Multi-region System Modeling by using Genetic Programming to Extract Rule Consequent Functions in a TSK Fuzzy System

Yu Zhang, *Member, IEEE,* Miguel Martínez-García, *Member, IEEE*,
José R. Serrano-Cruz and Anthony Latimer

***Abstract*—This paper aims to build a fuzzy system by means of genetic programming, which is used to extract the relevant function for each rule consequent through symbolic regression. The employed TSK fuzzy system is complemented with a variational Bayesian Gaussian mixture clustering method, which identifies the domain partition –simultaneously specifying the number of rules and the parameters of the fuzzy sets. The genetic programming approach is accompanied with an orthogonal least square algorithm to extract robust rule consequent functions for the fuzzy system. The proposed method provides a flexible grey-box modelling strategy, which brings three main advantages: 1) it is suitable for estimating a system with varying behavior across the elements of a partition of its domain; 2) it can automatically determine the necessary number of rules, without searching for redundant rule information; and 3) it only requires small data sets, in contrast with other machine learning approaches such as deep learning. The proposed model is validated with a synthetic surface, and then with real data from a gas turbine compressor map case, which is compared with an adaptive neuro fuzzy inference system model. The results have shown the adequateness of the proposed method for modelling systems with bifurcating dynamics, and where the analytical equations are not available, such as those in a typical industrial setting.**

I. Introduction

Machine learning, besides concentrating on data feature representation, should contribute to the interpretation of cognitive process – by replicating natural reasoning patterns – and hence sharing the motivation behind the classical symbolic *artificial intelligence* methodologies [1][2]. For instance, the artificial intelligence subdomain of *fuzzy inference systems,* aims at exposing new ways of performing human-like reasoning by encoding fuzzy inferential relations among symbols, and intends to solve real-world problems where the inferential boundaries of classical logic are too crisp to represent reality [3]. Hence fuzzy systems bridge the domains of symbolic artificial intelligence and machine learning into computa-tionally efficient yet bio-inspired methodologies; by an introduction of vagueness, fuzzy systems are suitable for training small data sets, described by the degree of fuzzy membership in the pre-defined complete set [4]. This has gained favor especially in industrial applications, where often limited data are available [5][6], and brings a significant advantage over machine learning techniques, such as deep learning, which normally require big data for training the models, and can essentially be regarded as black-box systems.

Other than being able to form inferential patterns from relatively small data-sets, human intelligence also surpasses artificial intelligence in that humans can more effectively perform extrapolation from existing knowledge. Traditional machine learning is generally more efficient at interpolating data, since it learns to fit, and every so often over-fit, the given data, disregarding its performance outside the measured regions – thus reducing the prediction accuracy [7]. In this context, fuzzy systems assist by introducing fuzziness outside the numerically explored regions. But, their hyper-parameters and estimation parameters, such as the membership functions and the rule consequent functions, etc., should also be representative for real-world applications, providing generic descriptions of the principal mechanisms. Fuzzy models are compared with the classical method of linear regression in [8], where they were shown to perform better when extrapolating from the given data.

There are two primary types of fuzzy inference systems – Mamdani [9] and Sugeno [10] inference types. Herein, fuzzy systems of the Sugeno type are considered – also referred as the TSK models [10]. They differ from the Mamdani models, in that each rule's consequent is a mathematical function in the TSK models, whereas the output membership functions of the Mamdani models are represented by fuzzy sets. The TSK models have several advantages, such as being compatible with other numerical analysis procedures, e.g. optimization algorithms, and maintaining the continuity of the output surface. Typically, the TSK model output membership functions are chosen as a constant or a linear function of the input. However, the model estimation result can be biased because of the specific constraint on function structure. Here, the method is extended to include generic machine learned functions for the rule consequent part, where both the function structure and the function parameters are optimized by an evolutionary computation technique.

Evolutionary computation is originated from genetic algorithms and intends to mimic the evolutionary processes found in nature, where an organism adapts to the environment [11]. Commonly it is used to solve complex optimization problems. *Genetic programming* is an extension of genetic algorithms, where the population members at each iteration are executable programs, represented as trees instead of strings of

* Research supported by EPSRC Grant EVES (EP/R029741/1).

Y. Zhang is with the School of Engineering, University of Lincoln, Lincoln, LN6 7TS, UK (corresponding author, phone: +44-1522-837938; e-mail: yzhang@ lincoln.ac.uk).

M. Martínez-García is with the School of Engineering, University of Lincoln, Lincoln, LN6 7TS, UK (e-mail: mmartinez@ lincoln.ac.uk).

J. R. Serrano-Cruz is with the School of Design Engineering, Universitat Politècnica de València, 46022 Valencia, Spain (e-mail: jrserran@ mot.upv. es)

A. Latimer is with the Siemens Industrial Turbomachinery Ltd., Lincoln, LN6 3AD, UK (e-mail: anthony.latimer@siemens.com).

numerical quantities [12]. Genetic programming is able to perform symbolic regression, which makes the model or model hyper-parameter selection process extremely flexible, which may generate a more adequate model structure than the default one [13]. Although it is useful for the model structure selection, e.g. the functions that conform the model and their orders, genetic programming cannot be applied directly for nonlinear system identification because of the estimation of the model parameters. To tailor genetic programming for this purpose, *Madar et al.* [14] have introduced a hybridization of genetic programming and the *orthogonal least squares* technique for concurrent model structure selection and its parameters for nonlinear but linear-in-parameters models, which can provide more meaningful results of the fitted functions by eliminating the insignificant terms.

Here, a fully automated TSK fuzzy model is designed. In the rest of the paper, the underlying principles of the utilized techniques are described briefly in Sec. II. Two case studies are provided, using a synthetic multi-region surface and a real gas turbine compressor map separately, in Sec. III. The results are compared with the well-known Adaptive Neuro Fuzzy Inference System (ANFIS) model [15]. Finally, Sec. IV concludes the paper, and identifies the future work.

II. Underlying Principles

A. *Fuzzy Inference System*

An example TSK fuzzy system, with two input variables ***x*** and ***y*** and one output ***z***, is presented in Fig. 1. The input fuzzy sets are A_1 and A_2 for ***x***, and B_1 and B_2 for ***y***. In the example, there are two rules, and the *firing strength* of each rule can be determined either by the minimum or by the product of the input membership values. Here, the product operator is used. The final output ***z*** is computed by taking a weighted average of the two consequent function outputs z_1 and z_2, where the weights are the resulted firing strengths w_1 and w_2.

The proposed method here is called VEFIS – representing the combination of the Variational Bayesian technique and the Evolutionary computation method, integrated in a Fuzzy Inference System. The proposed model is based on the structure described in Fig. 1, where the premise fuzzy sets and the number of rules are determined using the variational Bayesian Gaussian mixture model [16]. Then, the consequent functions f_n are determined using a hybrid genetic programming-orthogonal least squares approach. The involved techniques are described in the following sub-sections.

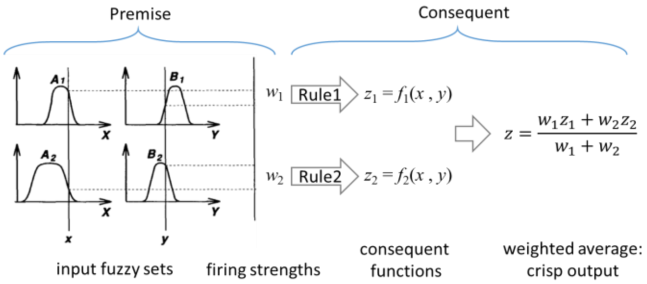


Figure 1. An example TSK fuzzy system architechture.

B. *Variational Bayesian Gaussian Mixture Model*

A *Gaussian mixture model* approximates the probability density function of sample data *x* as a linear combination of K Gaussian distributions with mean μ_k and standard deviation σ_k, denoted as

$$p(x|\boldsymbol{\theta}, \boldsymbol{\mu}, \boldsymbol{\sigma}) = \sum_{k=1}^{K} \theta_k \mathcal{N}(x|\mu_k, \sigma_k) \,, \qquad (1)$$

where θ_k are the mixing coefficients.

For each data sample x_n, where $\boldsymbol{x} \equiv \{x_n\}$, there exists a set of values $Z_{nk} \in \{0,1\}$, indicating whether the sample belongs to the k-th mixture component (as 1) or not (as 0). Normally the number of clusters K is pre-determined, but with the variational Bayesian method, K is obtained as an unknown by solving Z_{nk} as following.

First, the joint probability of all variables in the mixture model is given by

$$p(\boldsymbol{x}, \mathbf{Z}, \boldsymbol{\theta}, \boldsymbol{\mu}, \boldsymbol{\sigma}) = p(\boldsymbol{x}|\mathbf{Z}, \boldsymbol{\mu}, \boldsymbol{\sigma}) p(\mathbf{Z}|\boldsymbol{\theta}) p(\boldsymbol{\theta}) p(\boldsymbol{\mu}|\boldsymbol{\sigma}) p(\boldsymbol{\sigma}), \quad (2)$$

where $p(\boldsymbol{\theta})$, $p(\boldsymbol{\mu}|\boldsymbol{\sigma})$ and $p(\boldsymbol{\sigma})$ are established empirical distributions. Marginalizing (2) with respect to $\mathbf{Z}$, $\boldsymbol{\mu}$ and $\boldsymbol{\sigma}$ is difficult to achieve, thus it is convenient to apply a variational Bayesian framework on $p(\boldsymbol{x}|\boldsymbol{\theta}) = p(\boldsymbol{x}|\mathbf{Z}, \boldsymbol{\mu}, \boldsymbol{\sigma}) p(\mathbf{Z}|\boldsymbol{\theta})$.

Then, denoting $\boldsymbol{\Psi} \equiv \{\mathbf{Z}, \boldsymbol{\mu}, \boldsymbol{\sigma}\}$, the relation holds

$$p(\boldsymbol{x}|\boldsymbol{\theta}) = \int p(\boldsymbol{x}, \boldsymbol{\Psi}|\boldsymbol{\theta}) \, d\boldsymbol{\Psi} \,, \qquad (3)$$

where $p(\boldsymbol{\Psi}|\boldsymbol{\theta})$ can be approximated by a variational distribution, i.e. $q(\boldsymbol{\Psi}) \approx p(\boldsymbol{\Psi}|\boldsymbol{\theta})$. Here, the Kullback-Leibler divergence of p from q is applied – minimizing the KL divergence is to make q as close as p, which can be achieved by choosing suitable q distributions. For further simplification, $q(\boldsymbol{\Psi})$ can be considered to factorize over its subsets, i.e.

$$q(\mathbf{Z}, \boldsymbol{\mu}, \boldsymbol{\sigma}) = q_{\mathrm{Z}}(\mathbf{Z}) q_{\mu}(\boldsymbol{\mu}) q_{\sigma}(\boldsymbol{\sigma}). \qquad (4)$$

Consequently, the mixture model can be solved through the standard Expectation-Maximization (E-M) procedure. It includes iterative updating of the variational factors through the E-step until convergence to obtain q_i, and then the M-step for the mixing coefficients θ_k can also be achieved [17].

C. *Genetic Programming with Orthogonal Least Squares*

Genetic programming operates by evolving hierarchical computer programs from an initial population. The programs are represented as a tree structure, consisting of functions (operators) and terminals (variables and numbers), which are drawn from a pre-determined set of functions and terminals. Fig. 2 shows an example of a tree structure of genetic programming for the function (1 + *x* + (3 / *x*) + (*y* × 5)). The members of the population are then evolved through crossover and mutation operations, similar to those in the genetic algorithms.

The mutation mechanism in genetic programming is to delete one node including its children, and replace it with a randomly generated subtree. The crossover operation exchanges two randomly chosen subtrees between the parents. The general procedure includes: (i) initializing the population; (ii) calculating the fitness for each individual; (iii) reproducing selected individuals to form new population; (iv) performing crossover and mutation operations; and (v) iterating steps ii) to iv) until convergence.

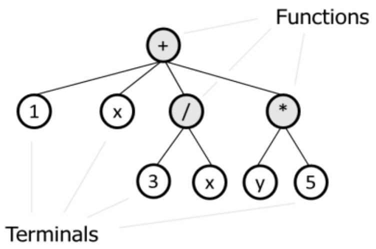


Figure 2. An example tree structure in genetic programming

In genetic programming, the tree size tends to increase over time, which can cause computational overflow. In order to provide an algorithmically efficient solution that does not produce excessive over-fitting and over-parametrizing, an orthogonal least squares algorithm can be employed. This method combines with genetic programming to eliminate the complex and unnecessary terms for the case of *linear-in-parameter* models [14].

A linear-in-parameters model is written as

$$\boldsymbol{o} = \mathbf{F}\boldsymbol{p} + \boldsymbol{e} \ , \tag{5}$$

where $\boldsymbol{p}$ is an M×1 parameter vector corresponding to the M terms of functions, $\mathbf{F}$ is an N×M regression matrix, $\boldsymbol{o}$ is the measured 1×N output vector, and $\boldsymbol{e}$ is the error vector.

The orthogonal least squares algorithm assumes that $\mathbf{F}$ can be orthogonally decomposed as $\mathbf{F} = \mathbf{WA}$, where $\mathbf{A}$ is an M×M upper triangular matrix, and $\mathbf{W}$ is an N×M matrix, which satisfies $\mathbf{W}^{\mathrm{T}}\mathbf{W} = \mathbf{D}$ (diagonal). The auxiliary parameter vector is introduced as $\boldsymbol{g} = \mathbf{D}^{-1}\mathbf{W}^{\mathrm{T}}\boldsymbol{o}$. Then, the output variance can be described as

$$\boldsymbol{o}^{\mathrm{T}}\boldsymbol{o} = \textstyle\sum_{i=1}^{\mathrm{M}} g_i^2 w_i^{\mathrm{T}} w_i + \boldsymbol{e}^{\mathrm{T}}\boldsymbol{e} \ , \tag{6}$$

and then the error reduction ratio $[err]_i$ for each term F$_i$ is

$$[err]_i = \frac{g_i^2 w_i^{\mathrm{T}} w_i}{o^{\mathrm{T}} o} \quad . \tag{7}$$

In this way, the order of the function terms F$_i$ can be determined according to their participation to the error reduction ratios in a linear-in-parameters model.

The concept of the hybrid genetic programming-orthogonal least squares approach can be summarized in the following steps: (i) the model is decomposed to the function terms (subtrees); (ii) the error reduction ratios of the function terms are computed according to (7); (iii) the terms that yield negligible error reduction are eliminated, e.g. 0.05 as threshold used in this paper; and (iv) the function terms' coefficients are fitted by a least squares method.

III. Case Studies

A. *Generated Surface*

First, the proposed model is tested with a synthetic piecewise surface. The piecewise surface has no discontinuity, and is conformed by four connected regions defined by the relations in (8). In each region, 100 points are randomly sampled as the testing data points. From the testing data, the variational Bayesian Gaussian mixture model is applied to determine the cluster number (K = 4). An example of the conforming clustered points are presented in Fig. 4, projected on the *x*-*y* plane for clarity.

$$z(x,y) = \begin{cases} \sin x^2 \cos y & -2 \le x \le 0, -2 \le y \le 0 \\ \sin x \quad \cos y^2 & 0 \le x \le 2,\ 0 \le y \le 2 \\ \sin x^2 \cos y^2 & -2 \le x \le 0,\ 0 \le y \le 2 \\ \sin x \quad \cos y & 0 \le x \le 2, -2 \le y \le 0 \end{cases} \tag{8}$$

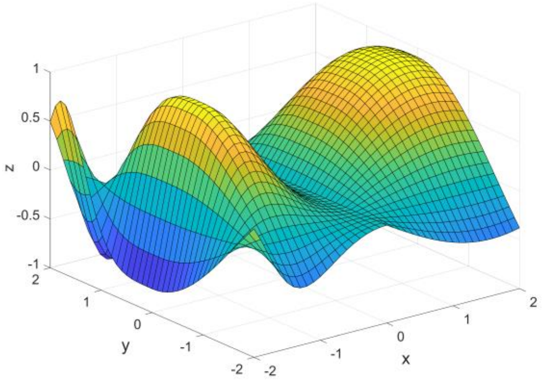


Figure 3. Synthetic surface for testing

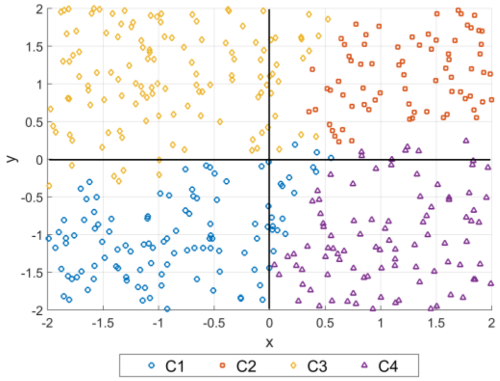


Figure 4. Clustering (C) results for the synthetic surface using the variational Bayesian Gaussian mixture model

The results do not exhibit a clear separation of the four surfaces due to smooth transitioning. However, each cluster contains most of the information of the underlying functionalities in that region. Then, the four clusters of data points are fitted separately using genetic programming. For approximation purposes, the set of operators in this case is selected as $F = \{+, \times\}$ only, i.e. higher order polynomial fitting. Table 1 shows the parameters of genetic programming, and it is trained for 500 epochs. The four corresponding functions after searching are

$f_1(x, y) = 0.1973\ (4\ x^2 + x^2 \times (x + y) + x \times (x^2 \times (x^3 + 3\ y) + 3\ x + y \times (3\ x + y))) + 0.5612\ (x^5 + y) + 0.0145\ x - 0.5382\ y + 0.06948;$

$f_2(x, y) = 0.2358\ (y \times (x + y) + y + 3\ x) + 0.00912\ (((y^3 + x \times y^3) + y) \times y^4 + x \times y) - 0.1612\ (3\ y^2 \times (x + y) + 2\ x) + 0.2303;$

$f_3(x, y) = 0.9884\ (x + y) + 0.1364\ ((x^2 \times y^2 + y) \times x^2 + x) + 0.3303\ (x + x \times y^2 + x \times (x + y) + (x + y) \times y \times x) - 0.5539\ (x^4 + x^2 + 2\ y\ + x \times y) - 1.4028\ x - 0.4925\ ((x + y) \times 2\ x^2 + x \times y + x\ ) + 0.1584;$

$f_4(x, y) = 0.5201\ x + 0.3603\ (x \times y + x) - 0.1046\ (y + (x + y) \times x^2 + (x \times y + y) \times y) + 0.04176.$

TABLE I. PARAMETERS OF GENETIC PROGRAMMING

Population size	40
Selection method	Roulette wheel
Crossover method	One point crossover
Mutation method	Point mutation
Replacement method	Elitist
Generation gap	0.8
Probability of crossover	0.7
Probability of mutation	0.5
Evaluation	All individuals

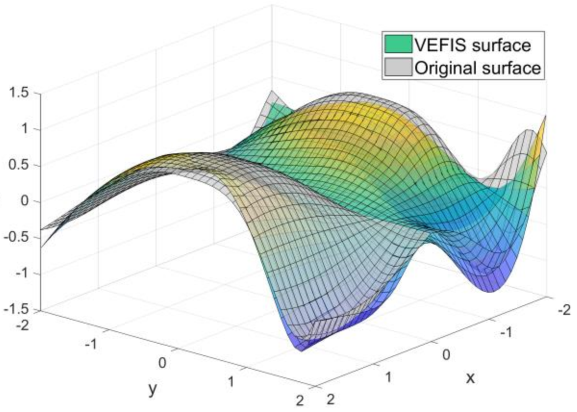


Figure 5. Generated surface from the VEFIS model compared with the reference surface

It is shown that the simpler function, e.g. for cluster 4, is fitted with a simpler structure of functions, while the more complex variations, e.g. in cluster 3, are captured by more compound polynomial terms.

The generated surface based on the proposed VEFIS model and the original synthetic surface are compared in Fig. 5. It is shown that the highly varying characteristics of the different regions are approximated relatively well by the simple fitting technique based on the proposed scheme. This case study serves as a simple demonstration of the proposed method. It is noticed that, if 'sin' and 'cos' are included in the selected function set, the precise original surface can be retrieved.

B. Compressor Map Estimation

The proposed model is now tested with real-world industrial data – a gas turbine compressor map. Adequate component maps, including the compressor map studied here, are important for use in gas turbine performance modelling and control [18]. Generally, engine tests can only produce limited data due to control limit and cost constraints. Estimation of the performance characteristics of the gas turbine component, which can provide both interpolation and extrapolation to complete the component maps, becomes therefore important.

Polynomial fitting and artificial neural networks are commonly used as compressor map prediction methods [19][20]. A single polynomial was used to describe the entire compressor map in [21]. Coefficients of different polynomials on related characteristics were computed by using genetic algorithms in [22]. And, different types of artificial neural networks were used and compared for modeling compressor maps in [23]. However, as mentioned in Sec. I, constraint of the selected function structure, e.g. the polynomials, can result in bias in the model prediction; and artificial neural networks are of a black-box nature, which may lead to overfitting within the known region, and significantly biased for extrapolation.

The data used in this paper are based on gas turbine measurements from NASA, which are publicly accessible in several reports [24]. There are ten speed lines, and each speed line contains seven engine test data points, as shown in Fig. 6. Based on the proposed method, first, the variational Bayesian technique is applied to the 70 data points. In this case, the determined cluster number K = 2, and the resultant clustered points are presented in Fig. 7, projected on the input 1-2 space for clarity.

To validate the proposed VEFIS model, the obtained results are compared with the results obtained using the Matlab ANFIS toolbox. For comparison purpose, the ANFIS model is also designed with two Gaussian membership functions for each input, and trained for 500 epochs.

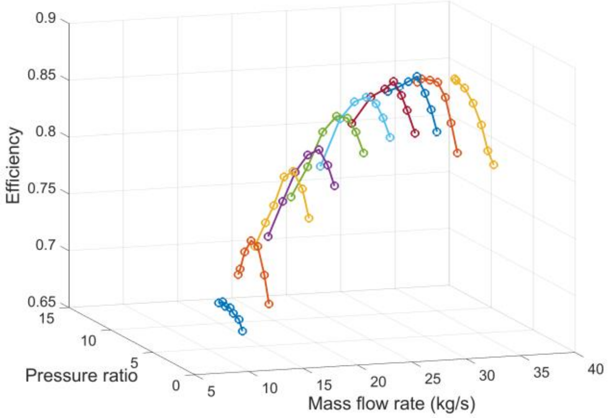


Figure 6. Compressor map

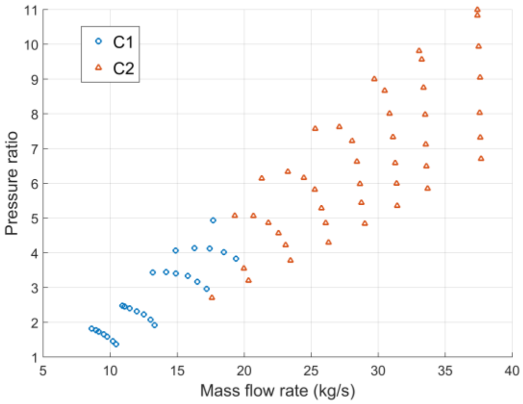


Figure 7. Clustering (C) results for the compressor map

The parameters of genetic programming are the same as listed in Table 1. The set of operators in this case is set to be F = {+, −, ×, /, log}, where '/' and 'log' are protected for numerical stability. The genetic programming is trained for the same number of epochs as before, i.e. 500.

The VEFIS output functions are:

$f_1(x, y) = 0.3436\ (x - 2\ y \times y\ /\ x) - 0.3642\ (x - y) - 0.06184\ (y\ /\ x + y) + 0.6077$;

$f_2(x, y) = -\ 0.05123\ ((1\ /\ y) - (x + 2\ y)) - 0.000434\ (x \times x \times \log(y)\ /\ \log(x)) + 0.002168\ (y \times y) - 0.1826\ y + 0.4726$.

And the ANFIS output functions are:

$f_1(x, y) = 0.003207\ x + 0.03015\ y + 0.2312$;

$f_2\ (x, y) = -\ 0.0163\ x + 0.0304\ y + 1.193$,

where x and y are the two inputs, mass flow rate and pressure ratio, separately. The optimized structures of functions based on genetic programming are more complex than those obtained by linear regression or polynomial fitting.

According to the results, the root-mean-square error based on the ANFIS is 0.0101, and the root-mean-square error based on the VEFIS is 0.0078, compared with the original 70 data points. The model prediction results are displayed in Fig. 8. It is shown that the VEFIS method captures better the nonlinear characteristics of the compressor map, especially for the low-speed lines, where usually the dynamics are more difficult to model with normal thermodynamics formulas, because their average behavior is more affected by perturbations.

The membership functions for the two inputs after ANFIS tuning and those based on the VEFIS method are illustrated in Fig. 9, where it is shown that, in both models, the fuzzy sets for Input 1 are similar, but they are significantly different for Input 2 in this case study. The VEFIS puts more weight on fitting low pressure ratio values, which correspond to low speed lines, justifying the results in Fig. 8 also.

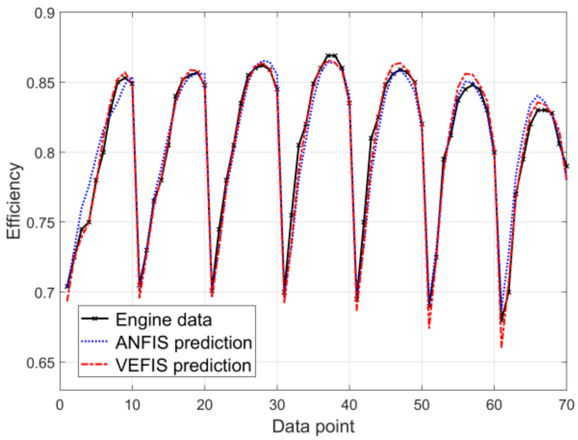


Figure 8. Model prediction results based on the proposed VEFIS method, compared with the original data and those from the ANFIS model

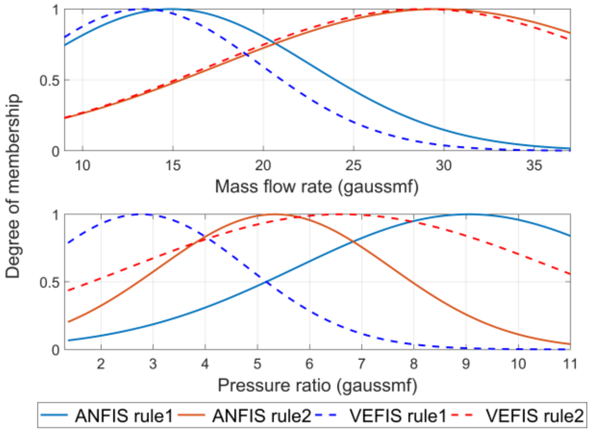
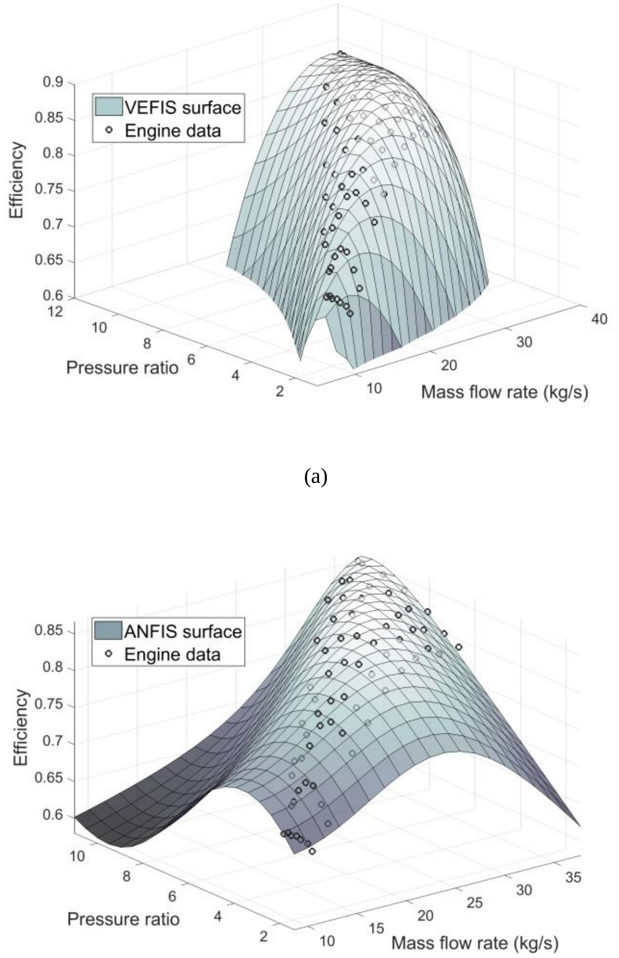


Figure 9. Gaussian membership functions for Input 1: mass flow rate and for Input 2: pressure ratio based on ANFIS and VEFIS models

The generated surfaces from the VEFIS model and from the ANFIS model are compared in Fig. 10. It is shown that the VEFIS model better represents the relations at lower speeds, in agreement with the results in Fig. 8. The compressor map estimation from Fig. 10(a) is qualitatively in agreement with the surface generated from a 4th order polynomial fitting of the compressor maps in [25]. This can provide a more accurate extrapolation capacity where the performance beyond stall and choke lines - on each side - has dropped significantly as compared with real conditions, which the ANFIS model could not capture as well (Fig. 10(b)).

IV. Conclusion

This paper has proposed a system modeling approach, based on a TSK fuzzy inference system framework, integrated with variational Bayesian Gaussian mixture model and genetic programming techniques. The variational Bayesian method is used to perform automatic Gaussian mixture clustering, which simultaneously gives the number of rules (clusters) and the fuzzy sets for the input variables (according to the centers and standard deviations of the Gaussian mixtures), so that the redundant rules have been disregarded since the initial clustering stage. Then, for each rule, the consequent functions are extracted using a hybrid genetic programming-orthogonal least squares method. This provides a more flexible solution when the characteristics of the system are difficult to model using the conventional linear functions such as those in the ANFIS model. Finally, the TSK fuzzy system is built to provide smooth transitions across different regions. The proposed modelling technique is tested with a synthetic surface, and then with an industrial compressor map case, where only small data are available. It has shown comparable performance to the well-known ANFIS model for the tested case. In future work, an integrated optimization strategy will be investigated for the proposed VEFIS framework, to take into consideration the coupled effect between each of the training stages – i.e. by fitting all the VEFIS parameters simultaneously.



(a)

(b)

Figure 10. Generated surface from (a) the VEFIS model and (b) the ANFIS model based on the compressor map data

References

[1] J. Haugeland, *Artificial Intelligence: The Very Idea*. MIT Press, Cambridge, MA, 1985.

[2] H. Prade, "Reasoning with data - a new challenge for AI?" In *Proc. Int. Conf. Scalable Uncertainty Management*, Nice, France, 2016.

[3] L. A. Zadeh, "Fuzzy sets as a basis for a theory of possibility," *Fuzzy Sets and Systems*, vol. 1, pp. 3–28, 1978.

[4] U. Mönks, D. Petker and V. Lohweg, "Fuzzy-pattern-classifier training with small data sets," In *Proc. Int. Conf. Information Processing and Management of Uncertainty in Knowledge-Based Systems*, Dortmund, Germany, 2010.

[5] Y. Zhang, J. Chen, C. Bingham and M. Mahfouf, "A new adaptive Mamdani-type fuzzy modeling strategy for industrial gas turbines," In *Proc. IEEE Int. Conf. Fuzzy Systems*, Beijing, China, 2014.

[6] C. Li, W. Zou, N. Zhang and X. Lai, "An evolving T–S fuzzy model identification approach based on a special membership function and its application on pump-turbine governing system," *Engineering Applications of Artificial Intelligence*, vol. 69, pp. 93-103, 2018.

[7] S. S. Sahoo, C. H. Lampert and G. Martius. "Learning equations for extrapolation and control," In *Proc. 35th Int. Conf. Machine Learning*, Stockholm, Sweden, 2018.

[8] C. Tzimopoulos, L. Mpallas and C. Evangelides, "Fuzzy model comparison to extrapolate rainfall data," *J. Environmental Science and Technology*, vol. 1, pp. 214-224, 2008.

[9] E.H. Mamdani and S. Assilian, "An experiment in linguistic synthesis with a fuzzy logic controller," *Int. J. Man-Machine Studies*, vol. 7, no. 1, pp. 1-13, 1975.

[10] M. Sugeno and G. Kang, "Structure identification of fuzzy model," *Fuzzy Sets and Systems*, vol. 28, pp.329-346, 1986.

[11] J. H. Holland, *Adaptation in Natural and Artificial Systems*, Ann Arbor: Univ. of Michigan Press, 1975.

[12] J. R. Koza, *Genetic Programming – on the Programming of Computers by means of Natural Selection*. Cambridge MA, USA: The MIT Press. 1992.

[13] A. Bastian, "Identifying fuzzy models utilizing genetic programming," *Fuzzy Sets and Systems*, vol. 113, no. 3, pp. 333-350, 2000.

[14] J. Madár, J. Abonyi and F. Szeifert, "Genetic programming for the identification of nonlinear input−output models," *Industrial & Engineering Chemistry Research*, vol. 44, no. 9, pp. 3178-3186, 2005.

[15] J.S.R. Jang, "ANFIS: adaptive-network-based fuzzy inference system," *IEEE Trans. Systems, Man and Cybernetics*, vol. 23, no. 3, pp. 665–685, 1993.

[16] Y. Zhang, M. Martinez-Garcia and A. Latimer, "Estimating gas turbine compressor discharge temperature using Bayesian neuro-fuzzy modelling," In *Proc. IEEE Int. Conf. Systems, Man and Cybernetics*, Banff, Canada, 2017.

[17] A. Corduneanu and C.M. Bishop, "Variational Bayesian model selection for mixture distributions," In *Proc. 8th Int. Conf. Artificial Intelligence and Statistics*, pp. 27-34, 2001.

[18] A. Chaibakhsh and S. Amirkhani, "A simulation model for transient behaviour of heavy-duty gas turbines," *Applied Thermal Engineering*, vol. 132, pp. 115-127, 2018.

[19] M. Zagorowska and N. F. Thornhill, "Compressor map approximation using Chebyshev polynomials," In *Proc. IEEE 25th Mediter. Conf. Control and Automation*, pp. 864-869, Malta, 2017.

[20] E. Tsoutsanis, N. Meskin, M. Benammar and K. Khorasani, "A component map tuning method for performance prediction and diagnostics of gas turbine compressors," *Applied Energy*, vol. 135, pp. 572-585, 2014.

[21] F. K. Moore and E. M. Greitzer, "A theory of post-stall transients in axial compression systems: Part I—development of equations," *J. Eng. Gas Turbines Power*, vol. 108, pp. 68–76, 1986.

[22] C. D. Kong, S. Kho and J. Y. Ki, "Component map generation of a gas turbine using genetic algorithms," *J. Eng. Gas Turbines Power*, vol. 128, no. 1, pp. 92–96, 2006.

[23] K. Ghorbanian and M. Gholamrezaei, "An artificial neural network approach to compressor performance prediction," *J. Appl. Energy*, vol. 86, no. 7, pp. 1210–1221, 2009.

[24] G. L. Converse, and R. G. Giffin, "Extended parametric representation of compressor fans and turbines," *CMGEN User's Manual, NASA–CR–174645*, vol. 1, 1984.

[25] H. Tu and H. Chen, "Modeling of a compressor's performance map by fitting function methodology," *Advanced Materials Research*, vol. 779-780, pp. 1194-1198, 2013.