Numerical Solutions for Systems of 'Qualitative' Nonlinear Algebraic Equations by Fuzzy Logic

Majura F. Selekwa; Emmanuel G. Collins, Jr.

Department of Mechanical Engineering
Florida A&M University- Florida State University
Tallahassee,FL 32310

Abstract

This paper presents a fuzzy logic approach for determining a numerical solution to a consistent system of algebraic equations $F(x) = 0$ in which the function $F(\cdot)$ is not explicitly defined and may be underdetermined. Such systems arise frequently in many engineering design problems where design parameters must be chosen using qualitative information by the designer to meet a set of desired performance constraints. The proposed method also can be used for a consistent system of nonlinear equations in which $F(\cdot)$ is explicitly defined and may have fewer independent equations than the number of unknowns. However, this method is very computationally demanding; hence, it is not advisable to apply it to problems involving explicit functions that can be solved using other existing numerical methods. It is seen that this method works quite well and numerical solutions for such problems can be obtained, although it is much slower than Newton's method when employed to consistent, explicit nonlinear equations.

Keywords: Analysis, Design, Fuzzy mathematical programming.

1 Introduction

The numerical solution of a consistent system of algebraic nonlinear equations $F(x) = 0$, where $F : \mathbb{R}^n \rightarrow \mathbb{R}^m$, arise quite often in engineering and the natural sciences. Many engineering design problems that must satisfy specified constraints can be expressed as systems of nonlinear equalities and inequalities. Numerical solutions for these problems can be attempted after simplifying and expressing all nonlinear inequalities as nonlinear equations. Analytical

[1] This research was supported in part by the Army Research Office grant DAAD19-01-1-0720

and numerical methods for solving these problems are limited to special cases where $F(\cdot)$ is explicitly defined, and $m = n$, i.e., the number of unknowns equals the number of independent equations. For some engineering problems these conditions are not met; $F(\cdot)$ is not known explicitly and there are more unknowns than the available independent equations, i.e., $m < n$. A system of nonlinear functions $F(x)$ is said to be inexplicitly defined if the form and coefficients of its terms are not known while the image of its arguments x can be described qualitatively. For example, the design of the classic Proportional-Integral-Derivative (PID) controller requires three gains to be chosen, K_P, K_I and K_D, that may be viewed as the elements of the vector x. It is desired to choose x so that a certain rise time, overshoot, settling time and steady-state error are achieved. This results in a problem of the form $F(x) = 0$ where $F(\cdot)$ is not explicitly known, but may be computed via numerical integration of certain differential equations. Problems of this kind are referred to in this paper as systems of 'qualitative' nonlinear algebraic equations. They in general arise when $F(\cdot)$ must be computed by complex numerical calculations or numerical simulation.

This paper develops a method of solving the above problems by using fuzzy logic. Variations of this method have been successfully used in modern control for choosing design parameters so that the resultant $\mathcal{H}_2$ or $\mathcal{H}_\infty$ controllers satisfy specified performance constraints for the controlled system [1–3]. These variations were developed based on the general trends observed in the responses of linear systems as the design parameters for these controllers are varied; therefore the resulting fuzzy algorithms are applicable only to the specific control design problems to which those trends apply. For the general scenario it may not be practically feasible to characterize the trends in how $F(x)$ changes in response to changes in the elements of x. The method presented in this paper does not need to have prior knowledge of the system behavior. It generates a sequence of x that is associated with a converging sequence of the response error. The desired solution is obtained when the response error is approximately zero.

The proposed method also can solve explicit systems of linear and nonlinear equations by finding the solution vector that results in minimum norm; additionally, it can also be applied in approximating a solution of an inconsistent system of nonlinear and linear equations. However, because of its computational complexity, there is no advantage in applying this method to these problems. Such problems should be handled by standard iterative methods such as the Successive Overrelaxation Method for linear problems and gradient methods for nonlinear problems.

Computational methods that use fuzzy logic have attracted attention in recent years; most of them have addressed optimization problems. The methods of [4,5] were for solving a crisp linear programming problem by searching and

reasoning. Extension of these methods to fuzzy mathematical programs in which the linear program has fuzzy coefficients was accomplished in [6] and [7,8]. Recently an aggressive fuzzy method to solve a multilevel fuzzy programming problem has been proposed [9,10]. None of these methods have directly addressed the problem of qualitative nonlinear equations as proposed here. Although the fuzzy linear (or nonlinear) programming problem also can be cast as $F(x) = 0$ where $F : \mathbb{R}^n \to \mathbb{R}^m$, methods designed specifically for these problems cannot generally solve $F(x) = 0$ unless $F(\cdot)$ has a special structure.

The paper is organized as follows: Section 2 reviews the problem of explicit nonlinear functions and the gradient methods for solving it. The section also describes the inexplicit nonlinear function problem that cannot be handled by the methods used for the explicit nonlinear functions problem, and hence motivate the need for the proposed method. Section 3 develops a fuzzy algorithm that solves this problem. Numerical examples using the proposed fuzzy algorithm are given in Section 4 and Section 5 closes the paper with concluding remarks.

Notation: $\mathbb{R}$ is a field of real numbers, $x^{(k)}$ is the k-th element in a sequence, x_k is the k-th element of the vector x. For $M \in \mathbb{R}^{m\times n}$, $||M||_\infty \triangleq \max_{i,j}\{M_{i,j}\}$, where $M_{i,j}$ is the (i,j) element of M. If $x = [x_1, x_2, \cdots, x_n]^T$, then $|x| \triangleq [|x_1|, |x_2|, \cdots, |x_n|]^T$. For $x, y \in \mathbb{R}^n$, $|x| \leq\leq |y|$ means $|x_i| \leq |y_i|$, $\forall i = 1, 2, \cdots, n$.

2 Solution systems of nonlinear equations and problem statement

The problem considered in this paper is that of determining the numeric value of $x \in \mathbb{R}^n$ that satisfies a consistent system of nonlinear equations

$$F(x) = 0, \tag{1}$$

where $F : \mathbb{R}^n \mapsto \mathbb{R}^m$ is an inexplicit nonlinear function. If the function $F(\cdot)$ is explicit and $m = n$, then it is possible to determine some function $\phi : \mathbb{R}^n \mapsto \mathbb{R}^n$ such that the desired numeric solution $\overline{x}$ of (1) is its fixed point, i.e.,

$$\overline{x} = \phi(\overline{x}), \text{ where } F(\overline{x}) = 0. \tag{2}$$

This function is then used in determining the solution iteratively from some initial estimate $x^{(0)}$ by

$$x^{(k+1)} = \phi(x^{(k)}), \tag{3}$$

until (2) is satisfied.

The most common approaches of implementing (3) are based on Newton's method and its variations (see [11–13]), all together known as gradient methods. These gradient methods cannot handle the problem addressed by this paper because of their need for $F(x)$ to be explicitly defined in terms of x, and m to be equal to n. This paper considers the problem of computing the solution for systems in which (1) is not explicitly defined, but is qualitatively known, and may be underdetermined (i.e., $m < n$).

3 Solution of qualitative nonlinear equations by fuzzy logic

This section describes the proposed fuzzy logic method for solving inexplicit nonlinear systems $F(x) = 0$, which also holds when $F : \mathbb{R}^n \mapsto \mathbb{R}^m$ with $m \leq n$. Like most other numerical methods, this method searches the solution by gradual minimization of the functional approximation error.

3.1 The basic principle

Consider any sequence $\{\alpha^{(k)}\} \subset \mathbb{R}^n$; if it satisfies $\left|\alpha^{(k+1)}\right| \leq\leq \left|\alpha^{(k)}\right|$, then it is said to be an *enclosed sequence* and it is always convergent to some value $\overline{\alpha}$. Suppose that for any sequence $\{x^{(k)}\} \subset \mathbb{R}^n$, there exists some sequence $\{\epsilon^{(k)}\} \subset \mathbb{R}^m$ such that

$$F(x^{(k)}) = \epsilon^{(k)}. \tag{4}$$

where $\epsilon^{(k)}$ is a sequence of the functional error. The solution to the problem in (1) is basically the value of $x^{(k)}$ that renders $\epsilon^{(k)} = 0$. Most iterative methods seek for the converging sequence $x^{(k)}$ with $\lim_{k\to\infty} x^{(k)} = \overline{x}$ where $\overline{x}$ is the desired solution. This results in a sequence $\left\{\epsilon^{(k)}\right\}$ that is not necessarily an enclosed sequence; however, any enclosed sequence $\left\{\epsilon^{(k)}\right\}$ with $\lim_{k\to\infty} \epsilon^{(k)} = 0$ is associated with a sequence $\left\{x^{(k)}\right\}$ for which $\lim_{k\to\infty} x^{(k)} = \overline{x}$.

The fuzzy logic method proposed here generates an enclosed sequence $\left\{\epsilon^{(k)}\right\}$ with $\lim_{k\to\infty} \epsilon^{(k)} = 0$; it is equivalent to finding the sequence

$$x^{(k)} = F^{\times}(\epsilon^{(k)}). \tag{5}$$

with $\lim_{k\to\infty} x^{(k)} = \overline{x}$, where the map $F^{\times} : \mathbb{R}^m \to \mathbb{R}^n$ satisfies $F^{\times}(F(x)) = x$. There may be several ways of generating a sequence $\left\{\epsilon^{(k)}\right\}$ with enclosure properties. If $\epsilon^{(k)} = [\epsilon_1^{(k)}, \quad \epsilon_2^{(k)}, \quad \cdots, \quad \epsilon_m^{(k)}]^T$, one of the most direct ways of generating $\left\{\epsilon^{(k)}\right\}$ with these properties is by searching for $x^{(k+1)}$ that sets

$\epsilon_l^{(k+1)} = 0$ without changing $\epsilon_j^{(k)}$ for $j \neq l$ were l is the index corresponding to the largest element in $|\epsilon^{(k)}|$. Under this ideal setup, the solution of (1) can be reached in m outer iterations. Inner iterations may be needed to find the appropriate $x^{(k)}$.

3.2 Implementation by fuzzy logic computation

An approximation of the above procedure can be implemented by the iteration

$$x^{(k+1)} = x^{(k)} + \Delta x^{(k)}, \qquad k = 1, 2 \cdots . \tag{6}$$

where $\Delta x^{(k)}$ is chosen by the fuzzy algorithm so that $x^{(k+1)}$ sets $\epsilon_l^{(k+1)} \approx 0$ while having a small effect on the elements $\epsilon_j^{(k+1)}$ for $j \neq l$. The choice of $\Delta x^{(k)}$ depends on both the value of $\epsilon^{(k)}$ and its *sensitivity* to the change in $x^{(k)}$ defined by $\frac{\partial \epsilon^{(k)}}{\partial x^{(k)}} \equiv \frac{\partial F(x^{(k)})}{\partial x^{(k)}}$. Note that if the sensitivity of $\epsilon^{(k)}$ to a change in $x^{(k)}$ is high, then changing $x^{(k)}$ by small amounts results in a large change in $\epsilon^{(k)}$ and vice versa. When $F(x)$ is explicit, the sensitivity matrix $\frac{\partial \epsilon^{(k)}}{\partial x^{(k)}}$ can be calculated analytically; however, for an inexplicit $F(x)$, the sensitivity matrix must be computed numerically by finite differences for each iteration.

The algorithm requires two fuzzy systems; the first system determines $\delta x_{i^*}^{(k)}$ that drives $\epsilon_l^{(k+1)}$ to zero assuming that the function $F(x^{(k)})$ is linear. Because of the inherent coupling in the system, the choice of $\delta x_{i^*}^{(k)}$ causes $\epsilon_j^{(k+1)}, j \neq l$ fail to maintain their original values $\epsilon_j^{(k)}$. The second fuzzy system is designed to detect changes in $\epsilon_j^{(k+1)}, j \neq l$ which must be compensated for before the end of each iteration k. After these changes have been detected, the first fuzzy system is applied to the whole system again by finding additional changes in all elements $\delta x_j^{(k)}$, such that $\epsilon_l^{(k+1)} \approx 0$ while elements $\epsilon_j^{(k+1)} \approx \epsilon_j^{(k)}, j \neq l$.

Let the first fuzzy system be FS-1 and the second fuzzy system be FS-2. The variables involved in this problem are the elements of vectors $\epsilon^{(k)}$, $x^{(k)}$, and the sensitivity matrix $\frac{\partial \epsilon^{(k)}}{\partial x^{(k)}}$. The inputs to FS-1 are the elements of the vector $\epsilon^{(k)}$ and the matrix $\frac{\partial \epsilon^{(k)}}{\partial x^{(k)}}$, and the resulting outputs are the elements of $x^{(k)}$. System FS-2 has the elements of vector $x^{(k)}$ and matrix $\frac{\partial \epsilon^{(k)}}{\partial x^{(k)}}$ as its inputs and the elements of $\epsilon^{(k)}$ as outputs. These systems are Mamdani-type systems in which the crisp outputs are obtained by center of area defuzzification method [14,15].

Group	Linguistic value	Symbol
1	Large Negative	LN
2	Negative	N
3	Zero	Z
4	Positive	P
5	Large Positive	LP

Table 1
The fuzzy groups for the variables used in the examples.

3.2.1 Fuzzification of the variables and design of the fuzzy rule bases

The fuzzy systems described above are for searching the solution space. In order to provide a balanced direction of search, all variables are fuzzified into an odd number of symmetric fuzzy groups centered at zero. Five groups as defined in Table 1 are used here. The three fuzzy groups around zero are slightly finer than the outer two fuzzy groups in order to have more precision when the variables are close to zero. The corresponding membership functions are shown in Figure 1; these are standard z-functions, triangular functions, and s-functions [14,15].

The universe of discourse for each of these fuzzy groups is the interval $[-Z, Z]$ where Z is the infinity norm of each variable, i.e., $\left|\left|\epsilon^{(k)}\right|\right|_\infty$, $\left|\left|x^{(k)}\right|\right|_\infty$, and $\left|\left|\frac{\partial \epsilon^{(k)}}{\partial x^{(k)}}\right|\right|_\infty$. Initially the universe of discourse will decrease gradually according to the infinity norm of each variable; after several iterations it can be held fixed since it is sufficiently small. In the examples given in this paper, it was always held fixed to 1 after reaching this value. The rule bases for both FS-1

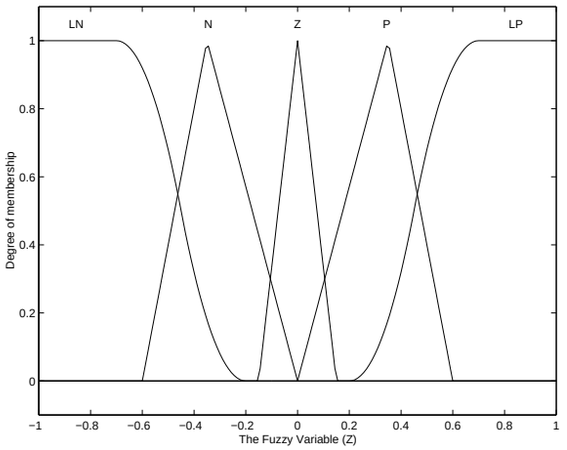

Fig. 1. Typical membership functions for the variables.

and FS-2 are designed as shown in Tables 2 and 3 respectively. These rules are formulated by assuming local linearity of the functionals. Tables 2 and 3

$\eta_{i^*l} \setminus \hat{\epsilon}_l$	**LN**	**N**	**Z**	**P**	**LP**
LN	LN	N	Z	P	LP
N	LN	LN	Z	LP	LP
Z	Z	Z	Z	Z	Z
P	LP	LP	Z	LN	LN
LP	LP	P	Z	N	LN

Table 2
The fuzzy rules for FS-1 where $\hat{\epsilon}_i \triangleq \max_i\{\epsilon_i\}$

are respectively collections of rules of the form

$$\textbf{IF } \epsilon_l \text{ and } \hat{\eta}_{i^*l} \textbf{ THEN } \delta x_{i^*}, \tag{7}$$

and

$$\textbf{IF } \delta x_{i^*} \text{ and } \eta_{i^*j} \textbf{ THEN } \epsilon_j. \tag{8}$$

$\eta_{i^*j} \setminus \ \delta x_i^*$	**LN**	**N**	**Z**	**P**	**LP**
LN	LP	LP	Z	LN	LN
N	LP	P	Z	N	LN
Z	Z	Z	Z	Z	Z
P	LN	N	Z	P	PL
LP	LN	LN	Z	LP	LP

Table 3
The fuzzy rules for FS-2.

3.2.2 The fuzzy logic algorithm

Define $\mu_{i,j}^{(k)}$ as the i,j-*th* element of the sensitivity matrix $\frac{\partial \epsilon^{(k)}}{\partial x^{(k)}}$. The fuzzy algorithm adjusts solution parameters $x_j^{(k)}$ to which the deviation vector $\epsilon^{(k)}$ is most sensitive. At each iteration, it seeks to bring down to zero the largest element of $\left|\epsilon^{(k)}\right|$ until $\left|\left|\epsilon^{(k)}\right|\right| = 0$. The algorithm flow is as follows:

Initialize; then for $k = 1, 2, \ldots m$ do:

Step 0: Compute $\epsilon^{(k)} = [\epsilon_1^{(k)}, \quad \epsilon_2^{(k)}, \quad \cdots, \quad \epsilon_m^{(k)}]^T$; if $||\epsilon^{(k)}||$ is small enough or cannot be improved further, then STOP.

Step 1: Find $\widehat{\varepsilon}_l^{(k)} = \max_i\{\epsilon_i^{(k)}\},\ i = 1, 2, \ldots, n$ and

$$\mu_{i^*,l}^{(k)} \triangleq \max_i \left[\text{row}_l \left(\frac{\partial \epsilon^{(k)}}{\partial x^{(k)}}\right)\right],$$

where

$$i^* = \arg\max_i \left[\text{row}_l \left(\frac{\partial \epsilon^{(k)}}{\partial x^{(k)}}\right)\right]$$

is the position of the largest element of $\mu_{i,l}$ in row l. Use the rules of FS-1 to determine the change in $x_{i^*}^{(k)}$ that would bring $\widehat{\varepsilon}_l^{(k)}$ to zero. Let it be $\delta x_{i^*}^{(k)}$.

Step 2: Use $\delta x_{i^*}^{(k)},\ \mu_{i^*,j}^{(k)},\ j = 1, 2, \ldots, n,,\ j \neq i^*$ and the rules of FS-2 to determine the amount of change $\epsilon_j^{(k)}$ by which all other elements of $\epsilon^{(k)}$ will be changed by $\delta x_{i^*}^{(k)}$. Let it be $\delta\epsilon_j^{(k)}$.

Step 3: For j =1, 2,$\ldots, n,\ j \neq l$

– Use $\delta\epsilon_j^{(k)}$, $\mu_{j,i^*}^{(k)}$, and the rules of FS-1 to determine $\delta x_j^{(k)}$ that offset the changes $\delta\epsilon_j^{(k)}$ in all elements $j \neq l$ as observed in Step 2 above.

Step 4: Set $\Delta x^{(k)} = [\delta x_1^{(k)}, \delta x_2^{(k)}, \cdots, \delta x_{i^*}^{(k)}, \cdots, \delta x_n^{(k)}]^T$ and update

$$x^{(k+1)} = x^{(k)} + \Delta x^{(k)}.$$

In Step 3 above, it may not be possible to compensate for the changes in each element $j : j \neq l$. Components that cannot be compensated for will have to be skipped, although this weakens the effectiveness of the algorithm. Since it is not possible to reduce the value of each element $\epsilon_i^{(k)}$ to zero, the algorithm will take more than m-outer iterations to converge. For that reason, the update element $\delta x_{i^*}^{(k)}$ in Step 4 will not necessarily be in a known order.

3.2.3 Complexity and Convergence of the Method

This method is computationally intensive, and is not suitable for explicit problems for which other numerical methods can be applied. The computational load of this method increases exponentially with the sum of the number of equations (m) and the number of parameters to be determined (n), i.e., $m+n$.

Although it is very hard to prove that this algorithm will eventually converge to the desired solution, the basic principle (Subsection 3.1) that leads to this algorithm shows that if a new element ϵ_i $(i = 1, 2, \ldots, m)$ can be reduced to zero at each iteration, then the algorithm will converge in m iterations. What is required is to show that this algorithm will actually conserve this condition. Currently tools for proving this are not known, and hence no guarantee can be made on the convergence of the algorithm. The convergence of fuzzy based algorithms is still an open research problem and many fuzzy based algorithms

are given without proof of convergence, such as those in [4–10]. However, in all numerical experiments that were conducted by the authors, the algorithm converged successfully.

4 Numerical Examples

Three numerical examples are given to show the application of the proposed algorithm. In the first example, the fuzzy algorithm is compared with other methods for a determinate problem, i.e., $m = n$. The second example illustrates the use of the fuzzy algorithm for an underdetermined problem, i.e $m \leq n$. Both these examples use functions that are explicitly defined and can be solved by other numerical methods. The objective of using these functions is to ascertain that the proposed method actually works. In the third example, an inexplicit problem is introduced and the proposed algorithm is applied to find the solution. In each of these examples, the fuzzy algorithm treats $F(x)$ as inexplicit, so that the sensitivity matrix $\frac{\partial \epsilon^{(k)}}{\partial x^{(k)}}$ is computed using finite differences.

4.1 Example 1

This example considers the Broyden banded function of [16]. This function is given by

$$F_i(x) = x_i(2 + 5x_i^2) + 1 - \sum_{j \in J_i} x_j(1 + x_j), \tag{9}$$

where $i = 1, 2, \ldots, m$ and J_i is defined such that

$$J_i = \{j : j \neq i, \max(1, i - 5) \leq j \leq \min(m, i + 1)\}. \tag{10}$$

As in [16], it was assumed that $m = 3$. When the fuzzy algorithm was employed with an initial estimate of $x_0 = (1, 1, 1)$, the following solution was obtained,

$$\begin{pmatrix} x_1 \\ x_2 \\ x_3 \end{pmatrix} = \begin{pmatrix} -0.42830 \\ -0.47657 \\ -0.47657 \end{pmatrix}. \tag{11}$$

This result agrees with both the results of [16] by interval computation and the Newton's method, where the initial vector was also chosen to be $x_0 = (1, 1, 1)$. However, the algorithm was significantly slower than Newton's method. Using a 400MHz Intel Celeron Processor, it required 66×10^{-2} seconds to solve this problem while Newton's method required only 6×10^{-2} seconds.

4.2 Example 2

This example illustrates the use of the proposed method for solving an underdetermined nonlinear system. The problem is defined by

$$F(x_1, x_2, x_3) = \begin{bmatrix} 3x_1 - \cos(x_2 x_3) - 0.5 \\ \exp(-x_1 x_2) + 20x_3 + (10\pi - 3)/3 \end{bmatrix}. \tag{12}$$

Using the fuzzy algorithm with $x_0 = (1, 1, 1)$, the solution obtained was

$$\begin{pmatrix} x_1 \\ x_2 \\ x_3 \end{pmatrix} = \begin{pmatrix} 0.45836 \\ 1.00000 \\ -0.50521 \end{pmatrix}. \tag{13}$$

This solution satisfies the above problem, although as a characteristic of all underdetermined systems, it is not the only solution. Many other solutions are possible depending on the initial estimate x_0; for instance when $x_0 = (5, 5, 5)$, the solution obtained by the fuzzy algorithm becomes:

$$\begin{pmatrix} x_1 \\ x_2 \\ x_3 \end{pmatrix} = \begin{pmatrix} 0.15854 \\ 5.00000 \\ -0.58407 \end{pmatrix}, \tag{14}$$

which also satisfies the system in (12)

4.3 Example 3

This example considers one of the systems of qualitative algebraic equations found in control design tasks. For further details about this problem, the reader is referred to [1]. For simplicity consider a linear system $\mathcal{G}(\cdot)$ defined such that

$$z(t) = \mathcal{G}(u(t), \nu(t)), \tag{15}$$

where, $u \in \mathbb{R}^m$ is the control input, $z \in \mathbb{R}^q$ is the output, and $\nu \in \mathbb{R}^q$ is a zero mean, white, Gaussian noise process with unity intensity. A standard Linear Quadratic Gaussian (LQG) control is one that minimizes the cost function

$$J = \mathrm{tr}(Q_z V_z) + \mathrm{tr}(Q_u V_u), \tag{16}$$

where $Q_z = Q_z^T \geq 0$, and $Q_u = Q_u^T > 0$, are the respective output and input performance weights selected by the designer, and $\mathrm{tr}(\cdot)$ is a matrix trace

operator. Matrices V_z and V_u denote the steady-state covariance matrices of the output and input vectors respectively; they can be computed using the controller and the plant information. The resulting LQG controller depends on the system $\mathcal{G}$, the noise ν and the weight matrices Q_z and Q_x. Quite often, designers are asked to select the weights Q_z and Q_u such that the resulting input and output variances of the system are constrained within some limits. There is no explicit function that relates the variances of the system and the weights Q_z and Q_u of the cost function.

It has been shown [1] that if the deviations of the variances from the constrains are denoted by E_i and matrices Q_z and Q_u are chosen to be diagonal, i.e.,

$$Q_z = \mathrm{diag}\left(\alpha_{z,1}, \cdots, \alpha_{z,q}\right), \quad Q_u = \mathrm{diag}\left(\alpha_{u,1}, \cdots, \alpha_{u,m}\right), \tag{17}$$

then the weight selection problem subject to some variance constraints is equivalent to solution of a system of nonlinear inequalities

$$\mathrm{E}_i(\alpha_{z,1}, \alpha_{z,2}, \cdots, \alpha_{z,q}, \alpha_{u,1}, \alpha_{u,2}, \cdots, \alpha_{u,m}) \leq 0, \quad i = 1, 2, \cdots, q+m. \tag{18}$$

One solution of (18) is found by replacing the inequality with an equality forming a system of qualitative algebraic inequalities that can be solved by using the proposed algorithm. The function E_i that relates the weights $\alpha_{i,j}$ to the variance deviations is not explicitly defined; the problem is solved by iteratively finding the elements $\alpha_{z,1}$,$\alpha_{z,2}$,$\cdots$,$\alpha_{z,q}$,$\alpha_{u,1}$, $\alpha_{u,2}$,$\cdots$,$\alpha_{u,m}$ of the weight matrices Q_z and Q_u whose image under E_i, $\forall i$, is closet to zero. In particular if the system $\mathcal{G}$ is described in the state space form as

$$\begin{aligned} \mathcal{G}: \quad & \dot{x}(t) = Ax(t) + Bu(t) + D_1\nu(t), && (19)\\ & y(t) = Cx(t) + D_2\nu(t), && (20)\\ & z(t) = Ex(t), && (21) \end{aligned}$$

where all symbols have their standard meaning, this algorithm can select the desired weights to satisfy any conceivable set of variance constraints. The algorithm was tested for the problem that was addressed in [1] in which

$$\begin{aligned} A &= \begin{bmatrix} 0.00 & 1.00 & 0.00 & 0.00 \\ -1.00 & -0.01 & 0.00 & 0.00 \\ 0.00 & 0.00 & 0.00 & 1.00 \\ 0.00 & 0.00 & -16.00 & -0.04 \end{bmatrix}, \quad B = \begin{bmatrix} 0.0000 & 0.0000 \\ 0.5878 & -1.0000 \\ 0.0000 & 0.0000 \\ 0.9511 & 2.0000 \end{bmatrix}, \\ C = E &= \begin{bmatrix} 1.0000 & 0.00 & 2.0000 & 0.00 \\ 0.9511 & 0.00 & -0.5878 & 0.00 \end{bmatrix}, \qquad \begin{aligned} D_1 &= 10^{-1.5} I_4, \\ D_2 &= 10^{-2} I_2. \end{aligned} \end{aligned} \tag{22}$$

Different sets of variance constraints on both the inputs and outputs were chosen and the algorithm yielded weights that satisfied these constraints. Typical

Variance Constraints		Obtained Variances		Selected Weights	
Input	**Output**	V_u	V_z	Q_u	Q_z
$\begin{bmatrix} 0.0004 \\ 0.0022 \end{bmatrix}$	$\begin{bmatrix} 0.0056 \\ 0.0020 \end{bmatrix}$	$\begin{bmatrix} 0.0004 \\ 0.0022 \end{bmatrix}$	$\begin{bmatrix} 0.0056 \\ 0.0020 \end{bmatrix}$	$\begin{bmatrix} 1.2194 \\ 0.7877 \end{bmatrix}$	$\begin{bmatrix} 0.4613 \\ 1.5115 \end{bmatrix}$
$\begin{bmatrix} 0.0002 \\ 0.0026 \end{bmatrix}$	$\begin{bmatrix} 0.0058 \\ 0.0020 \end{bmatrix}$	$\begin{bmatrix} 0.0002 \\ 0.0026 \end{bmatrix}$	$\begin{bmatrix} 0.0058 \\ 0.0020 \end{bmatrix}$	$\begin{bmatrix} 1.4249 \\ 0.5379 \end{bmatrix}$	$\begin{bmatrix} 0.3281 \\ 1.2096 \end{bmatrix}$
$\begin{bmatrix} 0.0074 \\ 0.0002 \end{bmatrix}$	$\begin{bmatrix} 0.0053 \\ 0.0020 \end{bmatrix}$	$\begin{bmatrix} 0.0073 \\ 0.0002 \end{bmatrix}$	$\begin{bmatrix} 0.0053 \\ 0.0020 \end{bmatrix}$	$\begin{bmatrix} 0.2299 \\ 3.2279 \end{bmatrix}$	$\begin{bmatrix} 0.3613 \\ 1.6648 \end{bmatrix}$
$\begin{bmatrix} 0.0003 \\ 0.004 \end{bmatrix}$	$\begin{bmatrix} 0.0049 \\ 0.0018 \end{bmatrix}$	$\begin{bmatrix} 0.0003 \\ 0.0040 \end{bmatrix}$	$\begin{bmatrix} 0.0049 \\ 0.0018 \end{bmatrix}$	$\begin{bmatrix} 2.5044 \\ 0.6801 \end{bmatrix}$	$\begin{bmatrix} 1.3764 \\ 2.7110 \end{bmatrix}$

Table 4
Selected variance constraints, obtained variances and selected weights

results are shown in Table 4.

5 Conclusion

A fuzzy logic method for finding a numeric solution to a system of qualitative nonlinear equations of the form $F(x) = 0$, has been presented. This method is an automated search procedure that starts by an initial guess that introduces a functional error. The method uses finite differences to compute an estimate of the sensitivity matrix $\frac{\partial F(x)}{\partial x}$, which along with the functional error are applied by the fuzzy system to determine values of x in a way that shrinks the functional error. The desired solution is obtained when the functional error is either small enough or cannot be further improved. Numerical results show that the proposed method works quite well and satisfactory solutions can be obtained for systems of explicitly and inexplicitly defined nonlinear equations, including underdetermined systems. By computing the sensitivity matrix numerically and searching the solution space for $\Delta x^{(k)}$, this method is relatively slow for explicitly defined determinate problems when compared with Newton's method.

References

[1] E. G. Collins, Jr., M. F. Selekwa, A Fuzzy Logic Approach to LQG Design with Variance Constraints, IEEE Trans. Control Syst. Tech. 10 (1) (2002) 32–42.

[2] M. F. Selekwa, E. G. Collins, Jr., $\mathcal{H}_2$ Optimal Reduced Order Control Design Using A Fuzzy Logic Methodology With Bounds on System Variances, IEEE Trans. Control Syst. Tech. 11 (1) (2003) 153–156.

[3] M. F. Selekwa, E. G. Collins, Jr., $\mathcal{H}_\infty$ Loop Shaping Control Design With Time Domain Constraints: Application of Fuzzy Logic, in: Proc. IEEE 40th Conf. on Dec. and Cont., 2001, pp. 4358–4363, orlando, Florida, USA.

[4] J. J. Buckley, Possibilistic linear programming with triangular fuzzy numbers, Journal of Fuzzy Sets and Systems 26 (1988) 135–138.

[5] J. J. Buckley, Solving possibilistic linear programming problems, Journal of Fuzzy Sets and Systems 31 (1989) 329–341.

[6] R. Fuller, H. J. Zimmermann, Fuzzy reasonong for solving fuzzy mathematical programming problems, Journal of Fuzzy Sets and Systems 60 (1993) 121–133.

[7] C. L. Cheng, D. Y. Sun, Solution of fuzzy dynamic optimization problems by adaptive stochastic algorithm, International Journal of Artificial Intelligence Tools 9 (4) (2000) 527–535.

[8] C. L. Cheng, D. Y. Sun, C. Y. Chang, Numerical solution of dynamic optimization problems by adaptive stochastic algorithm, Journal of Fuzzy Sets and Systems 127 (2002) 165–176.

[9] S. Sinha, Fuzzy mathematical programming applied to multi-level programming problems, Computers & Operations Research 30 (2003) 1259–1268.

[10] S. Sinha, Fuzzy mathematical programming approach to multi-level programming problems, Journal of Fuzzy Sets and Systems 136 (2003) 189–202.

[11] S. G. Nash, A. Sofer, Linear and Nonlinear Programming, McGraw Hill, New York, 1996.

[12] H. Ratschek, J. Rokne, New Computer Methods for Global Optimization, Mathematics and Its Applications, Ellis Horwood, Chichester, 1988.

[13] R. Fletcher, Practical Methods of Optimization: Second Edition, John Wiley, New York, 1987.

[14] N. Gulley, J. S. Jang, Fuzzy Logic Toolbox User's Guide, The Mathworks Inc, Massachusetts, 1995.

[15] G. J. Klir, B. Yuan, Fuzzy Sets and Fuzzy Logic, Prentice-Hall, New Jersey, 1995.

[16] E. R. Hansen, R. I. Greenberg, An Interval Newton Method, Applied Math. and Comp. 12 (1983) 89–98.