax^2(t) \tag{5.63}$$

where transfer area A is uncertain as $A = [(1+\alpha, 3-\alpha), p(0.8, 0.87, 0.95)]$, $\alpha \in [0,1]$. So (5.63) is a FDE based on Z-number. If the initial condition is $x(0) = [(0.5\sqrt{\alpha}, 0.2\sqrt{1-\alpha}+0.5), p(0.8, 0.92, 1)]$, the static Bernstein neural network (4.9) has the form of

$$\begin{cases} \underline{x}_m(t,\alpha) = 0.5\sqrt{\alpha} \\ +t\sum_{i=0}^{N}\sum_{j=0}^{M}\lambda_i\gamma_j\underline{W}_{i,j}t_i(T-t_i)^{N-i}\alpha_j(1-\alpha_j)^{M-j} \\ \overline{x}_m(t,\alpha) = 0.2\sqrt{1-\alpha}+0.5 \\ +t\sum_{i=0}^{N}\sum_{j=0}^{M}\lambda_i\gamma_j\overline{W}_{i,j}t_i(T-t_i)^{N-i}\alpha_j(1-\alpha_j)^{M-j} \end{cases}$$

where the approximate Z-number solution is termed as $[(\underline{x}_m(t,\alpha), \overline{x}_m(t,\alpha)), p(0.8, 0.9, 1)]$. With the learning rates $\eta = 0.002$ and $\gamma = 0.002$, the approximation results for Z-numbers are shown in Table 4. The results of Bernstein neural networks approximation for the fuzzy numbers are shown in Table 5.

Table 4. Bernstein neural networks approximate the Z-numbers

α	SNN	DNN
0	[(0.0582,0.0859),p(0.7,0.8,0.85)]	[(0.0250,0.0425),p(0.7,0.82,0.9)]
0.1	[(0.0449,0.0696),p(0.7,0.8,0.9)]	[(0.0224,0.0399),p(0.75,0.82,0.9)]
0.2	[(0.0419,0.0619),p(0.8,0.92,1)]	[(0.0207,0.0394),p(0.8,0.94,1)]
0.3	[(0.0250,0.0348),p(0.7,0.81,0.9)]	[(0.0226,0.0344),p(0.8,0.85,0.96)]
0.4	[(0.0487,0.0689),p(0.7,0.8,0.88)]	[(0.0271,0.0510),p(0.75,0.82,0.9)]
0.5	[(0.0534,0.0665),p(0.8,0.9,1)]	[(0.0160,0.0271),p(0.81,0.92,1)]
0.6	[(0.0494,0.0765),p(0.8,0.9,1)]	[(0.0201,0.0413),p(0.81,0.92,1)]
0.7	[(0.0630,0.0859),p(0.75,0.82,0.9)]	[(0.0303,0.0476),p(0.82,0.9,1)]
0.8	[(0.0393,0.0536),p(0.8,0.92,1)]	[(0.0164,0.0379),p(0.82,0.94,1)]
0.9	[(0.0422,0.0669),p(0.8,0.9,1)]	[(0.0212,0.0430),p(0.8,0.94,1)]
1	[(0.0443,0.0443),p(0.7,0.8,0.88)]	[(0.0186,0.0186),p(0.7,0.82,0.9)]

Table 5.Bernstein neural networks approximate the fuzzy numbers

α	SNN	DNN
0	[0.0511,0.0754]	[0.0224,0.0381]
0.1	[0.0402,0.0623]	[0.0203,0.0362]
0.2	[0.0398,0.0588]	[0.0197,0.0374]
0.3	[0.0224,0.0312]	[0.0211,0.0321]
0.4	[0.0433,0.0613]	[0.0246,0.0462]
0.5	[0.0507,0.0631]	[0.0152,0.0258]
0.6	[0.0469,0.0726]	[0.0191,0.0392]
0.7	[0.0571,0.0778]	[0.0288,0.0452]
0.8	[0.0373,0.0509]	[0.0157,0.0362]
0.9	[0.0401,0.0635]	[0.0202,0.0408]
1	[0.0394,0.0394]	[0.0167,0.0167]

Example 5.5 *A generalized model of a tank system is displayed in Figure 5.6. Assume* $I = t + 1$ *be inflow disturbances of the tank that will generate vibration in liquid level* x*, here* $R = 1$ *will be the flow obstruction that can be curbed using the valve and* $A = 1$ *is considered to be cross section of the mentioned tank. The expression in relation to the liquid*

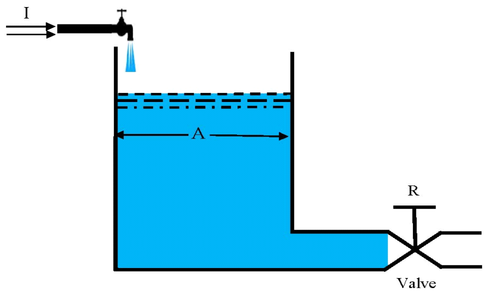

Figure 5.6: Liquid tank system

level considering the tank can be described as [196]:

$$\frac{d}{dt}x(t) = -\frac{1}{AR}x(t) + \frac{I}{A} \tag{5.64}$$

If the initial condition is $x(0) = [(0.96 + 0.04\alpha, 1.01 - 0.01\alpha), p(0.75, 0.82, 0.9)]$, *the static Bernstein neural network (4.9) has the form of*

$$\begin{cases} \underline{x}_m(t,\alpha) = (0.96 + 0.04\alpha) \\ +t\sum_{i=0}^{N}\sum_{j=0}^{M}\lambda_i\gamma_j\underline{W}_{i,j}t_i(T-t_i)^{N-i}\alpha_j(1-\alpha_j)^{M-j} \\ \overline{x}_m(t,\alpha) = (1.01 - 0.01\alpha) \\ +t\sum_{i=0}^{N}\sum_{j=0}^{M}\lambda_i\gamma_j\overline{W}_{i,j}t_i(T-t_i)^{N-i}\alpha_j(1-\alpha_j)^{M-j} \end{cases}$$

where $t \in [0,1]$ *and the approximate* Z*-number solution is termed as* $[(\underline{x}_m(t,\alpha), \overline{x}_m(t,\alpha)), p(0.75, 0.81, 0.95)]$. *We also use dynamic Bernstein neural network (4.13) to approximate the solutions. To compare our results, we use the other generalization of neural network method [77]. The comparison results for* Z*-numbers are shown in Table 6. The specifications quoted here are* $\eta = 0.001$ *and* $\gamma = 0.001$. *Corresponding error plots are shown in Figure 5.7. The results of Bernstein neural networks approximation for the fuzzy numbers are shown in Table 7.*

Table 6. Solutions of different methods based on Z*-numbers*

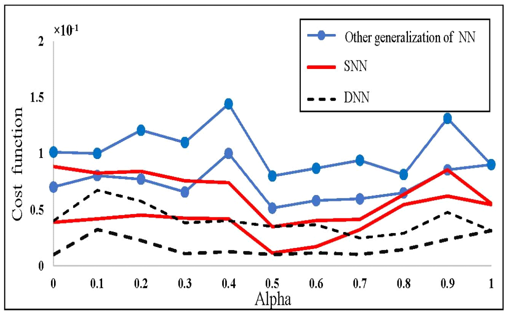

Figure 5.7: Plot of absolute error between the exact solution and the approximated one with SNN, DNN and other generalization of neural network methodology based on Z-numbers

α	*SNN*	*DNN*	*Neural network*
0	*[(0.0435, 0.0994),p(0.72,0.81,0.87)]*	*[(0.0112, 0.0442),p(0.75,0.82,0.88)]*	*[(0.0798, 0.1153),p(0.7,0.75,0.85)]*
0.2	*[(0.0504, 0.0940),p(0.7,0.8,0.9)]*	*[(0.0248, 0.0635),p(0.75,0.82,0.9)]*	*[(0.0878, 0.1375),p(0.7,0.8,0.85)]*
0.4	*[(0.0441, 0.0802),p(0.8,0.85,0.92)]*	*[(0.0131, 0.0422),p(0.8,0.9,1)]*	*[(0.1105, 0.1592),p(0.75,0.83,0.9)]*
0.6	*[(0.0178, 0.0423),p(0.8,0.92,1)]*	*[(0.0121, 0.0384),p(0.81,0.94,1)]*	*[(0.0613, 0.0915),p(0.8,0.9,1)]*
0.8	*[(0.0608, 0.0709),p(0.71,0.8,0.9)]*	*[(0.0154, 0.0309),p(0.8,0.87,0.95)]*	*[(0.0739, 0.0925),p(0.7,0.8,0.85)]*
1	*[(0.0611, 0.0611),p(0.75,0.82,0.91)]*	*[(0.0335, 0.0335),p(0.8,0.87,0.92)]*	*[(0.1007, 0.1007),p(0.7,0.8,0.9)]*

Table 7. Solutions of different methods based on fuzzy numbers

α	*SNN*	*DNN*	*Neural network*
0	*[0.0387, 0.0884]*	*[0.0101, 0.0398]*	*[0.0701, 0.1012]*
0.2	*[0.0451, 0.0841]*	*[0.0225, 0.0575]*	*[0.0771, 0.1207]*
0.4	*[0.0407, 0.0740]*	*[0.0125, 0.0401]*	*[0.1001, 0.1442]*
0.6	*[0.0169, 0.0402]*	*[0.0115, 0.0365]*	*[0.0582, 0.0868]*
0.8	*[0.0544, 0.0635]*	*[0.0144, 0.0289]*	*[0.0649, 0.0812]*
1	*[0.0554, 0.0554]*	*[0.0311, 0.0311]*	*[0.0901, 0.0901]*

5.8 Conclusions

We use dual fuzzy equations, whose coefficients are Z-numbers, to model uncertain nonlinear systems. Then we give the relation between the solution of the fuzzy equations and the controllers. The controllability of the fuzzy system is proposed. Two types of neural networks are applied to approximate the solutions of the fuzzy equations. Modified gradient descent algorithms are used to train the neural networks. The novel methods are validated by several benchmark examples.

Chapter 6

Conclusions

Many uncertain nonlinear systems can be modeled by linear-in-parameter models. The uncertainties can be regarded as parameter changes, which can be described as fuzzy numbers. These models are fuzzy equations or FDEs. The nonlinear system modeling corresponds to find the fuzzy parameters of the fuzzy equations or FDEs, and the fuzzy control is to design suitable nonlinear functions in the fuzzy equation or FDE. Both fuzzy modeling and fuzzy control via fuzzy equations and FDEs need solution of these equations. There are various approaches. However, all of analytical methods for the solutions of fuzzy equations or FDEs are very difficult to be applied, especially for nonlinear fuzzy equations or FDEs. Moreover, the numerical methods are very complex and the approximation accuracy of the numerical calculations are normally less.

Here, the modeling and controlling uncertainty nonlinear systems with fuzzy equations is proposed. We discuss more general fuzzy equations: dual fuzzy equations. The controllability condition is given for the fuzzy control through these equations. Two types of neural networks are applied to approximate the solutions of the mentioned equations. They are controller design process. For modeling, we first transform the fuzzy equation into a neural network. Then we modify the normal gradient descent method to train the fuzzy coefficients. With this modification, we can apply normal neural modeling methods to uncertain nonlinear system modeling with fuzzy equations. The approximation theory for crisp models are extended into fuzzy cases. The upper bounds of the modeling errors with fuzzy equations are estimated. In continue the solutions of the FDEs are approximated by two types of Bernstein neural networks. These numerical methods use generalized differentiability of FDEs. The solutions of FDE is substituted into four ODEs. Then the corresponding Bernstein neural networks are applied. Also a methodology involving novel iterative technique considering neural networks is suggested to extract approximate solution for the second-order nonlinear PDEs with real constant coefficients (RCCs) taking into account initial and boundary conditions. This perspective is designed to grant good approximation on the basis of learning technique which is associated with quasi-Newton rule. A trial solution of this system is subdivided into two parts. The initial and boundary conditions compensate the first part that contains no adjustable parameters. The involvement of neural network containing adjustable parameters (weights and biases) concludes the second part. Also, a sophisticated methodology is provided in order to solve PDEs on the basis of the application of Bernstein polynomial that is modeled with the help of two pattern of neural networks, static and dynamic models. Furthermore to continue, we use dual fuzzy equations, whose coefficients are Z-numbers, to model uncertain nonlinear systems. The relation between the solution of the fuzzy equations and the controllers is given. Two types of neural networks are implemented to approximate solutions of the fuzzy equations with Z-number coefficients. Also, the solutions of FDEs based on Z-numbers are approximated by two types of Bernstein neural networks.

Bibliography

[1] L.P. Aarts, P. Van der veer, Neural network method for solving partial differential equations, *Neural Processing Letters,* Vol.14, pp.261-271, 2001.

[2] K. Abbaoui, Y. Cherruault, Convergence of Adomian's method applied to nonlinear equations, *Math. Comput. Model.* Vol.20, pp.69-73, 1994.

[3] S. Abbasbandy, Improving Newton–Raphson method for nonlinear equations by modified Adomian decomposition method, *Appl. Math. Comput.* Vol.145, pp.887-893, 2003.

[4] S. Abbasbandy, The application of homotopy analysis method to nonlinear equations arising in heat transfer, *Phys. Lett. A.* Vol.360, pp.109-113, 2006.

[5] S. Abbasbandy, T. Allahvinloo, Numerical solutions of fuzzy differential equations by Taylor method, *J. Comput. Meth. Appl. Math.* Vol. 2, pp.113-124, 2002.

[6] S. Abbasbandy, T. Allahvinloo, Numerical solutions of fuzzy differential equation, *Meth. Comput. Appl.* Vol.7, pp.41-52, 2002.

[7] S. Abbasbandy, T. Allahvinloo, Numerical solution of fuzzy differential equation by Runge–Kutta method, *Nonlinear Studies,* Vol.11, pp.117-129, 2004.

[8] S. Abbasbandy, B. Asady, Newton's method for solving fuzzy nonlinear equations, *Appl. Math. Comput.* Vol. 159, pp.349-356, 2004.

[9] S. Abbasbandy, R. Ezzati, Newton's method for solving a system of fuzzy nonlinear equations, *Appl. Math. Comput.* Vol.175, pp.1189-1199, 2006.

[10] S. Abbasbandy, A. Jafarian, Steepest descent method for solving fuzzy nonlinear equations, *Appl. Math. Comput.* Vol.174, pp.669-675, 2006.

[11] S. Abbasbandy, M. Otadi, Numerical solution of fuzzy polynomials by fuzzy neural network, *Appl. Math. Comput.* Vol.181, pp.1084-1089, 2006.

[12] G. Adomian, R. Rach, On the solution of algebraic equations by the decomposition method, *Journal of Mathematical Analysis and Applications,* Vol.105, pp.141-166, 1985.

[13] M. Afshar Kermani, F. Saburi, Numerical Method for Fuzzy Partial Differential Equations, *Appl. Math. Sci.* Vol.1, pp.1299-1309, 2007.

[14] R.P. Agarwal, V. Lakshmikantham, J.J. Nieto, On the concept of solution for fractional differential equations with uncertainty, *Nonlinear Anal.,* Vol.72, pp.2859-2862, 2010.

[15] S. Agatonovic-Kustrin, R. Beresford, Basic concepts of artificial neural network (ANN) modeling and its application in pharmaceutical research, *Journal of Pharmaceutical and Biomedical Analysis.* vol.22, pp.717-727, 2000.

[16] M.Z. Ahmadi, M.K. Hasan, A new fuzzy version of Euler's method for solving differential equations with fuzzy initial values, *Sains Malaysiana,* Vol.40, pp.651-657, 2011.

[17] M.Z. Ahmadi, M.K. Hasan, Incorporating optimisation technique into Euler method for solving differential equations with fuzzy initial values, *Proceeding of the 1st Regional Conference on Applied and Engineering Mathematics,* Malaysia, 2010.

[18] M.B. Ahmadi, N.A. Kiani, N. Mikaeilvand, Laplace transform formula on fuzzy nth-order derivative and its application in fuzzy ordinary differential equations, *Soft Comput.* Vol. 18, pp.2461-2469, 2014.

[19] R.A. Aliev, A.V. Alizadeh, O.H. Huseynov, The arithmetic of discrete Z-numbers, *Inform. Sci.* Vol.290, pp.134-155, 2015.

[20] R.A. Aliev, O.H. Huseynov, Decision theory with imperfect information, *World Scientific,* 2014.

[21] R.A. Alieva, W. Pedryczb, V. Kreinovich, O.H. Huseynov, The general theory of decisions, *Inform. Sci.* Vol.327, pp.125-148, 2016.

[22] T. Allahviranloo, Difference methods for fuzzy partial differential equations, *Computational Methods in Applied Mathematics,* Vol.2, pp.233-242, 2002.

[23] T. Allahviranloo, N. Ahmadi, E. Ahmadi, Numerical solution of fuzzy differential equations by predictor-corrector method, *Inform. Sci.* Vol.177, pp.1633-1647, 2007.

[24] T. Allahviranloo, N. Ahmadi, E. Ahmadi, A method for solving n-th order fuzzy linear differential equations, *Comput. Math. Appl.* Vol.86, pp.730-742, 2009.

[25] T. Allahviranloo, S. Asari, Numerical solution of fuzzy polynomials by Newton–Raphson method, *J. Appl. Math.* Vol.27, pp.17-24, 2011.

[26] T. Allahviranloo, L. Gerami Moazam, The solution of fully fuzzy quadratic equation based on optimization theory, *The Scientific World Journal* Vol.2014, Article ID 156203, 6 pages.

[27] T. Allahviranloo, N.A, Kiani, M. Barkhordari, Toward the exiistence and uniqueness of solution of second-order fuzzy differential equations, *Inform. Sci.* Vol.179, pp.1207-1215, 2009.

[28] T. Allahviranloo, N.A, Kiani, N. Motamedi, Solving fuzzy differential equations by differential transformation method, *Inform. Sci.* Vol.179, pp.956-966, 2009.

[29] T. Allahviranloo, M. Otadi, M. Mosleh, Iterative method for fuzzy equations, *Soft Computing,* Vol.12, pp.935-939, 2007.

[30] H. Alli, A. Ucar, Y. Demir, The solutions of vibration control problems using artificial neural networks, *J. Franklin Inst.* Vol.340, pp.307-325, 2003.

[31] M. Amirfakhrian, Numerical solution of algebraic fuzzy equations with crisp variable by Gauss–Newton method, *Appl. Math. Model.* Vol.32, pp.1859-1868, 2008.

[32] S. Arshad, V. Lupulescu, On the fractional differential equations with uncertainty, *Nonlinear Anal.* Vol.74, pp.3685-3693, 2011.

[33] F. Ayaz, On the two-dimensional differential transform method, *Appl. Math. Comput.* Vol.143, pp.361-374, 2003.

[34] E. Babolian, J. Biazar, Solution of nonlinear equations by modified Adomian decomposition method, *Appl. Math. Comput.* Vol.132, pp.167-172, 2002.

[35] E. Babolian, J. Biazar, On the order of convergence of Adomian method, *Appl. Math. Comput.* Vol.130, pp.383-387, 2002.

[36] E. Babolian, Sh. Javadi, Restarted Adomian method for algebraic equations, *Appl. Math. Comput.* Vol.146, pp.533–541, 2003.

[37] A. Ban, B. Bede, Properties of the cross product of fuzzy numbers, *Journal of Fuzzy Mathematics,* Vol.14, pp.513-531, 2006.

[38] M. Barkhordari Ahmadi, N.A. Kiani, Solving fuzzy partial differential equation by differential transformation method, *Journal of Appl. Math.* Vol.7, pp.1-16, 2011.

[39] V. Barthelmann, E. Novak, K. Ritter, High dimensional polynomial interpolation on sparse grids, *Adv Comput Math.* Vol.12, pp.273-88, 2000.

[40] B. Bede, Note on numerical solutions of fuzzy differential equations by predictor corrector method, *Inform. Sci.* Vol.178, pp.1917-1922, 2008.

[41] B. Bede, S.G. Gal, Generalizations of differentiablity of fuzzy number valued function with application to fuzzy differential equations, *Fuzzy Sets Syst.* Vol.151, pp.581–599, 2005.

[42] B. Bede, J. Imre, C. Rudas, L. Attila, First order linear fuzzy differential equations under generalized differentiability, *Inform. Sci.* Vol.177, pp.3627–3635, 2007.

[43] B. Bede, I.J. Rudas, A.L. Bencsik, First order linear fuzzy differential equations under generalized differentiability, *Inform. Sci.* Vol.177, pp.1648-1662, 2007.

[44] B. Bede, L. Stefanini, Generalized differentiability of fuzzy-valued functions, *Fuzzy Sets Syst.* Vol.230, pp.119-141, 2013.

[45] F.P. Beer, E.R. Johnston, *Mechanics of materials*, 2nd Edition, mcgraw-Hill. 1992.

[46] S.S. Behzadi, T. Allahviranloo, Solving fuzzy differential equations by using picard method, *Iranian Journal of Fuzzy Systems.* Vol.13, pp.71-81, 2016.

[47] D. Boffi, L. Gastaldi, Interpolation estimates for edge finite elements and application to band gap computation, *Appl. Numer. Math.* Vol.56, pp.1283-1292, 2006.

[48] A. Bonarini, G. Bontempi, A qualitative simulation approach for fuzzy dynamical models, *ACM Transaction Modelling and Computer Simulation,* Vol.4, pp.285-313, 1994.

[49] J.J. Buckley, E. Eslami, Neural net solutions to fuzzy problems: The quadratic equation, *Fuzzy Sets Syst.* Vol.86, pp.289-298, 1997.

[50] J.J. Buckley, E. Eslami, T. Feuring, Fuzzy mathematics in economics and engineering, *Studies in fuzziness and soft computing, Phisica-Verlag,* 2002.

[51] J.J. Buckley, E. Eslami, Y. Hayashi, Solving fuzzy equations using neural nets, *Fuzzy Sets Syst.* Vol.86, pp.271-278, 1997.

[52] J.J. Buckley, T. Feuring, Fuzzy differential equations, *Fuzzy Sets Syst.* Vol.110, pp.69-77, 2000.

[53] J.J. Buckley, T. Feuring, Fuzzy initial value problem for nth-order linear differential equations, *Fuzzy Sets Syst.* Vol.121, pp.247-255, 2001.

[54] J.J. Buckley, T. Feuring, Y. Hayashi, Solving fuzzy equations using evolutionary algorithms and neural nets, *Soft comput.* Vol.6, pp.116-123, 2002.

[55] J.J. Buckley, Y. Hayashi, Can fuzzy neural nets approximate continuous fuzzy functions, *Fuzzy Sets Syst.* Vol.61, pp.43-51, 1994.

[56] J.J. Buckley, Y. Qu, Solving linear and quadratic fuzzy equations, *Fuzzy Sets Syst.* Vol.35, pp.43-59, 1990.

[57] B. Bulbul, M. Sezer, Taylor polynomial solution of hyperbolic type partial differential equations with constant coefficients, *Int. J. Comput. Math.* Vol 88, pp.533-544, 2011.

[58] O. Brudaru, F. Leon, O. Buzatu, Genetic algorithm for solving fuzzy equations, *8th International Symposium on Automatic Control and Computer Science, Iasi, ISBN,* 2004.

[59] J.D. Faires, Numerical Analysis, *seventh ed., PWS-Kent, Boston,* 2001.

[60] R.L. Burden, J.D. Faires, Numerical Analysis, *Thomson, Pacific Grove, Calif, USA, 8th edition,* 2005.

[61] H.B. Cary, S.C. Helzer, Modern Welding Technology. Upper Saddle River, *New Jersey: Pearson Education.* 2005.

[62] D. Catania, V. Georgiev, Blow-up for the semilinear wave equation in the Schwarzschild metric, *Differ. Integral. Equ.* Vol.19 pp.799-830, 2006.

[63] J.-C. Chang, H. Chen, S.-M. Shyu, W.-C. Lian, Fixed-point theorems in fuzzy real line, *Comput. Math. Appl.* Vol.47, pp.845-851, 2004.

[64] S.L. Chang, L.A. Zadeh, On fuzzy mapping and control, *IEEE Trans. Syst. Man Cybern.* Vol.2, pp.30-34, 1972.

[65] S.M. Chen, Fuzzy system reliability analysis using fuzzy number arithmetic operations, *Fuzzy Sets Syst.* Vol.64, pp.31-38, 1994.

[66] Y. Cherruault, Convergence of Adomian's method, *Kybernetes,* Vol.9, pp.31-38, 1988.

[67] L. Colzani, A. Cominardi, K. Stempak, Radial solutions to the wave equation, *Ann. Mat. Pura Appl.* Vol.181, pp.25-54, 2002.

[68] S. Curtis, S. Ghosh, A variable selection approach to monotonic regression with Bernstein polynomials, *J. Appl. Stat.* Vol.38, pp.961-976, 2011.

[69] G. Cybenko, Approximation by Superposition of Sigmoidal Activation Function, *Math. Control, Sig Syst*, Vol.2, 303-314, 1989.

[70] P.J. Davis, Interpolation and approximation, *Dover Publications, Inc., New York,* 1975.

[71] M. Delgado, M.A. Vila, W. Voxman, On a canonical representation of fuzzy Numbers, *Fuzzy Sets Syst.* Vol.93, pp.125-135, 1998.

[72] J. Demongeot, C. Jezequel, S. Sylvain Sene, Boundary Conditions and Phase transitions in Neural Networks, *Theoretical results. Neural Networks.* Vol.21, pp.971-979, 2008.

[73] P. Diamond, Fuzzy least squares, *Inform. Sci.* Vol.46, pp.141-157, 1988.

[74] M.W.M.G. Dissanayake, N. Phan-Thien, Neural-network based approximations for solving partial differential equations, *Communications in Numerical Methods in Engineering.* Vol.10, pp.195-201, 2000.

[75] D. Dubois, H. Prade, Towards fuzzy differential calculus: part 3 differentiation, *Fuzzy Sets Syst.* Vol.8, pp.225-233, 1982.

[76] D. Dubois, H. Prade, Fuzzy Set Theoretic Differences and Inclusions and Their Use in The analysis of Fuzzy Equations, *Control Cybern.* Vol.13, pp.129-146, 1984.

[77] S. Effati, M. Pakdaman, Artificial neural network approach for solving fuzzy differential equations, *Inform. Sci.* Vol.180, pp.1434-1457, 2010.

[78] S.S. Esmaili, A.M. Nasrabadi, Different initial conditions in fuzzy Tumor model, *Journal of Biomedical Science and Engineering,* Vol.3, pp.1001-1005, 2010.

[79] S. Evje, K.H. Karlsen, Monotone difference approximations of BV solutions to degenerate convection-diffusion equations, *SIAM. J. Numer. Anal.* Vol.37, pp.1838-1860, 2000.

[80] S. Falletta, G. Monegato, L. Scuderi, A space–time BIE method for nonhomogeneous exterior wave equation problems, *The Dirichlet case, IMA J. Numer. Anal.* Vol.32, pp.202-226, 2012.

[81] A. Farajzadeh, A. Hossein Pour, M. Amini, An explicit method for solving fuzzy partial differential equation, *International Mathematical Forum,* Vol.5, pp.1025 - 1036, 2010.

[82] O.S. Fard, A. Esfahani, A.V. Kamyad, On solution of a class of fuzzy BVPs, *Iran. J. Fuzzy Syst.* Vol.9, pp.49-60, 2012.

[83] G. Feng, A survey on analysis and design of model-based fuzzy control systems, *IEEE Trans. Fuzzy Syst.* Vol.14, pp.676-697, 2006.

[84] C. Filici, On a neural approximator to ODEs. *IEEE Transactions on Neural Networks,* Vol.19, pp.539-543, 2008.

[85] M. Friedman, M. Ming, A. Kandel, Fuzzy linear systems,*Fuzzy Sets Syst.* Vol.96, pp.201-209, 1998.

[86] M. Friedman, M. Ming, A. Kandel, Duality in fuzzy linear systems,*Fuzzy Sets Syst.* Vol.109, pp.55-58, 2000.

[87] L.A. Gardashova, Application of operational approaches to solving decision making problem using Z-Numbers, *Journal of Applied Mathematics.* Vol.5, pp.1323-1334, 2014.

[88] M. Gerdts, G. Greif, H.J. Pesch, Numerical optimal control of the wave equation: Optimal boundary control of a string to rest in finite time, *Math. Comput. Simul.* Vol.79, pp.1020-1032, 2008.

[89] B. Ghazanfari, A. Shakerami, Numerical solutions of fuzzy differential equations by extended Runge–Kutta-like formulae of order 4, *Fuzzy Sets Syst.* Vol.189, pp.74-91, 2011.

[90] J.S. Gibson, An analysis of optimal modal regulation: convergence and stability, *SIAM J. Control. Optim.* Vol.19, pp.686-707, 1981.

[91] R. Goetschel, W.Voxman, Elementary calculus, *Fuzzy Sets Syst.* Vol.18, pp.31-43, 1986.

[92] S. Guo, L. Mei, Y. Zhou, The compound $\frac{G'}{G}$-expansion method and double non-traveling wave solutions of (2+1)-dimensional nonlinear partial differential equations, *COMPUT. MATH. APPL.* Vol.69, pp.804-816, 2015.

[93] Y.M. Ham, C. Chun, S.G. Lee, Some higher-order modifications of Newton's method for solving nonlinear equations, *J. Comput. Appl. Math.* Vol. 222, pp.477-486, 2008.

[94] M.S. Hashemi, J. Malekinagad, Series solution of fuzzy wave-like equations with variable coe±cients, *J. Intell. Fuzzy Syst.* Vol.25, pp.415-428, 2013.

[95] F. Hawrra, K.H. Amal, On fuzzy Laplace transform for fuzzy differential equations of the third order, *Journal of Kerbala University,* Vol.11, pp.251-256, 2013.

[96] S. Haykin, Neural Networks. A comprehensive Foundation, *IEEE Press, New York,* 1994.

[97] M. Hazewinkel, Oscillator harmonic, *Springer, ISBN,* 2001.

[98] S. He, K. Reif, R. Unbehauen, Multilayer neural networks for solving a class of partial differential equations, *Neural Networks.* Vol.13 pp.385-396, 2000.

[99] R.L. Higdon, Absorbing boundary conditions for difference approximations tothe multidimensional wave equation, *Math. Comput.* Vol.47, pp.437-459, 1986.

[100] J.P. Holman, *Heat transfer*, 8th Ed, McGraw-Hill. New York. 1997.

[101] E. Hüllermeier, An approach to modeling and simulation of uncertain dynamical systems, *Internat. J. Uncertainty Fuzzyness Knowledge-Based Syst.* Vol.5, pp.117-137, 1997.

[102] T.J.R. Hughes, The finite element method, *Prentice Hall, New Jersey,* 1987.

[103] E.A. Hussian , M.H. Suhhiem, Numerical solution of fuzzy differential equations based on Taylor series by using fuzzy neural networks, *advance in mathematics,* Vol.11, pp.4080-4092, 2015.

[104] E.A. Hussian , M.H. Suhhiem, Numerical solution of fuzzy differential equations by using modified fuzzy neural network, *International Journal of Mathematical Archive,* Vol.6, pp.84-94, 2015.

[105] E.A. Hussian , M.H. Suhhiem, Numerical solution of fuzzy partial differential Equations by using modified fuzzy neural networks, *British Journal of Mathematics and Computer Science,* Vol.12, pp.1-20, 2016.

[106] J.M. Hyman, M. Shashkov, The approximation of boundary conditions for mimetic finite difference methods, *Computers Math. Applic.* Vol.36, pp.79-99, 1998.

[107] H.N.A. Ismail, K. Raslan, A.A.A. Robboh, Adomain decomposition method for Burgers-Huxley and Burger's-Fisher equation, *Appl. Math. Comput.* Vol.159, pp.291–301, 2004.

[108] Y. Ito, Independence of unscaled basis functions and finite mappings by neural networks, *Math. Sci.* Vol.26, pp.117-126, 2001.

[109] H. Jafari, M. Saeidy, M.A. Firoozjaee, The homotopy analysis method for solving higher dimensional initial boundary value problems of variable coefficients, *Numer. Meth. Part. D. E.* Vol.26, pp.1021-1032, 2010.

[110] R. Jafari, W. Yu, Fuzzy Control for Uncertainty Nonlinear Systems with Dual Fuzzy Equations, *Journal of Intelligent and Fuzzy Systems.* Vol.29, pp.1229-1240, 2015.

[111] A. Jafarian, R. Jafari, A. Khalili, D. Baleanud, Solving fully fuzzy polynomials using feed-back neural networks. *International Journal of Computer Mathematics*, Vol.92, pp.742-755, 2015.

[112] A. Jafarian, S. Measoomynia, Solving fuzzy polynomials using neural nets with a new learning algorithm, *Appl. Math. Sci.* Vol.5, pp.2295-2301, 2011.

[113] M.J. Jang, C.L. Chen, Y.C. Liu, Two-dimensional differential transform for partial differential equations, *Appl. Math. Comput.* Vol.121, pp.261-270, 2001.

[114] M. Javidi, Spectral collocation method for the solution of the generalized Burger's-Fisher equation, *Appl. Math. Comput.* Vol.174, pp.345-352, 2006.

[115] T. Jayakumar, D. Maheskumar, K. Kanagarajan, Numerical solution of fuzzy differential equations by Runge Kutta method of order five, *Appl. Math. Sci.* Vol.6, pp.2989-3002, 2012.

[116] T. Jayakumar, K. Kanagarajan, S. Indrakumar, Numerical solution of Nth-order fuzzy differential equation by Runge-Kutta Nystrom method, *International Journal of Mathematical Engineering and Science,* Vol.1, pp.2277-6982, 2012.

[117] L. Jin, Homotopy perturbation method for solving partial differential equations with variable coefficients, *Int. J. Contemp. Math. Sci.* Vol.3, (2008) pp.1395-1407, 2008.

[118] U. Kadak, F. Basar, On some sets of fuzzy-valued sequences with the level sets, *Contemporary Analysis and Applied M athematics*, Vol.1, pp.70-90, 2013.

[119] M. Kajani, B. Asady, A. Vencheh, An iterative method for solving dual fuzzy nonlinear equations. *Appl. Math. Comput.* Vol.167, pp.316-323, 2005.

[120] K. Kanagarajan, M. Sambath, Runge-Kutta Nystrom method of order three for solving fuzzy differential equations, *Computational Methods in Applied Mathematics,* Vol.10, pp.195-203, 2010.

[121] B. Kang, D. Wei, Y. Li, Y. Deng, A method of converting Z-number to classical fuzzy number, *Journal of Information and Computational Science.* Vol.9, pp.703-709, 2012.

[122] B. Kang, D. Wei, Y. Li, Y. Deng, Decision making using Z–Numbers under uncertain environemnt, *Journal of Computational Information Systems.* Vol.8, pp.2807-2814, 2012.

[123] K. Kemati, M. Matinfar, An implicit method for fuzzy parabolic partial differential equations, *J. Nonlinear Sci. Appl.* Vol.1, pp.61-71, 2008.

[124] A. Kharab, R. Kharab, Spreadsheet solution of hyperbolic partial differential equation, *IEEE Trans. Educ.* Vol.40, pp.103-110, 1997.

[125] A. Khastan, K. Ivaz, Numerical solution of fuzzy differential equations by Nyström method, *Chaos, Solitons. Fractals.* Vol. 41, pp.859-868, 2009.

[126] A. Khastan, J.J. Nieto, R. Rodríguez-López, Periodic boundary value problems for first-order linear differential equations with uncertainty under generalized differentiability, *Inform. Sci.* Vol.222, pp.544-558, 2013.

[127] H. Kim, R. Sakthivel, Numerical solution of hybrid fuzzy differential equations using improved predictor–corrector method, *Commun Nonlinear Sci Numer Simulat,* Vol.17, pp.3788-3794, 2012.

[128] D. Kincaid, W. Cheney, Numerical Analysis, *Mathematics of Scientific Computing, Pacific Grove. CA: Brooks/Cole,* 1991.

[129] A. Kröner, K. Kunisch, A minimum effort optimal control problem for the wave equation, *Comput. Optim. Appl.* Vol.57, pp.241-270, 2014.

[130] I.E. Lagaris, A. Likas, D.I. Fotiadis, Artificial neural networks for solving ordinary and partial differential equations, *IEEE Transactions on Neural Networks.* Vol.9, pp.987-1000, 1998.

[131] V. Laksmikantham, R.N. Mohapatra, Theory of Fuzzy Differential Equations and Inclusions, *Taylor and Francis,* NewYork, 2003.

[132] H. Lee, I.S. Kang, Neural algorithms for solving differential equations, *Journal of Computational Physics.* Vol.91, pp.110-131, 1990.

[133] R.J. LeVeque, Finite difference methods for differential equations, *University of Washington,* 2005.

[134] F.T Lin, simulating fuzzy numbers for solving fuzzy equations with constraints using genetic algorithm, *International Journal of Innovative Computing, Information and Control* Vol.6, pp.239–253, 2010.

[135] H.K. Liu, Comparison results of two-point fuzzy boundary value problems, *Int. J. Comput. Math. Sci.* Vol.5, pp.1-7, 2011.

[136] B. Llanas, F.J. Sainz, Constructive approximate interpolation by neural networks, *J. Comput. Appl. Math.* Vol. 188, pp.283-308, 2006.

[137] W.A. Lodwick, K.D. Jamison, Special issue: interfaces between fuzzy set theory and interval analysis, *Fuzzy Sets Syst.* Vol.135, pp.1-3, 2003.

[138] V. Lupulescu, On a class of fuzzy functional differential equations, *Fuzzy Sets Syst.* vol.160, pp.1547-1562, 2009.

[139] M.A. Ma, M. Friedman, A. Kandel, Numerical solutions of fuzzy differential equations, *Fuzzy Sets Syst.* Vol.105, pp.133-138, 1999.

[140] E.H. Mamdani, Application of fuzzy algorithms for control of simple dynamic plant, *IEE Proceedings-Control Theory and Applications.* Vol.121, pp.1585-1588, 1976.

[141] P. Mansouri, B. Asady, N. Gupta, An approximation algorithm for fuzzy polynomial interpolation with Artificial Bee Colony algorithm, *Appl. Soft. Comput.* Vol.13, pp.1997-2002, 2013.

[142] P. Martinez, A new method to obtain decay rate estimates for dissipative systems, *ESAIM Control Optim. Calc. Var.* Vol.4, pp.419-444, 1999.

[143] M.H. Mashinchi, M.R. Mashinchi, S. Mariyam, H.J. Shamsuddin, A genetic algorithm approach for solving fuzzy linear and quadratic equations, *World Academy of Science, Engineering and Technology,* Vol.1, No:4, 2007.

[144] M. Mastylo, Interpolation estimates for entropy numbers with applications to non-convex bodies, *J. Apprpx. Thery.* Vol.162, pp.10-23, 2010.

[145] M. Mazandarani, A.V. Kamyad, Modified fractional Euler method for solving fuzzy fractional initial value problem, *Commun. Nonlinear Sci. Numer. Simul.* Vo.18, pp.12-21, 2013.

[146] A.J. Meada, A.A. Fernandez, The numerical solution of linear ordinary differential equation by feed forward neural network, *Math. Comput. Model.* Vol.19, pp.1-25, 1994.

[147] A.J. Meade, A.A. Fernandez, Solution of nonlinear ordinary differential equations by feedforward neural networks, *Math. Comput. Model.* Vol.20, pp.19-44, 1994.

[148] H.N. Mhaskar, D.V. Pai, Fundamentals of Approximation Theory, *CRC Press, Boca Raton. FL*, 2000.

[149] Z. Michalewicz, Genetic Algorithms + Data Structures = Evolution Program, *2nd ed., Artificial Intelligence, Springer Verlag, Berlin,* 1994.

[150] N. Mikaeilvand, S. Khakrangin, Solving fuzzy partial differential equations by fuzzy two-dimensional differential transform method, *Neural. Comput. Appl.* Vol.21, pp.307-312, 2012.

[151] M. Moghimi, F.S.A. Hejazi, Variational iteration method for solving generalized Burger-Fisher and Burger equation, *Chaos, Solitons and Fractals.* Vol.33, pp.1756-1761, 2007.

[152] C. Montelora, C. Saloma, Solving the nonlinear schrodinger equation with an unsupervised neural network: Estimation of error in solution, *Opt. Commun.* Vol.222 pp.331-339, 2003.

[153] M. Mosleh, Evaluation of fully fuzzy matrix equations by fuzzy neural network, *Appl. Math. Model.* Vol.37, pp.6364-6376, 2013.

[154] M. Mosleh, M. Otadi, S. Abbasbandy, Fuzzy polynomial regression with fuzzy neural networks, *Appl. Math. Model.* Vol.35, pp.5400-5412, 2011.

[155] M. Mosleh, M. Otadi, Simulation and evaluation of fuzzy differential equations by fuzzy neural network, *Appl. Soft. Comput.* Vol.12, pp.2817-2827, 2012.

[156] M. Mosleh, M. Otadi, Solving the second order fuzzy differential equations by fuzzy neural network, *Journal of Mathematical Extension* Vol.8, pp.11-27, 2014.

[157] E. Nasrabadi, Fuzzy linear regression models analysis: mathematical-programming methods, *M.Sc. Thesis, Department of Industrial Engineering, Sharif University of Technology, Tehran,* 2003.

[158] M.M. Nasrabadi, E. Nasrabadi, A.R. Nasrabadi, Fuzzy linear regression analysis: a multi-objective programming approach, *Appl. Math. Comput.* Vol. 163, pp.245-251, 2005.

[159] R.D. Neidinger, Multivariable interpolating polynomials in newton forms, *In Joint mathematics meetings, Washington, DC*, pp.5-8, 2009.

[160] K. Nemati, M. Matinfar, An implicit method for fuzzy parabolic partial differential equations, *J. Nonlinear Sci. Appl.* Vol.1, pp.61-71, 2008.

[161] I. Newton, Methodus fluxionum et serierum infinitarum, 1664-1671.

[162] J.J. Nieto, A. Khastan, K. Ivaz, Numerical solution of fuzzy differential equations under generalized differentiability, *Nonlinear Anal. Hybrid Syst.* Vol.3, pp.700-707, 2009.

[163] J.J. Nieto, R. Rodríguez-López, M. Villanueva-Pesqueira, Exact solution to the periodic boundary value problem for a first-order linear fuzzy diffe rential equation with impulses, *Fuzzy Optim. Decis. Making.* Vol.10, pp.323-339, 2011.

[164] H.T. Nguyen, A note on the extension principle for fuzzy sets, *J. Math. Anal. Appl.* Vol.64, pp.369-380, 1978.

[165] A. Noorani, J. Kavikumar, M. Mustafa, and S. Nor, Solving dual fuzzy polynomial equation by ranking method, *Far East J. Math. Sci.* Vol.15, pp.151-163, 2011.

[166] R. Nurhakimah Ab, A. Lazim, Solving Fuzzy Polynomial Equation and The Dual Fuzzy Polynomial Equation Using The Ranking Method, *The 3th International conference on Mathematics Scinces* pp. 798-804, 2014.

[167] R. Nurhakimah Ab, A. Lazim, An Interval Type-2 Dual Fuzzy Polynomial Equations and Ranking Method of Fuzzy Numbers, *International Journal of Mathematical, Computational, Physical, Electrical and Computer Engineering,* Vol.8, 2014.

[168] R. Nurhakimah Ab, A. Lazim, Ab.G. Ahmad Termimi, A. Noorani, Solutions of Interval Type-2 Fuzzy Polynomials Using A New Ranking Method, *AIP Conference,* Vol.1682, 2015.

[169] P.J. Olver, On multivariate interpolation, *Studies in Applied Mathematics,* Vol.116, pp.201-240, 2006.

[170] M. Otadi, Fully fuzzy polynomial regression with fuzzy neural networks, *Neurocomputing,* Vol.142, pp.486-493, 2014.

[171] M. Otadi, M. Mosleh, Solution of fuzzy polynomial equations by modified Adomian decomposition method, *Soft. Comput.* Vol.15, pp.187-192, 2011.

[172] M. Otadi, M. Mosleh, S. Abbasbandy, Numerical solution of fully fuzzy linear systems by fuzzy neural network, *Soft Comput.* Vol.15, pp.1513-1522, 2011.

[173] S.C. Palligkinis, G. Papageorgiou, I.T. Famelis, Runge-Kutta methods for fuzzy differential equations, *Applied Mathematics and Computation* Vol.209, pp.97-105, 2009.

[174] S. Pederson, M. Sambandham, The Runge-Kutta method forhybrid fuzzy differential equation, *Nonlinear Anal. Hybrid Syst.* Vol.2, pp.626-634, 2008.

[175] S. Pederson, M. Sambandham, Numerical solution to hybrid fuzzy systems. *Math. Comput. Model.* Vol.45, pp.1133–1144, 2007.

[176] G. Peters, Fuzzy linear regression with fuzzy intervals, *Fuzzy Sets Syst.* Vol.63, pp.45-55, 1994.

[177] H. Pohlheim, Documentation for genetic and evolutionary algorithm toolbox for use with matlab (GEATbx), *version 1.92, http://www.geatbx.com,* 1999.

[178] H. Radstrom, An embedding theorem for spaces of convex sets, *Proc. Amer. Math. Soc.* Vol.3, pp.165-169, 1952.

[179] D. Rajan, S. Chaudhuri, Generalized interpolation and its application in super-resolution imaging, *Image. Vision. Comput.* Vol.19, pp.957-969, 2001.

[180] D. Ralescu, Toward a general theory of fuzzy variables, *Journal of Mathematical Analysis and Applications,* Vol.86, pp.176-193, 1982.

[181] A. Ramli, M.L. Abdullah, M. Mamat, Broyden's method for solving fuzzy nonlinear equations, *Advances in Fuzzy Systems,* Vol.2010, Article ID 763270, 6 pages, doi:10.1155/2010/763270.

[182] H. Roman-Flores, M. Rojas-Medar, Embedding of level-continuous fuzzy sets on Banach spaces, *Inform. Sci.* Vol.144, pp.227-247, 2002.

[183] H. Rouhparvar, Solving fuzzy polynomial equation by ranking method, *First Joint Congress on Fuzzy and Intelligent Systems, Ferdowsi University of Mashhad, Iran,* 2007.

[184] Xu. Ruoning, S-curve regression model in fuzzy environment, *Fuzzy Sets Syst.* Vol.90, pp.317-326, 1997.

[185] M. Sakawa, H. Yano, Multi objective fuzzy linear regression analysis for fuzzy input–output data, *Fuzzy Sets Syst.* Vol.47, pp.173-181, 1992.

[186] S. Salahshour, T. Allahviranloo, S. Abbasbandy, Solving fuzzy fractional differential equations by fuzzy Laplace transforms, *Commun. Non-linear Sci. Numer. Simul.* Vol.17, pp.1372-1381, 2012.

[187] E. Sanchez, Solution of Fuzzy Equations with Extended Operations,*Fuzzy Sets Syst.* Vol.12, pp.273-248, 1984.

[188] D. Savic and W. Pedryzc, Evaluation of fuzzy linear regression models, *Fuzzy Sets Syst.* Vol.39, pp.51-63, 1991.

[189] H. Schroeder, V.K. Murthy, E.V. Krishnamurthy, Systolic algorithm for polynomial interpolation and related problems, *Parallel Computing.* pp.493-503, 1991.

[190] P. Sevastjanov, L. Dymova, A new method for solving interval and fuzzy equations: Linear case, *Inform. Sci.* Vol.179, pp.925-937, 2009.

[191] O. Solaymani Fard, T. AliAbdoli Bidgoli, The Nyström method for hybrid fuzzy differential equation IVPs, *Journal of King Saud University-Science,* Vol.23, pp.371-379, 2011.

[192] O. Solaymani Fard, N. Ghal-Eh, Numerical solutions for linear system of first-order fuzzy differential equations with fuzzy constant coeffcients, *Inform. Sci.* Vol.181, pp.4765-4779, 2011.

[193] G.D. Smith, Numerical solution of partial differential equations: finite difference methods. *Clarendon Press, Oxford,* 1978.

[194] M.W. Spong, M. Vidyasagar, Robot Dynamics and Control, *John Wiley and Sons,* 1989.

[195] L. Stefanini, B. Bede, Generalized Hukuhara differentiability of interval-valued functions and interval differential equations, *J. Nonlinear Anal. (TMA)* Vol.71, pp.1311-1328, 2009.

[196] V.L. Streeter, E.B. Wylie, E. Benjamin, *Fluid mechanics*, 4th Ed, Mc. Graw-Hill Book Company. 1999.

[197] P.V. Subrahmanyam, S.K. Sudarsanam, On some fuzzy functional equations, *Fuzzy Sets Syst.* Vol.64, pp.333-338, 1994.

[198] N. Sukavanam, V. Panwar, Computation of boundary control of controlled heat equation using artificial neural networks, *Int. Commun. Heat Mass Transfer.* Vol.30, pp.1137-1146, 2003.

[199] J.A.K. Suykens, J. De Brabanter, L. Lukas, J. Vandewalle, Weighted least squares support vector machines: robustness and sparse approximation, *Neurocomputing,* Vol.48 pp.85-105, 2002.

[200] J. Szabados, P. Vertesi, Interpolation of Functions, *World Scientific*, Singapore, 1990.

[201] A. Tahavvor, M. Yaghoubi, Analysis of natural convection from a column of cold horizontal cylinders using artificial neural network, *Appl. Math. Model.* Vol.36, pp.3176-3188, 2012.

[202] T. Takagi, M. Sugeno, Fuzzy identification of systems and its applications to modeling and control, *IEEE Trans. Syst., Man, Cybern.* Vol.15, pp.116-132, 1985.

[203] H. Tanaka, I. Hayashi, J. Watada, Possibilistic linear regression analysis for fuzzy data, *Eur. J. Oper. Res.* Vol.40, pp.389-396, 1989.

[204] H. Tanaka, S. Uegima, K. Asai, Linear regression analysis with fuzzy model, *IEEE Trans. Syst. ManCybern.* Vol.12, pp.903-907, 1982.

[205] S. Tapaswini, S. Chakraverty, Euler-based new solution method for fuzzy initial value problems, *Int. J. Artificial. Intell. Soft. Comput.* Vol.4, pp.58-79, 2014.

[206] S. Tapaswini, S. Chakraverty, A new approach to fuzzy initial value problem by improved Euler method, *Fuzzy Inf. Eng.* Vol.3, pp.293-312, 2012.

[207] V.M. Tikhomirov, Approximation theory, *In analysis II, R.V. Gamkrelidze, ed., Encyclopaedia of Mathematical Sciences,* Springer-Verlag, Berlin, Vol.14, 1990.

[208] A.R. Vahidi, B. Jalalvand, Improving the Accuracy of the Adomian Decomposition Method for Solving Nonlinear Equations, *Appl. Math. Sci.* Vol.6, pp.487-497, 2012.

[209] N.K. Vitanov, Z.I. Dimitrova, H. Kantz, Modified method of simplest equation and its application to nonlinear PDEs, *Appl. Math. Comput.* Vol.216, pp.2587-2595, 2010.

[210] M. Wagenknecht, R.Hampel, V.Schneider, Computational aspects of fuzzy arithmetics based on archimedean t-norms, *Fuzzy Sets Syst.* Vol.123, pp.49-62, 2001.

[211] H. Wang, Y. Liu, Existence results for fuzzy integral equations of fractional order, *Int. J. Math. Anal.* Vol.5, pp.811-818, 2011.

[212] H.F. Wang, R.C. Tsaur, Resolution of fuzzy regression model, *Europ. J. Oper. Res.* Vol.126, pp.637-650, 2000.

[213] M. Waziri, Z. Majid, A new approach for solving dual fuzzy nonlinear equations using Broyden's and Newton's methods, *Advances in Fuzzy Systems*, Vol.2012, Article 682087, 5 pages, 2012.

[214] A.M. Wazwaz, The variational iteration method: A reliable analytic tool for solving linear and nonlinear wave equations, *Comput. Math. Appl.* Vol.54, pp.926-932, 2007.

[215] D. Xiu, J.S. Hesthaven. High-order collocation methods for differential equations with random inputs, *SIAM J Sci Comput.* Vol.27, pp.1118-1139, 2005.

[216] L.B. Yang, Y.Y. Gao, Fuzzy Mathematics-Theory and its Application, *published by South China University of Technology, Guangzhou China*, 1993.

[217] H.S. Yazdi, R. Pourreza, Unsupervised adaptive neural-fuzzy inference system for solving differential equations, *Appl. Soft. Comput.* Vol.10, pp.267-275, 2010.

[218] W. Yu, X. Li, Fuzzy identification using fuzzy neural networks with stable learning algorithms, *IEEE Transactions on Fuzzy Systems*, Vol.12, pp.411-420, 2004.

[219] L.A. Zadeh, The concept of a liguistic variable and its application to approximate reasoning, *Inform. Sci.* Vol.8, pp.199-249, 1975.

[220] L.A. Zadeh, Toward a generalized theory of uncertainty (GTU) an outline, *Inform. Sci.* Vol.172, pp.1-40, 2005.

[221] L.A. Zadeh, Generalized theory of uncertainty (GTU)-principal concepts and ideas, *Computational Statistics and Data Analysis.* Vol.51, pp.15-46, 2006.

[222] L.A. Zadeh, A note on Z-numbers, *Inform. Sci.* Vol. 181, pp.2923-2932, 2011.

[223] A. Zolic, Numerical mathematics, *Faculty of mathematics: Belgrade.* pp.91-97, 2008.