Chapter 2
Genetic Fuzzy System

The damage detection problem is a pattern classification problem which is based on ambiguous, noisy, or missing input information. The input information is typically obtained from sensors placed on the structure or embedded in the structure. Sensor measurements are often subjected to some processing before being used as damage indicators. Though considerable effort is made to find damage indicators which show a high degree of sensitivity to the damage size and location, the problems of ambiguity, imprecision, and noise present in the measured data are likely to propagate into the damage indicators. In fact, in some cases, the signal processing performed on the sensor measurements may increase the noise level in the damage indicators. As an example, differentiation of data to get a damage indicator can amplify the noise contamination.

Based on this information from sensors, a damage detection system should provide definite outputs to help the maintenance engineers. For example, the following questions need to be answered.

1. Is there damage in the system?
2. Where is the damage?
3. What is the damage size?
4. What needs to be done?
5. How much longer can the structure be used?

The answers to these questions should be as accurate as possible. Answers expressed in words instead of in numbers are often more useful for maintenance engineers. Among the several soft computing methods, fuzzy logic is the one which maps numerical inputs into linguistic outputs. However, fuzzy logic lacks the capability of learning from the given data, and the rules which govern the fuzzy system must be developed by human experts. This process of developing the fuzzy rule base is difficult and can become very complicated if the number of inputs and outputs increases. The hybridization of fuzzy logic with a genetic algorithm gives an advanced soft computing algorithm called the genetic fuzzy system (GFS), which generates the fuzzy rules automatically from the data. In this chapter, the terms used in the formulation of a GFS for a damage detection problem are explained. The first step in developing the GFS is to understand the concepts underlying fuzzy logic.

P.M. Pawar, R. Ganguli, *Structural Health Monitoring Using Genetic Fuzzy Systems*,
DOI 10.1007/978-0-85729-907-9_2,

2.1 Fuzzy Logic System

Fuzzy logic is a unique soft computing method which simultaneously handles numerical data and linguistic knowledge. This unique feature has led it to be called "computing with words." People use fuzzy logic all the time to arrive at decisions in the complex settings in which they operate. For instance, we may conclude that a "fat" person who is "sedentary" is likely to have "health problems." In this case, the words "fat" and "sedentary" represent fuzzy concepts. These concepts could be numerically quantified using exact weight bounds and exercise bounds. However, human reasoning does not operate using such numbers and still often reaches surprisingly accurate conclusions using fuzzy rules. Another important fact in fuzzy logic is the degree of membership. For example, a person may have "fatness" between the levels of 0 and 1. He or she may also be "sedentary" between the levels of 0 and 1. A fuzzy rule could then be stated as: If a person is "fat" and "sedentary" then he or she has "health problems." We can see that an extremely overweight person who is completely sedentary is "very likely" to have health problems. A marginally "fat" person who is completely sedentary may also have some health problems. A very "fat" person who is not "sedentary" may also have some health problems. Finally, a much less "fat" person who is much less "sedentary" may have far fewer health problems. Thus, fuzzy logic can address a variety of situations using the concept of degree of membership in a fuzzy set. Thus the membership in the fuzzy sets "fat" and "sedentary" is not binary (1 and 0) but can have any number between 0 and 1. This simple example clearly shows how humans constantly assign fuzzy memberships to concepts to perform decision making. The output in this case clearly represents a diagnosis of health.

We see that a key feature of fuzzy logic is the use of words for computing. Other soft computing methods such as neural networks and genetic algorithms typically work with numerical data only. In other words, they convert a set of numbers to another set of numbers. However, fuzzy logic converts a set of words into another set of words. Expert systems can also work with words but use crisp logic in the sense that the concepts can only be either 1 or 0. So the rules of expert systems are similar to if-then-else rules in computer programs. Expert systems tend to deteriorate very rapidly in the presence of noise in the data. Despite these differences, fuzzy systems can be interpreted as generalizations of rule-based expert systems with the binary logic framework being replaced by fuzzy logic.

A fuzzy logic system is a nonlinear mapping of an input feature vector into a scalar output [1]. Fuzzy set theory and fuzzy logic provide the framework for the nonlinear mapping. Fuzzy logic systems have been widely used in engineering applications, because of the flexibility they offer designers and their ability to handle uncertainty. A fuzzy logic system can be expressed as a linear combination of fuzzy basis functions and is a universal function approximator. A schematic diagram of fuzzy logic is shown in Fig. 2.1 for $\Delta\omega$ as crisp inputs and *damage size* and *damage location* as crisp outputs. Here, $\Delta\omega$ is a measurement delta which can be a change in frequency between the undamaged and damaged structure, or a change in the strain output measured by a piezoelectric sensor, for example. The measurement

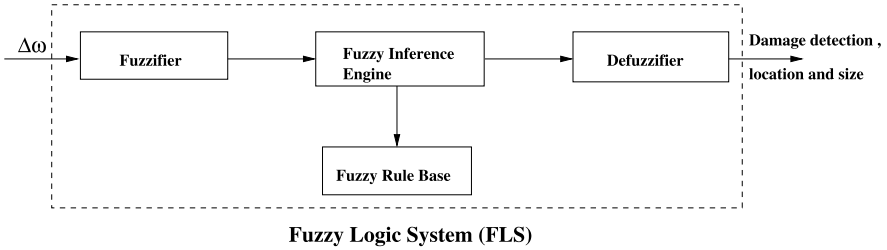

Fig. 2.1 Schematic representation of fuzzy logic system for SHM

deltas could also be changes in strains, blade tip responses, and blade root loads in a helicopter rotor blade. A very condensed introduction to fuzzy logic is provided below. Further information on fuzzy logic systems is available from textbooks [1–3]. Though fuzzy logic has become a vast topic, its application for structural health monitoring (SHM) requires the knowledge of only a few key concepts, which are outlined next.

A typical multi-input single-output (MISO) fuzzy logic system performs a mapping from a set $V \in R^m$ to $W \in R$ using four basic components: rules, fuzzifier, inference engine, and defuzzifier. Here, R^m refers to the space of m real numbers and assumes that m measurement deltas are present as inputs to the fuzzy system. The notation R refers to the space of real numbers and refers to the output of the fuzzy system. Thus the fuzzy system performs the mapping

$$F : V \in R^m \rightarrow W \in R$$

$$\text{where } V = V_1 \times V_2 \times \cdots \times V_n \in R^m$$

is the input space and $W \in R$ is the output space. A typical fuzzy logic system maps crisp inputs to crisp outputs using four basic components: rules, fuzzifier, inference engine, and defuzzifier. Once the rules driving the fuzzy logic system have been fixed, the fuzzy logic system can be expressed as a mapping of inputs to outputs.

Rules can come from experts or can be obtained from numerical data. When the rules come from experts, they can be directly represented as words. For example, an expert may suggest that when oil temperature measured by a sensor is high and vibration level at a particular accelerometer A is high, then there is damage at a location B in the structure. A process of interviewing of experts is often the best way to develop a fuzzy rule base directly from human knowledge which encodes the expert information often used by maintenance engineers. However, for many engineering problems, expert knowledge may not be available about the different conditions of the damaged system. In such cases, a mathematical model of the damaged system is invaluable for creating a rule base linking seeded damages to the damaged indicators. Therefore, the rules for a fuzzy logic system can come either from experts or from a mathematical model, depending on the system under consideration. In either case, engineering rules are expressed as a collection of IF-THEN statements such as "IF u_1 is HIGH, and u_2 is LOW, THEN v is LOW." We can see that some of the rules can come from experts while others can come from models.

For SHM, the designer must select the input damage indicators to the fuzzy system and the outputs of the fuzzy system. Various examples and case studies considered in this book will show some typical input-output sets. To physically understand the fuzzy rules, a damage detection problem for a cantilever beam can be considered where the inputs are the changes in natural frequencies relative to the baseline undamaged beam and the output is the location of the damage. To formulate a fuzzy rule, we need an understanding of

1. Linguistic variables (words) versus numerical values of a variable (e.g., HIGH versus 3.5% change in the fundamental frequency),
2. Quantifying linguistic variables (e.g., ω_1 (fundamental frequency) will have a finite number of linguistic terms associated with it, ranging from NEGLIGIBLE, SMALL, MEDIUM, to VERY HIGH), which is done using fuzzy membership functions,
3. Logical connections between linguistic variables (e.g., AND, OR etc.), and
4. Implications such as IF A THEN B. We also need to understand how to combine more than one rule.

Once the inputs of the fuzzy logic system are identified, a set of linguistic variables must be associated with each input. For instance, the change in the first natural frequency may vary from 0–5%. This set of $(0, 5)\%$ is now expressed as words, i.e., (negligible, small, medium, high, very high) assuming a discretization of five levels. This process of moving from the number space to the word space is accomplished by the fuzzification process. The fuzzifier performs the fuzzification and maps numbers into fuzzy sets. Thus a 0.5% change in the input may become a "small" change, and a 4.5% change in the input may become a "very high" change. This number-to-word transformation is very important in fuzzy logic, as all further operations such as rules are performed on the words. Thus, the fuzzifier is needed to activate rules that are expressed in terms of linguistic variables. An inference engine of the fuzzy logic system maps fuzzy sets to fuzzy sets and determines the way in which the fuzzy sets are combined. The inference engine therefore performs the operations for the fuzzy rules by converting the inputs expressed in words to the output expressed in words. The application of the rules on the words rather than on the numbers is the main source of the strength of the fuzzy logic system. The words are relatively insensitive to small changes in numbers and therefore are robust to uncertainty in the inputs. In several applications, numbers are needed as an output of the fuzzy logic system. In those cases, a defuzzifier is used to calculate crisp values from fuzzy values. Thus, the defuzzifier converts the words back to numbers. The internal architecture of the fuzzy system thus operates with words, but the interface to the outside world can be through numbers because of the fuzzifier and the defuzzifier. We will now formally define the various fuzzy logic terms.

Fuzzy Sets A fuzzy set F is defined on a universe of discourse U and is characterized by a degree of membership $\mu(x)$, which can take values between 0 and 1. A fuzzy set generalizes the concept of an ordinary set whose membership function only takes two values, zero and unity. Thus, an element must either belong to an ordinary set or not belong to it. However, an element can belong to two or more fuzzy

sets with different degrees of memberships. For example, if we classify height as "tall," "normal," and "short," then a given person may have a high degree of membership in "tall," a low degree of membership in "normal," and a very low degree of membership in "short." This allows the system to deal with ambiguity. Though the concept of height of a person is associated with a clear numerical value, human decisions which are made with height as an input typically use height as a fuzzy variable.

Linguistic Variables A linguistic variable u is used to represent the numerical value x, where x is an element of U. A linguistic variable is usually decomposed into a set of terms $T(u)$, which cover its universe of discourse.

Membership Functions The most commonly used shapes for membership functions $\mu(x)$ are triangular, trapezoidal, piecewise linear, or Gaussian. The designer selects the type of membership function used. There is no theoretical requirement that membership functions overlap. However, one of the major strengths of fuzzy logic is that membership functions can overlap. Fuzzy logic systems are robust because decisions are distributed over more than one input class. For convenience, membership functions are normalized to one so they take values between 0 and 1, and thus define the fuzzy set. One advantage of the Gaussian membership functions is that they do not suddenly go to zero. This helps in a progressive degradation of the behavior of the fuzzy logic system.

Inference Engine Rules for the fuzzy system can be expressed as

$$R_i : \text{IF } x_1 \text{ is } F_1 \text{ AND } x_2 \text{ is } F_2 \text{ AND } \cdots \text{ AND } x_m \text{ is } F_m \text{ THEN } y = C_i$$
$$i = 1, 2, 3, \ldots, M$$

where m and M are the number of input variables and rules, x_i and y are the input and output variables, and $F_i \in V_i$ and $C_i \in W$ are fuzzy sets characterized by membership functions $\mu_{F_i}(x)$ and $\mu_{C_i}(x)$, respectively. Each rule can be viewed as a fuzzy implication

$$F_{12\cdots m} = F_1 \times F_2 \times \cdots \times F_m \rightarrow C_i$$

which is a fuzzy set in $V \times W = V_1 \times V_2 \times \cdots \times V_m \times W$ with membership function given by

$$\mu_{R_i}(x, y) = \mu_{F_i}(x_1) * \mu_{F_2}(x_2) * \cdots * \mu_{F_m}(x_m) * \mu_{C_i}(y)$$

where the asterisk can be the min or product operator with $x = [x_1, x_2, \ldots, x_m] \in V$ and $y \in W$. This sort of rule covers many applications. The algebraic product is one of the most widely used operators in applications and leads to product implication. Underlying all this mathematics is a very simple concept that the degrees of memberships of the different parts of the rule are multiplied to get the degree to which a rule has fired. The Gaussian fuzzy sets are useful here, as they ensure that each rule is fired to some degree as none of the membership functions will become exactly zero. In pattern recognition problems, the outputs are often crisp sets, and $\mu_{C_i}(y) = 1$ is often used for the product implication formula. In this book, product implication will be used for the fuzzy systems.

Defuzzification Popular defuzzification methods include maximum matching and centroid defuzzification. While centroid defuzzification is widely used for fuzzy control problems where a numerical output is needed, maximum matching is often used for pattern matching problems where we need to know the output class. For example, a fuzzy control application may require an output in terms of an angle (degree or radian) for a given actuator. In these applications, the defuzzifier plays an important role. However, in SHM, it may be better to know that the damage is in the "outboard" section of the beam which could have been defined at the region between 60 and 80% of the beam length during the data gathering process. There is no great value in knowing that the damage is at an exact numerical location such as 73% of the beam length, as such a diagnosis is likely to be erroneous given the uncertainties in the problem. The output in words given by the fuzzy system is also useful for maintenance engineers and can be easily fed directly into the graphical user output.

Suppose there are K fuzzy rules and among them, K_j rules ($j = 1, 2, \ldots, L$ and L is the number of classes) produce class C_j. Let D_p^i be the measurements of how the pth pattern matched the antecedent conditions (IF part) of the ith rule, which is given by the product of membership grades of the pattern in the regions which the ith rule occupies:

$$D_p^i = \prod_{i=1}^{m} \mu_{li}, \tag{2.1}$$

where m is the number of inputs and μ_{li} is the degree of membership of measurement l in the fuzzy regions that the ith rule occupies. Let $D_p^{\max}(C_j)$ be the maximum matching degree of the rules (rules $j_l, l = 1, 2, \ldots, K_j$) generating class C_j:

$$D_p^{\max}(C_j) = \max_{l=1}^{K_j} D_p^{jl}. \tag{2.2}$$

Then the system will output class C_j* provided that

$$D_p^{\max}(C_j*) = \max_{j} D_p^{\max}(C_j). \tag{2.3}$$

If there are two or more classes which achieve the maximum matching degree, we will select the class which has the largest number of fired fuzzy rules (a fired rule has a matching degree of greater than zero).

There are several applications of fuzzy logic systems in SHM as well as in the broader area of engineering. Despite their considerable success, fuzzy systems are limited to problems with a small number of input variables. In addition, the process of developing the fuzzy system requires a lot of judgement and experience on the part of the designer. Two aspects in the design of the fuzzy system are particularly difficult: (1) generating the best rule set and (2) tuning the membership functions. The rules and the membership functions must accurately capture the relationship between the independent and dependent variables.

Unfortunately, the tasks of tuning the membership function and generating rules are not independent. The task of selecting membership functions and rule values is difficult since the information has to be obtained from numerical data of the system

to be modeled. Another problem is selecting an appropriate number of fuzzy sets. Most studies use experience to come up with this number. Often, symmetric fuzzy sets are assumed. However, assuming symmetry in the fuzzy sets also implies assuming symmetry in the system being modeled [4]. The results of successful fuzzy logic systems which one reads in papers and which have been implemented in many practical systems have come after much trial and error on the part of the designer. Typically, the designer selects a level of discretization for the measurement, then assigns the membership function for each fuzzy set, creates the rules, and checks for performance. If the performance of the fuzzy system is not good, the level of discretization, membership functions, and rules are manually tuned until a reasonably good level of performance is obtained. However, the danger of this approach is that it is ad hoc in nature and the fuzzy systems developed using this method are not optimal.

To use the power of fuzzy logic for realistic health monitoring problems, it is necessary to automate the process of fuzzy rule creation. For SHM problems, a clear metric for maximization is the success rate of the fuzzy system when confronted with test data. The design of the best fuzzy system for SHM is essentially an optimization problem which involves maximization of the success rate. As discussed in the previous chapter, the process of designing the best fuzzy logic system can be accelerated by using genetic algorithms. We therefore discuss genetic algorithms in the next section.

2.2 Genetic Algorithms

The genetic algorithm (GA) is a search process based on the laws of natural selection and genetics. The GA was one of the early examples of bio-inspiration in engineering and has paved the way for many other such methods and concepts. The GA searches the design space for an optimal design point to maximize a fitness function value. Generally, a simple GA contains three basic operations: selection, genetic operations, and replacement. A typical GA cycle is shown in Fig. 2.2. If Fig. 2.2 is examined, it will be seen that the first step needed to start the GA process is the construction of an initial population. The initial population is the potential solution set comprising a reasonably large number of points in the design space and is generated randomly or heuristically. The general method used at this stage is a random construction of individuals. It is also possible to use a combination of uniform sampling of the design space and random sampling in order to ensure that each part of the design space is represented in the initial population.

The next step of the pre-evolution phase is to evaluate the initial individuals. This step is needed to determine the next generation that will constitute the subpopulation. For a typical problem, the fitness function value needs to be calculated at each of the points in the population. There is a very small but finite probability that the initial population may possibly include the solution.

After this pre-evolution stage, the evolution phase loops until (1) a solution is found or (2) the generation number reaches the predetermined maximum generation

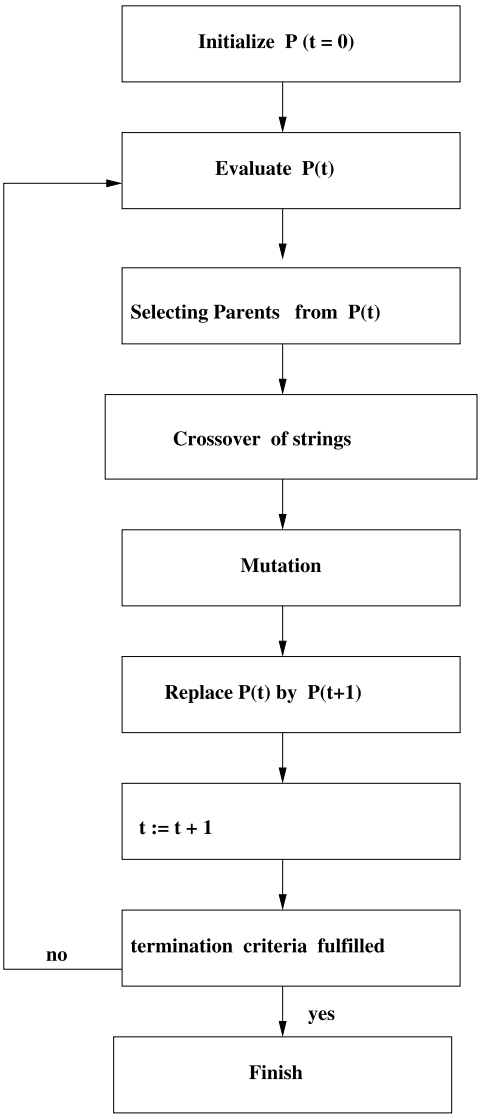

Fig. 2.2 Flowchart of genetic algorithm

number or (3) there remains no mutation combination left to try and any increase in the average success of the population cannot be achieved. Details about GAs are available in textbooks [5, 6], and their use in genetic fuzzy systems is described in [7]. GAs are theoretically and empirically proven to provide a robust search in complex spaces [8]. A GA operates on a population of randomly generated points (P). Each point is sometimes called a chromosome and is often represented by binary strings. There exist both binary and real coded versions of GAs; however, in this book we will use the binary GA, as this works quite well when a low level of discretization in terms of the numerics are needed. In a binary GA, any numbers

are first converted into binary form and then the genetic operators are applied to the resulting strings of 1's and 0's. The binary form resulting after the operations can then again be converted to real number form. However, the binary form requires a specification of the number of bits used for representing each design variable. These bits can be kept to a low number for SHM applications in genetic fuzzy systems as they typically represent the characteristics of the fuzzy set.

The GA is an optimization algorithm, and its advantage relative to traditional gradient-based algorithms lies in its ability to locate the global minimum and also operate with discrete or integer design variables. Several terms are widely used in the GA literature, and they are discussed next.

2.2.1 Operations During a GA Process

Encoding Encoding is the first part of a GA process, because problem-related information is encoded into a structure called a chromosome or string. A chromosome is generally a sequence of variables of a problem placed in an organized manner. Every variable sequenced to construct the chromosome is called a gene. These definitions come from the biologically inspired nature of the GA. However, as the GA moved away from its biological roots, genes were replaced by bits and chromosomes by strings. For instance, a design variable x can be represented by the string 110110. If there are two such design variables, they can be put side by side and result in a string of double the size. This process of moving from the real number space to the binary space is called encoding.

Fitness Evaluation GAs mimic the survival-of-the-fittest principle of nature to perform a search process. Therefore, GAs are naturally suitable for solving maximization problems where a fitness function is maximized. Minimization problems are usually transformed to maximization problems by some suitable transformation. In general, a fitness function $F(x)$ is first derived from the objective function $f(x)$ and used in successive genetic operations. For maximization problems, the fitness function can be considered to be the same as the objective function, i.e., $F(x) = f(x)$. For minimization problems, the fitness function is an equivalent maximization problem chosen such that the optimum point remains unchanged. A number of such transformations are possible. The following fitness function is often used:

$$F(x) = 1/\big(1 + f(x)\big).$$

GA operators typically require the function value to remain positive. Therefore, the process of changing the sign of the fitness function which is popular in gradient-based optimization is not used in GAs. If there is some chance that the fitness function may become negative in the design space, a large positive number can be added to it to ensure that the fitness stays positive.

Genetic Operations The operation of GAs begins with a population of random strings representing design variables. Thereafter, each string is evaluated to find the fitness value. The population is then operated by three main operators, *reproduction*, *crossover*, and *mutation*, to create a new population of points. These operators are described below.

Reproduction: Reproduction is the first operator applied on a population. Reproduction selects good strings in a population and forms a mating pool. In doing so, it mimics the courtship phase of natural selection. There exist a number of reproduction operators in the GA literature, but the essential idea in all of them is that above-average strings are picked from the current population and their multiple copies are inserted in the mating pool in a probabilistic manner. The commonly used reproduction operator is proportionate reproduction, where a string is selected for the mating pool with a probability proportional to its fitness. This approach is also known as roulette wheel selection. Thus, good strings in a population are probabilistically assigned a larger number of copies and a mating pool is formed. Another approach to reproduction occurs in tournament selection. Here tournaments are arranged between any two random strings, and the winners are selected for mating. It is important to note that no new strings are formed in the reproduction phase.

Crossover: At the end of the reproduction process, the mating pairs are selected. The idea of exchange of genetic information which occurs between the male and female is now mimicked during the crossover process. Recall that each individual is a point or a binary string. Thus, information exchange involves the swapping of some bits between the male and female strings. The process of information exchange between the individuals is called crossover and is a basic property of GAs. The crossover procedure creates new chromosomes or strings from the two parents. Crossover is performed after selection of a subpopulation of individuals according to their fitness values and collection of the selected individuals into a gene pool. Crossover is achieved in three stages. The first stage is matching. Matching is the selection of two individuals in the gene pool randomly. In the second stage, a crossover point is determined in each of the individuals. In the final stage, two parts of the individuals are replaced with each other. Typically, the crossover performs an operation where two parents lead to two children. An example of crossover is shown below:

Parent Strings	Children Strings
100 | 1001	100 | 0111
==Crossover⟹	
111 | 0111	111 | 1001

Besides the single-point crossover shown above, multipoint crossover can also be used. In such a case, the parent strings between two sites are swapped to get the

child strings.

Parent Strings	Children Strings
100 | 1001 | 1001001	100 | 0111 | 1001001
==Crossover⟹	
111 | 0111 | 0101010	111 | 1001 | 0101010

Multipoint crossover is better when a large number of design variables result in long strings. For such long strings, single-point crossover is biased toward the right of the strings and changes these design variables much more often than those at the left of the string.

Mutation: Mutation means a random change in the information of a chromosome or string. In other words, mutation is an operation that defines the variation in a chromosome. This variation may be local or global. A probability test determines whether a mutation will be carried out or not. For example, if the average fitness of the new generation is smaller than the average fitness of the previous generation, bit y of the chromosome x can be changed. A bit mutation applied to a chromosome is shown below:

Before mutation: 100 | 1 | 1101

After mutation: 100 | 0 | 1101

Mutation can be useful for improving the population. Since the initial population is a subset of all possible solutions, an important bit of all the chromosomes may be 0 while it must be 1 to be optimal. Crossover may not solve this problem and mutation is indispensable for the solution. In general, mutation leads to small local moves in the design space, while crossover leads to larger global moves. Mutation may often lower the average fitness of the population, but it also allows the GA to escape from a local minimum by adding diversity.

There are two proposed methods for allowing a subpopulation to replace its ancestors. One of the methods is generational replacement. In this method, a population of size n entirely replaces the new generation. The other method is steady-state reproduction, which replaces only a few individuals in a generation. A small number of strings with high fitness values are sometimes shielded from the crossover and mutation operations. This is known as elitism.

2.2.2 Performance Factors

Population size is a factor that affects the performance of the GA. Increasing the population means a longer computation time. On the other hand, if the population size is decreased, the accuracy of the solution also decreases because of reduced variation of chromosomes. In GA design, there must be a balance between the generation numbers and population size. Population size has another effect in the GA; it

reduces the effect of the highest fitness valued chromosomes. For example, in a population of 10 chromosomes, if one of the chromosomes has a fitness value of 9 while the others have a fitness value of 1, half of the parents will be chosen from among the relatively low fitness valued chromosomes, although the best fitness valued chromosome is nine times better. Evaluation of chromosomes and fitness calculations are the most time-consuming parts of the GA. If the evaluation operation is reduced, the GA process will work faster, and this can be achieved by reducing the population size and number of generations needed to reach a solution.

2.3 Genetic Fuzzy System

The principles and operations of GAs and fuzzy logic have been briefly described in the previous sections. These two soft computing tools can be combined to form the genetic fuzzy system [9]. The GA provides good global search capability. Fuzzy logic presents robust and flexible inference methods in problems subject to imprecision and uncertainty. The linguistic representation of knowledge permits a person to interact with a fuzzy system in an easy manner. The hybridization of the GA and fuzzy logic gives an advanced soft computing method called the genetic fuzzy system (GFS), in which a GA is used to evolve a fuzzy system by tuning fuzzy membership functions and learning fuzzy rules.

A key objective of this section is to use GAs to automate the design of fuzzy systems. The generalized GFS algorithm is explained for a damage detection problem.

Input and Output Suppose inputs to the fuzzy systems are represented by $\mathbf{z}$ and outputs are represented by $\mathbf{x}$. The objective is to find the mapping between $\mathbf{z}$ and $\mathbf{x}$. In damage detection problems, the measurement deltas Δz's (changes in measurements between damaged and undamaged model) can be used as inputs, and the combinations of damage levels and locations will be the output of the fuzzy system. Here $\mathbf{x} = \{location_1, location_2, \ldots, location_n\}^T$ and $\mathbf{z} = \{\Delta z_1, \Delta z_2, \ldots, \Delta z_d\}^T$, where n are the user-defined locations and d is a user-defined number of measurements. Each measurement delta has uncertainty.

Fuzzification The structure can be divided into various locations. For example, "$location_1$" ranges from 0% to say n_1% and so on until all the locations are labeled per user requirement. To get a degree of resolution of the extent of damage [10], each of these damage locations is allowed several levels of damage and split into linguistic variables. For example, consider "$location_1$" as a linguistic variable. Then it can be decomposed into k levels and the undamaged level,

$$T(location_1) = \{Undamaged, level_1\ Damage, level_2\ Damage, \ldots, level_k\ Damage\}$$

where each term in $T(location_1)$ is characterized by a fuzzy set in the universe of discourse $U(location_1) = \{0, damage\ parameter\}$. The other structural damage variables are fuzzified in a similar manner.

The measurement deltas $\Delta z_1, \Delta z_2, \ldots$ and Δz_d are also treated as fuzzy variables. Fuzzy sets with Gaussian membership functions are used to define these input

variables. These fuzzy sets can be defined using the following equation:

$$\mu(x) = e^{-0.5(\frac{x-m}{\sigma})^2}, \tag{2.4}$$

where m is the midpoint of the fuzzy set.

Δz's are used as midpoints for the respective fuzzy sets. The standard deviation associated with each variable is denoted by σ and is obtained using a GA for maximization of the success rate S_R, as discussed later. After optimization of the fuzzy system for its success rate, we get a different σ value for every fuzzy set. We can see that the Gaussian fuzzy sets depend on the appropriate choice of the midpoint and the standard deviation. The midpoint is a measure of the point of maximum likelihood of a fuzzy set, while the standard deviation represents the scatter and accounts for the uncertainty. For instance, we could define a 0% change in the natural frequency as "negligible" and give a standard deviation of 0.3%. This ensures that a Gaussian fuzzy set for the word "negligible" is centered at 0 change in frequency representing the undamaged condition and also that very small shifts from the 0 position are also classified as negligible. Thus, the numerical value is fuzzified by the spread of the Gaussian fuzzy set, and the standard deviation allows us to control how slowly or fast the "negligible" membership function decays.

Rule Generation Rules for the fuzzy system can be obtained by fuzzification of the numerical values obtained by numerical analysis of the model using the following procedure.

1. The fuzzy sets corresponding to $\Delta z_1, \Delta z_2, \ldots, \Delta z_d$ are generated by taking the Δz's obtained by numerical analysis as midpoints of membership functions corresponding to a given structural damage. This strategy for selecting the midpoint ensures that the maximum degree of membership ($\mu = 1$) for each fuzzy set occurs at the values of Δz since the Gaussian function is highest at the midpoint. The standard deviation of each set is initially fixed randomly within a prescribed range.
2. For each measurement delta corresponding to given fault, the degree of membership in the fuzzy set is calculated.
3. Each measurement delta is assigned to the fuzzy set with the maximum degree of membership. One rule is obtained for each damage type by relating the measurement deltas. The "IF" rule segment before "THEN" part will change from structure to structure but the rule segment after "THEN" will remain the same unless the number of damage locations and levels of damages are changed. Suppose for rule i we are giving d inputs. Then the fuzzy system will generate d membership functions A taking the change in measurements obtained by numerical analysis of the model as the midpoint. The standard deviation is obtained by optimization as discussed later.
4. The fuzzy system is then fine tuned by changing the standard deviation σ for each set.

Table 2.1 Rules for fuzzy system

Rule	Δz_1	Δz_2	...	Δz_d
Undamaged	A_{11}	A_{12}	...	A_{1d}
$level_1$ Damage at $location_1$	A_{21}	A_{22}	...	A_{2d}
$level_1$ Damage at $location_2$	A_{31}	A_{32}	...	A_{3d}
⋮				
$level_1$ Damage at $location_k$	$A_{(k+1)1}$	$A_{(k+1)2}$	...	$A_{(k+1)d}$
⋮				
$level_n$ Damage at $location_k$	$A_{((n\times k)+1)1}$	$A_{((n\times k)+1)2}$	...	$A_{((n\times k)+1)d}$

So rule i will be in generalized form as:

IF
Δz_1 is A_{i1} AND
Δz_2 is A_{i2} AND
⋮
Δz_d is A_{id} AND
THEN

corresponding level of damage at corresponding location.

In the preceding rules, the membership values of membership function A will change from structure to structure. These rules are symbolically tabulated in Table 2.1.

For calculation of the uncertainty associated with variables, i.e., the standard deviation of the Gaussian membership functions, we use a GA for optimization of the success rate. As we have already discussed, there will be uncertainty and some noise in the measurement deltas. By generating noisy deltas and testing the fuzzy system for a known damage, we can define a success rate. Our optimization problem can be written in standard form as

$$\begin{aligned} \text{Maximize} \quad & S_R \\ \text{For design variable} \quad & \sigma^{\min} \leq \sigma_{ij} \leq \sigma^{\max} \\ & \text{where } i = 1, 2, \ldots, M, \\ & \qquad j = 1, 2, \ldots, d, \end{aligned}$$

where M is the number of rules and d is the number of fuzzy sets.

The success rate is calculated using the results obtained after defuzzification. If we test N_T samples of noisy data and out of that the system correctly classifies N_C times, then the success rate for rule p as a percentage is given as [11]

$$S_R = \frac{N_C}{N_T} 100. \tag{2.5}$$

The added noise in the data simulates the uncertainty present in the experimental measurements and the modeling process. Given a computed measurement delta Δz, random number u in the interval $[-1, 1]$, and a noise level parameter α, the noisy simulated data is given as

$$\Delta z^{\text{noisy}} = \Delta z + u\alpha. \tag{2.6}$$

The parameter α defines the maximum variance between the computed value of Δz and simulated measured value Δz^{noisy} which is a simulation of a practical measurement. For example, if $\alpha = 0.15$, then the simulated measurement delta can be different by as much as 0.15 from the ideal value predicted by simulations. Thus, α can be used to control noise levels in the simulated data used for testing the fuzzy logic system.

2.4 Summary

This chapter gives a brief introduction to fuzzy logic, genetic algorithms, and a few other terms used in the formulation of the genetic fuzzy system. Fuzzy logic involves computing with words instead of numbers and is robust to the presence of uncertainty in the inputs. However, fuzzy logic systems are difficult to design due to the interaction between the choice of appropriate membership functions, the rule base obtained, and the performance of the fuzzy system. The process of designing fuzzy systems can be automated by using an optimization procedure for tuning the membership functions and rule base. Genetic algorithms provide an excellent approach for finding the global minimum and can be used to design the fuzzy systems. The genetic fuzzy system combines the uncertainty representation characteristics of fuzzy logic with the learning ability of the genetic algorithm. Using the changes in measurement deltas, a fuzzy system is generated, and the rule base and membership functions are optimized by the genetic algorithm. The generalized formulation of the genetic fuzzy system is explained using a damage detection problem. The genetic fuzzy system will be used for solving damage detection problems using modal data in the next two chapters. These applications will make the process of developing genetic fuzzy systems for structural health monitoring very clear.

References

1. Kosko, B.: Fuzzy Engineering. Prentice-Hall, Englewood Cliffs (1997)
2. Ross, T.J.: Engineering Applications of Fuzzy Logic. Wiley, New York (2010)
3. Tanaka, K., Nimura, T.: An Introduction to Fuzzy Logic for Practical Applications. Springer, Heidelberg (1996)
4. Karr, C.L., Stanley, D.A., McWhorter, B.: Optimization of hydrocyclone operation using a geno-fuzzy algorithm. Comput. Methods Appl. Mech. Eng. **186**(2–4), 517–530 (2000)
5. Goldberg, D.: Genetic Algorithms in Search, Optimization, and Machine Learning. Pearson Education, Boston (2001)

6. Haupt, R.L., Haupt, S.E.: Practical Genetic Algorithms. Wiley Interscience, New York (2004)
7. Cordon, O., Herrera, F., Villar, P.: Generating the knowledge base of a fuzzy rule-based system by the genetic learning of the data base. IEEE Trans. Fuzzy Syst. **9**(4), 667–674 (2001)
8. Coello, C.A.C.: Theoretical and numerical constraint handling techniques used with evolutionary algorithms: a survey of the state-of-the art. Comput. Methods Appl. Mech. Eng. **191**, 1245–1287 (2002)
9. Cordon, O., Harrera, F., Hoffman, F., Magdalena, L.: Genetic Fuzzy Systems. World Scientific, Singapore (2002)
10. Ganguli, R.: A fuzzy logic system for ground based structural health monitoring of a helicopter rotor using modal data. J. Intell. Mater. Syst. Struct. **12**(6), 397–408 (2001)
11. Abe, S.: Pattern Classification: Neuro-Fuzzy Methods and Their Comparison. Springer, London (2001)

http://www.springer.com/978-0-85729-906-2

Structural Health Monitoring Using Genetic Fuzzy Systems

Pawar, P.M.; Ganguli, R.

2011, VIII, 132 p., Hardcover
ISBN: 978-0-85729-906-2