Nuclear Engineering and Technology 52 (2020) 287–295

Contents lists available at ScienceDirect

Nuclear Engineering and Technology

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

Original Article

Analyzing nuclear reactor simulation data and uncertainty with the group method of data handling

Majdi I. Radaideh*, Tomasz Kozlowski

Department of Nuclear, Plasma, and Radiological Engineering, University of Illinois at Urbana Champaign, Urbana, IL, 61801, USA

ARTICLE INFO

Article history:
Received 30 January 2019
Received in revised form
24 July 2019
Accepted 24 July 2019
Available online 27 July 2019

Keywords:
Uncertainty quantification
GMDH
Surrogate modeling
Deep learning
Reactor simulations

ABSTRACT

Group method of data handling (GMDH) is considered one of the earliest deep learning methods. Deep learning gained additional interest in today's applications due to its capability to handle complex and high dimensional problems. In this study, multi-layer GMDH networks are used to perform uncertainty quantification (UQ) and sensitivity analysis (SA) of nuclear reactor simulations. GMDH is utilized as a surrogate/metamodel to replace high fidelity computer models with cheap-to-evaluate surrogate models, which facilitate UQ and SA tasks (e.g. variance decomposition, uncertainty propagation, etc.). GMDH performance is validated through two UQ applications in reactor simulations: (1) low dimensional input space (two-phase flow in a reactor channel), and (2) high dimensional space (8-group homogenized cross-sections). In both applications, GMDH networks show very good performance with small mean absolute and squared errors as well as high accuracy in capturing the target variance. GMDH is utilized afterward to perform UQ tasks such as variance decomposition through Sobol indices, and GMDH-based uncertainty propagation with large number of samples. GMDH performance is also compared to other surrogates including Gaussian processes and polynomial chaos expansions. The comparison shows that GMDH has competitive performance with the other methods for the low dimensional problem, and reliable performance for the high dimensional problem.

1. Introduction

After the sharp growth in deep learning development, researchers started to utilize deep networks for many applications such as pattern recognition, big data analysis, signal processing, text generation, automatic game playing, and many other applications [1,2]. Deep learning is a form of machine learning that relies on multiple processing layers to learn representations of big datasets with abstract and complex structure. Therefore, deep learning networks are expected to avoid the curse of dimensionality, which is a major problem for large-scale models with complex structure and large number of inputs [3–5].

Uncertainty Quantification (UQ) and Sensitivity Analysis (SA) are broad areas in scientific computing to assess the performance of mathematical and engineering models [6,7]. Most of the UQ and SA tasks require several runs of a computer model that can be computationally demanding. A substitute called surrogate model, metamodel, or reduced order model (ROM) is constructed based on training data from the original model, and it can be used instead of the original model for UQ and SA [8,9]. Training surrogates is performed using machine learning methods including but not limited to: Kriging or Gaussian processes [10,11], radial basis functions [12], neural networks [13], polynomial chaos expansions [14], and many others. The curse of dimensionality causes many of the surrogate modeling methods to perform poorly when the number of input features, number of output variables, and/or number of data samples is high. Deep learning demonstrated good performance in such cases. For example, a high dimensional UQ framework was developed in Ref. [15] through learning deep neural networks (DNN) with parameterization done through recovering a low-dimensional nonlinear active subspace. In a recent research, a Bayesian fully convolutional encoder–decoder network for surrogate modeling and UQ was introduced by Ref. [16], with an application to stochastic partial differential equations with high dimensional nature.

Group method of data handling (GMDH) networks [17,18] are considered one of the earliest deep learning methods by some authors [19]. GMDH has been developed with an aim of parametric optimization and mathematical modeling of complex systems. The

* Corresponding author.
E-mail address: radaide2@illinois.edu (M.I. Radaideh).

https://doi.org/10.1016/j.net.2019.07.023

GMDH was revisited and developed by S.J. Farlow in Ref. [20]. A review by Ref. [21] highlighted the problems that can be solved by GMDH such as pattern recognition, clustering, prediction, and many others. GMDH networks are created based on special polynomial activation functions called Kolmogorov–Gabor polynomials, which are widely used to construct general non-linear models [22]. Given a training data set, GMDH layers are incrementally grown and trained by regression methods. External validation criterion is used to optimize the neurons within the hidden layers to stop training. GMDH was utilized in many applications such as semiconductor manufacturing [23], recognition of medical image of blood vessels [24], fault identification [25], fluid flow simulations [26], and many more. For nuclear applications, GMDH was used for different purposes such as predicting the break size in loss of coolant accidents [27], heat transfer applications [28], reactor power shaping and sensing [29,30], and monitoring and fault diagnosis in steam generators and thermal systems [31,32].

High dimensionality in nuclear reactor modeling is a known problem, especially in multiphysics and coupled simulations, which require surrogate models with good performance in high dimensional spaces. The goal of this study is to introduce GMDH as an example of deep learning in the context of UQ of reactor simulations, as utilizing GMDH for UQ is limited in the literature. The GMDH approach is adopted in this study to construct deep networks to capture the input-output relationship. Important aspects of constructing GMDH networks such as number of neurons, polynomial activation function, validation criteria, etc. are discussed and explored. Comparison of GMDH performance with other common machine learning methods is conducted including Gaussian processes and polynomial chaos expansions. Two case studies in reactor physics and thermal-hydraulics simulations are used to investigate GMDH performance. The first case is a low dimensional problem of two-phase flow in a boiling water reactor (BWR) channel. The second case is a high dimensional problem of the 8-group homogenized neutron cross-sections to predict the lattice reactivity (k_∞). Afterward, GMDH networks are utilized to perform SA and UQ tasks such as variance decomposition and uncertainty propagation. Conclusions and observations about the applied methods are drawn and discussed based on the results. The remaining sections of this paper are organized as follows: section 2 discusses the theory, the methodology, and the applications utilized in this study. Learning GMDH networks is described first, followed by a brief description of the other metamodelling methods used for comparison. Afterward, descriptions of the validation metrics and applications used to assess all methods are presented. Section 3 presents the results obtained from this study and their discussion, followed by the conclusions in section 4.

2. Methodology and applications

In this section, the GMDH method implemented in this study is described in detail, followed by a brief description of the other metamodelling techniques used for comparison with GMDH performance. The validation metrics used to evaluate the constructed models as well as the reactor applications used in this study are described next.

2.1. GMDH

2.1.1. Principles of GMDH

GMDH is an iterative multi-layer approach trained by regression methods [18,20]. The fundamental difference between GMDH and other linear/nonlinear regression methods is that it offers an efficient optimization and search for the best model representing the relationship between the input-output. This means that some input parameters can be eliminated during the search process in GMDH, which can be valuable for complex and high dimensional problems. GMDH constructs a high-order polynomial called Kolmogorov–Gabor, which has the following form [17,18,20].

$$Y^{GMDH} = a_0 + \sum_{i=1}^{d} a_i x_i + \sum_{i=1}^{d}\sum_{j=1}^{d} a_{ij} x_i x_j + \sum_{i=1}^{d}\sum_{j=1}^{d}\sum_{k=1}^{d} a_{ijk} x_i x_j x_k + \ldots, \tag{1}$$

where Y^{GMDH} is the GMDH prediction of a single model output, x_i is a model input parameter, a's represent polynomial coefficients, and d is the number of input parameters in the model (input dimensionality). There are different GMDH training algorithms, both in parametric and non-parametric forms [33]. In this work, the parametric multi-layer GMDH algorithm is used. GMDH multi-layer approach is an inductive procedure that sorts out polynomial models, and selects the best solution (neuron) by means of an external criterion (to be defined next).

2.1.2. Components of GMDH networks

GMDH networks, like other neural networks, consist of three main components: input layer, hidden layer(s), and output layer. Each hidden layer contains a set of neurons determined by the number of possible combinations between the input parameters. The three layers are described as follows:

- Input layer: This layer has the training samples for all input parameters in the model ($\overrightarrow{x}$).
- Output layer: This layer has the output prediction (Y^{GMDH}) by the GMDH networks.
- Hidden layer(s): The first hidden layer takes its input from the input layer, while each subsequent hidden layer takes the input from the preceding layer. For example, for two input parameters (x_i,x_j), the output of a neuron in the first hidden layer (assuming linear model with interaction) can be written as

$$Y = a_0 + a_1 x_i + a_2 x_j + a_3 x_i x_j, \tag{2}$$

the second hidden layer uses the first layer as an input to construct another polynomial as follows

$$Z = b_0 + b_1 Y_i + b_2 Y_j + b_3 Y_i Y_j, \tag{3}$$

and so on for the other layers. The output of the last hidden layer is forwarded to the output layer. Neurons in the hidden layers are the most important part of GMDH, as they control the input-output relationship to be passed between layers.

Two major parameters associated with the neurons need to be specified by the user. The first parameter is the number of input factors (n_x) accepted by each neuron, which is used to generate all neurons in each hidden layer based on all possible combinations of the input factors. For example, if $n_x = 2$, then the first neuron takes input from ($x_1 x_2$), second neuron takes input from ($x_1 x_3$), and so on for all remaining input factors. The second parameter is the order of the polynomial constructed within each neuron (p). This parameter determines the polynomial order and hence the number of coefficients to be determined by least-squares within each neuron. For example, for $n_x = 2$, the output from each neuron can be determined for the first, second, and third order polynomials as follows

$$
\begin{aligned}
Y &= a_0 + a_1 x_i + a_2 x_j + a_3 x_i x_j, \text{ for } p = 1,\\
Y &= a_0 + a_1 x_i + a_2 x_j + a_3 x_i x_j + a_4 x_i^2 + a_5 x_j^2, \text{ for } p = 2,\\
Y &= a_0 + a_1 x_i + a_2 x_j + a_3 x_i x_j + a_4 x_i^2 + a_5 x_j^2 + a_6 x_i^2 x_j + a_7 x_i x_j^2 + a_8 x_i^3 + a_9 x_j^3, \text{ for } p = 3,
\end{aligned} \tag{4}
$$

where a_1-a_9 are the polynomial coefficients to be determined. Notice that for $p = 1$, the fourth term is called an interaction term as $x_i \neq x_j$, which represents one-way interaction between the parameters. If $n_x = 3$, then additional terms and coefficients are added to account for the additional parameter x_k.

Another parameter controlled by the user is the maximum number of neurons to be used in a hidden layer (n_{max}). This parameter is usually assumed to be equal to the number of input parameters (d). From layer to layer, the best neurons that satisfy the external criterion are used in the next layer, with maximum number restricted to n_{max}. The last option to be highlighted here is whether or not to include the original input layer in the training process of every hidden layer. This practice has proved to enhance the GMDH model for certain problems. In this case, x_i inputs from the input layer are feeding each hidden layer plus the input from the preceding layer. This practice is expected to increase the problem dimensionality (e.g. may double it if $n_x = d$), which could slow the training process due to the large number of combinations (neurons) to be evaluated. The summary of this discussion on GMDH networks is shown schematically in Fig. 1. This GMDH example has the following parameters: $d = 4$, $n_x = 2$, $p = 2$, $n_{max} = 4$, and a continuous feedback from the input layer.

2.1.3. External criterion for testing

There are a variety of options for the external criterion to evaluate each neuron. Two alternatives are highlighted and used in this study. The first is based on using additional test set from the original sample set to test the neurons. It is common in machine learning terminology to divide the sample set into *training set*, which is used to train the neurons in GMDH networks, and *test set*, which is used to test the trained model after it is trained and validated. The test set contains samples independent from the training set to evaluate the model capability in predicting new unseen points by the model (to avoid overfitting to the training data). In this work for GMDH, additional *validation set* is used as an external criterion during network training to prune the weak neurons. The disadvantage of this approach is that it consumes more samples from the ensemble, but it could result in better quality models as they are validated based on real data from the original model.

The second alternative is based on Akaike's Information Criterion (AIC) which is a common estimator to evaluate the quality of statistical models (e.g. regression models) [34]. This approach is preferred for limited number of samples.

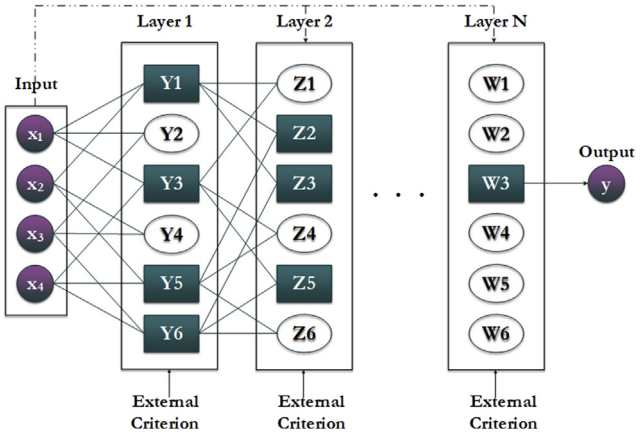

Fig. 1. Schematic of multi-layer GMDH networks (green rectangles and white ellipses represent selected and rejected neurons, respectively). (For interpretation of the references to colour in this figure legend, the reader is referred to the Web version of this article.)

2.1.4. GMDH training

After describing the GMDH structure and components, the training process of the model can be performed by the following steps:

1. The first (or next) hidden layer is constructed based on all possible combinations of the input layer (with size d), which can be determined based on the n_x value.
2. If continuous feedback from the input layer is requested, then additional d inputs and their associated neurons should be considered for each layer after the first.
3. Least-squares regression can be used to estimate the polynomial coefficients in Eq. (4) for each neuron in the layer, such that they fit the training data.
4. Compute the loss function for each neuron, by applying the external validation criterion, such as mean squared error in the validation set.
5. The best neurons (i.e. having small error) are selected as candidates for the next hidden layer. The number of candidate neurons is controlled by n_{max}.
6. Steps 1–5 are repeated for each hidden layer after the first. If the best neuron in the current layer has smaller error than the previous layer, stop the training and forward the best neuron prediction to the output layer (e.g. W3 neuron in Fig. 1). Otherwise, return to step 1.

2.2. Surrogate models via machine learning

Two other methods based on machine learning techniques are selected in this study to compare with GMDH. These methods are described briefly in this section. The first method is based on Gaussian Processes (GP) modeling [10,11]. GPs are well known surrogates in machine learning, which can be thought of as Gaussian distributions over functions instead of points. GPs can be trained in supervised form using training data and can be used for prediction with confidence interval [35]. The model output can be approximated by GP as [36].

$$
Y^{GP}(\vec{x}) = \sum_{i=0}^{k} \alpha_i \phi_i(\vec{x}) + z(\vec{x}), \tag{5}
$$

which consists of two main parts: the mean or regression part (also called trend) and the variance part (also called GP variance). The regression part consists of the regression coefficients (α_i), and the basis functions (ϕ_i) of order k. It is common to use a linear trend which is given by $\alpha_0 + \sum_{i=1}^{d} \alpha_i x_i$. The second part is represented by

covariance matrix expressed by

$$Cov\left[z\left(\vec{x}^{i}\right), z\left(\vec{x}^{j}\right)\right] = \sigma^2 \mathscr{R}\left(\vec{x}^{i}, \vec{x}^{j}\right), \tag{6}$$

where σ^2 is the process variance and $\mathscr{R}$ is called the correlation or kernel function of the process. The kernel is a fundamental part of GP, which describes the "similarit" between observations and new points. There are different options for the kernel function (e.g. linear, Gaussian, Matérn). In this study, Matérn-5/2 kernel is used, which is expressed for any two points (x, x') as

$$\mathscr{R}(x, x'; \theta) = \left(1 + \frac{\sqrt{5}\left|x - x'\right|}{\theta} + \frac{5h^2}{3\theta^2}\right) exp\left(\frac{-\sqrt{5}\left|x - x'\right|}{\theta}\right), \tag{7}$$

where θ is the scale parameter. The parameters θ and σ^2 are usually called hyperparameters of the GP model and they are determined by methods such as maximum likelihood estimation (MLE) or cross validation. One of the attractive features of GP models is that they automatically capture their uncertainty (also known as interpolation or metamodel uncertainty). Therefore, GP model predicts new points with uncertainty [10,11].

Another common metamodelling approach that has been widely used is polynomial chaos expansions (PCE) [14,37]. PCE can be perceived as a way of representing a complex function with stochastic parameters into simpler polynomial expansions [38]. PCE approximates the model output using a spectral representation of polynomial basis functions. The model output can be approximated by PCE as [39].

$$Y^{PCE} = \sum_{\vec{\beta} \in B} \alpha_{\vec{\beta}} \Phi_{\vec{\beta}}(\vec{x}), \tag{8}$$

where $\vec{\beta}$ is an index vector to identify the order of the components of $\Phi_{\vec{\beta}}(\vec{x})$, which are multivariate orthogonal polynomials (e.g. Hermite, Legendre), with coefficients expressed by $\alpha_{\vec{\beta}}$. The space B contains all polynomials in the d input variables of total degree less than or equal to p (user defined). The multivariate polynomials can be calculated by tensor product of their univariate polynomials

$$\Phi_{\vec{\beta}}(\vec{x}) = \prod_{i=1}^{d} \phi_{\beta_i}^{(i)}(x_i), \tag{9}$$

where $\phi_{\beta_i}^{(i)}(x_i)$ is a univariate polynomial for the i^{th} input with a degree of β_i. PCE requires evaluating the polynomial coefficients $\alpha_{\vec{\beta}}$, which can be performed using least-squares or quadrature methods.

In this study, GMDH is implemented in MATLAB. For other metamodelling methods (PCE, GP), MATLAB UQLab framework is used, which provides a flexible implementation of these methods [40].

2.3. Validation metrics

In machine learning and surrogate modeling, it is important to evaluate the prediction performance of the constructed surrogate, especially in prediction of the test samples, which are not used in training/validation of the surrogate model. Since this work focuses on continuous variables and regression problems, three proper metrics are selected. Mean absolute error (MAE) measures the average over the test set of the absolute differences between prediction and target observation, where all individual differences have equal weight. MAE is defined as follows

$$MAE = \frac{1}{n_t} \sum_{i=1}^{n_t} \left|Y_i - \widehat{Y}_i\right|, \tag{10}$$

where n_t is the number of samples in the test set, Y_i is the target output of the i^{th} test sample, which comes from the original model (e.g. computer code), and $\widehat{Y}_i$ is the surrogate prediction of the i^{th} test sample. Smaller MAE values imply better prediction performance. The second metric is the mean squared error (MSE), which is the average over the test set of the squared differences between prediction and target observation as follows

$$MSE = \frac{1}{n_t} \sum_{i=1}^{n_t} \left(Y_i - \widehat{Y}_i\right)^2, \tag{11}$$

where small MSE value means the surrogate is good for prediction. In general, MSE gives relatively high weight to large errors compared to MAE which gives equal weight. Both MAE and MSE are selected as they are the most common metrics in machine learning regression. The previous two metrics have units related to the target output. The third dimensionless metric is called Q_2, and it is similar to the coefficient of determination R^2, except that Q_2 is applied over the test set instead of the training set (R^2). The Q_2 metric is expressed by

$$Q_2 = 1 - \frac{\sum_{i=1}^{n_t} \left(Y_i - \widehat{Y}_i\right)^2}{\sum_{i=1}^{n_t} \left(Y_i - \overline{Y}\right)^2}, \tag{12}$$

where $\overline{Y}$ is the mean of the Y samples. Q_2 measures the fraction of the explained variance in the test samples by the surrogate. Therefore, Q_2 is selected here because it refers to how much of the variance that we aim to quantify in UQ can be explained by the surrogate. As Q_2 approaches 1.0 (maximum value), the quality of the surrogate increases. Q_2 can be zero or even negative at which the surrogate fails to capture the target variance (i.e. the surrogate variance is higher), implying a poor performance by the surrogate model.

2.4. Selected applications and reactor simulations

The first application (low dimensional) is a reactor thermal-hydraulics problem based on the widely-used BWR Full-size Fine-mesh Bundle Test (BFBT) [41]. The lattice is a full-scale 8 × 8 BWR fuel bundle with a wide range of power, pressure, flow and inlet temperature experimental conditions. The void fractions are measured using radiation-based tomography at 4 axial locations. The void fraction at the axial location z = 170.6 cm is selected as the main output of interest for this application (it is referred to by DEN2 in the BFBT benchmark). The details of experimental conditions and uncertainties considered in BFBT experiments are listed in Table 1.

Table 1
Parametric uncertainties of the 9 input parameters used in the application of the BFBT void fraction measurement [41,43].

Parameter	Mean Value	Uncertainty
Pressure (MPa)	8.71	1%
Temperature (°C)	291.65	±1.5 °C
Mass Flow Rate (kg/s)	15.16	1%
Power (MW)	0.21–7.35	1.5%
LiqWalHTC	0.616	0.211
SubBoilHTC	1.236	0.089
WallDrag	1.411	0.183
B/S-IntDragBund	1.339	0.116
B/S-IntDragVess	1.234	0.345

Table 2
GMDH optimized training parameters used in the case studies presented in this section.

Parameter	Case 1 (sec. 3.1)	Case 2 (sec. 3.2)
Number of input parameters (d)	9	65
Number of inputs per neuron (n_x)	3	3
Polynomial order per neuron (p)	2	1
Maximum number of neurons per layer (n_{max})	9	65
Continuous input layer feedback	Yes	Yes
External criterion	Test Set	AIC
Number of training samples	400	400
Number of validation samples	200	–
Number of test samples	300	300

Based on previous UQ studies on this benchmark [42,43], **9** input parameters are expected to have large influence on the void fraction uncertainty, which can be classified into two main categories.

- Boundary conditions: system pressure (**P**), inlet temperature (**T**), mass flow rate (**MassFlow**), and system power (**Power**).
- Physical model parameters: single phase liquid to wall heat transfer coefficient (**LiqWalHTC**), subcooled boiling heat transfer coefficient (**SubBoilHTC**), wall drag coefficient (**WallDrag**), interfacial drag (bubbly/slug rod bundle-Bestion) coefficient (**B/S-IntDragBund**), and interfacial drag (bubbly/slug vessel) coefficient (**B/S-IntDragVess**).

Notice that GMDH is used in this work to perform a parametric UQ, where the distribution as well as the input parameters are predefined. Other non-parametric UQ approaches can be used (e.g. Wilk's formula, kernel-based methods) [44]. To summarize this application, the input-output problem can be written as

$$Y = \alpha = F(\mathrm{P}, \mathrm{T}, \ldots, \mathrm{B/S} - \mathrm{IntDragVess}), \tag{13}$$

where α is the void fraction output at $z = 170.6$ cm and F is the TRACE safety analysis code used to model the problem [45]. All input parameters are sampled from a univariate normal distribution based on the parameters reported in Table 1.

The second application (high dimensional) is a nuclear data problem, which aims to map the relationship between the uncertainty in the input parameters: the homogenized cross-sections (HXS), to the output: lattice k_∞. To achieve this, a two-step process is followed to propagate the uncertainty into the output. First, the uncertainty in the fundamental microscopic cross-section data needs to be propagated into the HXS. The Sampler module in the SCALE code system [46,47] is used for uncertainty propagation. The 56-group cross-section and covariance libraries are used in the lattice physics code TRITON, which calculates the HXS. The covariance library contains nuclide-dependent microscopic cross-

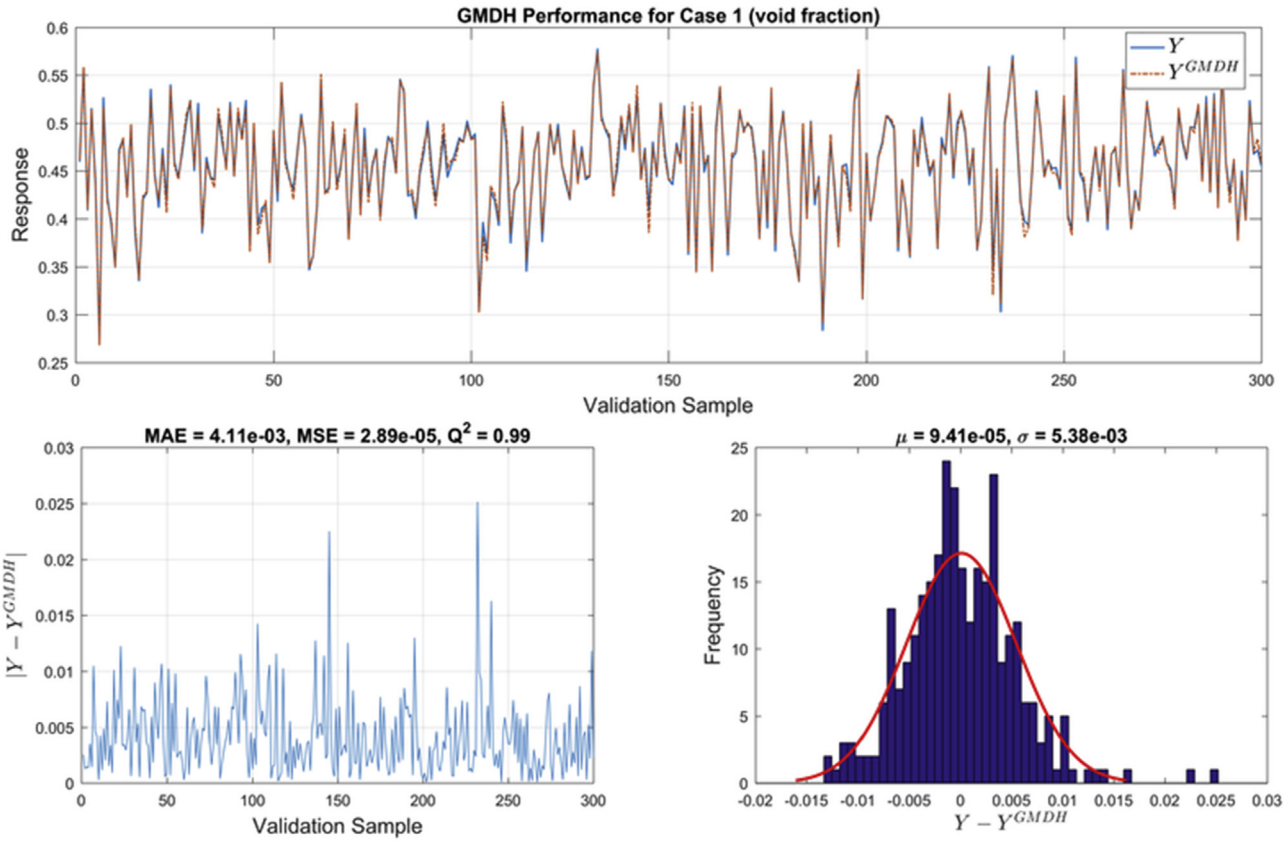

Fig. 2. GMDH performance metrics for the low dimensional Case 1 (void fraction).

section uncertainties for most of the isotopes relevant to reactor applications (e.g. U-235, U-238, Pu-239, etc.). The multigroup cross-sections are then collapsed into an 8-group form. Microscopic cross-section uncertainties are sampled from a multivariate normal distribution using the covariance matrices in the 56-group covariance library. In the second step, the HXS are used to calculate the lattice reactivity (i.e. k_∞) which is the output of interest. It is worth mentioning that this problem was used in a 2-group form in our previous study, which focused on variance decomposition by Shapley effect [48]. Since this is a high-dimensional problem, the 8-group HXS parameters are too long to be listed explicitly, but they can be classified into three main categories:

- Non-scattering reactions: these include cross-sections of fission (Σ_f^g), absorption (Σ_a^g), and (n,2n) reactions, homogenized over all isotopes in the problem. The superscript g refers to the energy group, where $g = 1, 2, \ldots, 8$.
- In-group scattering (i.e. $g = g'$): the neutron does not change the energy group after encountering a scattering reaction (e.g. $\Sigma_s^{4\to4}$, $\Sigma_s^{8\to8}$).
- Group-transfer scattering (i.e. $g \neq g'$): the neutron changes the energy group after scattering. For example, $\Sigma_s^{6\to7}$ is the scattering cross-section from energy group 6 to 7. This also includes upscattering cross-sections, e.g. $\Sigma_s^{5\to4}$ is the scattering cross-section from lower energy group 5 to higher energy group 4.

After removing all HXS with insignificant value (essentially zero) from all energy groups, the final list contains **65** HXS parameters from the eight groups. Indeed, most of the negligible HXS are from the (n,2n) and the two scattering categories. The geometry selected for this application is a pressurized water reactor (PWR) lattice geometry based on the Benchmark for Evaluation And Validation of Reactor Simulations (BEAVRS) [49]. The lattice is a 17 × 17 PWR design, with 264 UO_2 fuel rods, 24 guide tubes, and one instrumentation tube. The lattice is designed in a quarter symmetry in TRITON and depleted to 50 GWD/MTU. Two main responses/outputs are analyzed in this study: (1) k_∞ at the beginning of life (BOL) without burnup, and (2) k_∞ at the end of life (EOL) with burnup of 50 GWD/MTU. More details about the geometry and material specifications are given in the benchmark report [49]. To summarize this application, the input-output problem can be written as

$$Y = k_\infty = F\left(\Sigma_f^1, \Sigma_a^1, \ldots, \Sigma_s^{8\to8}\right), \tag{14}$$

where k_∞ is the lattice infinite multiplication factor output and F is the TRITON lattice physics code in the SCALE code system [46,47].

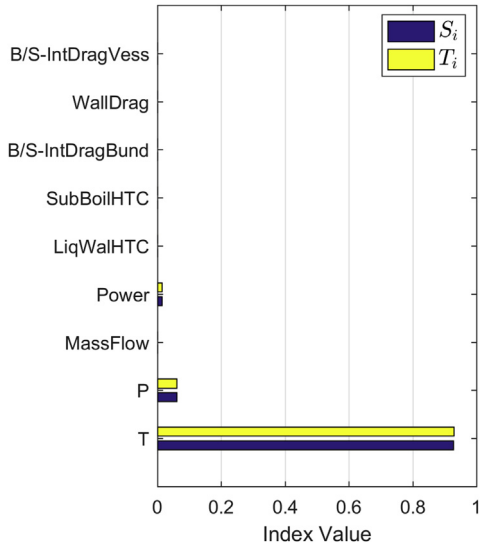

Fig. 3. GMDH-based normalized Sobol indices for the void fraction variance (based on 10^4 samples).

3. Results and discussion

Performance analysis of GMDH networks in the context of UQ is presented in this section. The application based on BFBT void fraction prediction is presented first, followed by the application of the 8-group HXS. The authors performed a thorough analysis of the GMDH parameters to ensure their convergence, but detailed results are not presented here for conciseness. The summary of the GMDH parameters used in the two case studies is presented in Table 2. After presenting the results of GMDH for the two case studies, a comparison of GMDH with other metamodelling methods is presented.

3.1. Case 1: two-phase flow in a BWR channel

This application has 9 input parameters, and this number is used as the maximum number of neurons per layer. Combination of three inputs in a second order polynomial is used in each neuron. A total of 900 samples is used for this problem, and these samples are divided into 400 training samples, 200 validation samples, and 300 test samples. The original input layer is used to feed every hidden layer in this problem. GMDH results are presented in Fig. 2 for the 300 test samples. It is clear that GMDH network is able to capture the void fraction in the test samples, as can be told from the very good agreement of Y and Y^{GMDH} in Fig. 2. The validation metrics of GMDH based on the test set demonstrate very good performance as Q_2 reports that GMDH captures 99% of the variance in the test set. This is also true based on the small values of MAE and MSE for this test case. The network converged after 1 hidden layer. In general, these results show that GMDH can perform well in low dimensional problems.

After validating GMDH performance, it can be used to perform variance decomposition using Sobol indices, which is a typical and expensive UQ task [50]. The first-order index (S_i) measures the main effect of varying the parameter x_i alone on the output variance, and it is defined as

$$S_i = \frac{Var[E[y|x_i]]}{Var[y]}, \tag{15}$$

where $Var[E[y|x_i]]$ represents the reduction in $Var[y]$ when x_i is fixed. The total index (T_i) measures the total effect on the output variance from x_i alone as well as its interactions with other parameters, and it is defined as

$$T_i = 1 - \frac{Var[E[y|\vec{x}_{\sim i}]]}{Var[y]}, \tag{16}$$

which expresses the remaining variance of y, when all input parameters other than x_i (i.e. $\vec{x}_{\sim i}$) are fixed. The condition $T_i \geq S_i$ must be satisfied, where the equality holds when the parameter i has no interactions with other parameters. These Sobol indices are estimated in this paper by Monte Carlo methods used before in Refs. [48,51]. Since estimating Sobol indices requires executing the original model, the accurate and fast GMDH network is used instead of TRACE, and the results are plotted in Fig. 3 using 10^4 Monte Carlo samples.

First, the void fraction variance is dominated by the inlet temperature (subcooling) as T contributes to more than 90% of the total variance. The pressure and power conditions have 5% and 1% variance contribution, respectively, and the remaining parameters contribute to less than 1% of the output variance. Also, the first-order and total indices have close numerical estimates, implying small effect of parameter interactions on the output variance. From physics point of view, inlet temperature of the fluid is the most sensitive parameter, as a dense liquid needs additional time to boil to form gas void than a light fluid entered with high inlet temperature. In addition, the void fraction measurement is at z = 170.6 cm, which is close to the inlet, and hence sensitive to the inlet temperature. In general, we can observe that the uncertainty in the boundary conditions (e.g. T, P, etc.) is more influential on the void fraction uncertainty than the physical model parameters, based on the parametric uncertainties assigned in Table 1.

3.2. Case 2: 8-group homogenized cross-sections

This problem has 65 input parameters, and this number is used as the maximum number of neurons per layer. Combination of three inputs in a first order polynomial with one-way interaction is used in each neuron. A total of 700 samples is used for this problem, and these samples are divided into 400 training samples and 300 test samples, where AIC is used as the external criterion. The original input layer is also used to feed every hidden layer in the network. GMDH results are presented for this case study in Fig. 4 for the 300 test samples. The results presented in Fig. 4 are for lattice k_∞ at EOL with burnup of 50 GWD/MTU. Notice that GMDH also demonstrated similar performance at BOL, but results are not shown here for brevity. GMDH network maintains its performance, even though the dimensionality increases to 65. MSE and MAE are decreased further for Case 2, and Q_2 remains close to 99%. The good agreement in EOL's k_∞ can be clearly seen in Y and Y^{GMDH} close predictions. At BOL, the network converged after 9 hidden layers, while at EOL the network needed 8 hidden layers to converge.

After validating GMDH performance for the high dimensional case, it can be used to perform uncertainty propagation task with large number of samples. Monte Carlo uncertainty propagation using GMDH networks is performed using 10^4 samples, generated from the covariance matrix of the HXS parameters. The uncertainty results of k_∞ are plotted in Fig. 5 for BOL and EOL cases, which correspond to 1-σ around the mean. The uncertainty in lattice k_∞ at BOL is about 726 pcm, which is equivalent to 0.58% of the mean. At EOL, the lattice k_∞ decreases due to fuel depletion across the cycle. The k_∞ uncertainty (1-σ) at EOL is 497 pcm, which corresponds to 0.61% of the mean.

3.3. Comparison between the methods

Finally, a comparison between GMDH performance with other machine learning and metamodelling methods is conducted. The validation metrics for all methods are listed in Table 3. For the GP model in Case 1: Matérn-5/2 kernel, MLE estimation for the hyperparameters, and a quadratic trend (GP mean) are used. In Case 2, similar settings are used except that a linear trend is used, as the quadratic trend gives poor results. For PCE, Hermite polynomials and a truncation polynomial degree of 2 are used for both cases as they yielded best results. The training and test samples used in GMDH and the other two methods are similar.

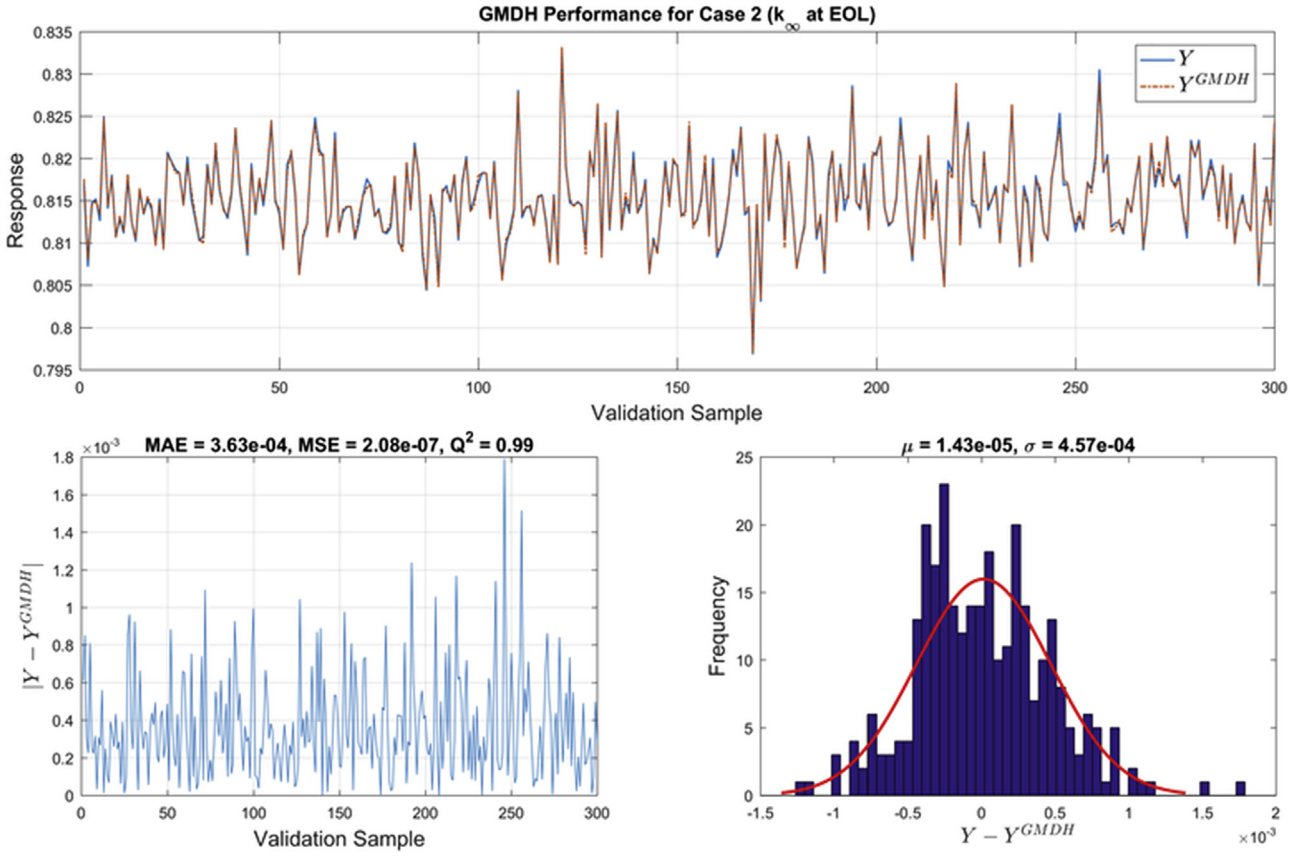

Fig. 4. GMDH performance metrics for the high dimensional Case 2 (k_∞ at EOL).

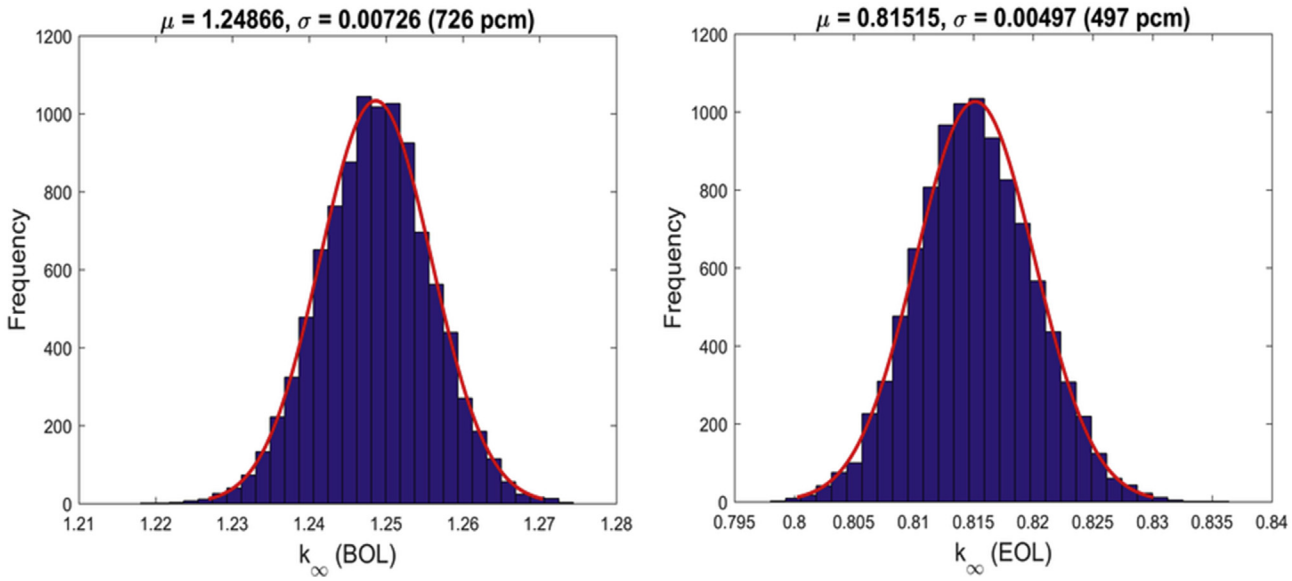

Fig. 5. GMDH-based uncertainty propagation of lattice k_∞ at BOL and EOL using 10^4 samples.

Table 3
Comparison of metrics between GMDH and other metamodelling methods for the two case studies.

Method	Case 1			Case 2		
	MAE	MSE	Q_2	MAE	MSE	Q_2
GMDH	4.112E-03	2.887E-05	0.9901	3.633E-04	2.082E-07	0.9915
GP	5.385E-04	4.281E-07	0.9999	3.169E-04	1.694E-07	0.9931
PCE	1.719E-03	6.042E-06	0.9979	3.183E-03	1.597E-05	0.3460

Based on the comparison in Table 3, we can find that for the low dimensional case, GP and PCE have a slightly better performance than GMDH in terms of MAE and MSE. However, all three methods have Q_2 value close to 1.0. In general, PCE is expected to perform well in low-dimensional problems. However, moving to the high dimensional Case 2, PCE shows poor Q_2 value relative to the other two methods, where GMDH and GP demonstrate competitive performance. Notice that the small values of MAE and MSE for PCE in Case 2 can be misleading. For example, MAE for PCE in Case 2 is 3.18×10^{-3}, which is equivalent to 318 pcm. This uncertainty resulted from PCE model mis-prediction is considered significant for k_∞, especially that the total uncertainty we aim to capture is 500 pcm (also notice the difference in the sample sizes between Fig. 5 and the test set). In this situation, Q_2 is considered a better metric as it indicates that only 35% of the sample variance in the test set is captured by PCE. Even though GP shows a slightly better performance than GMDH in case 2, it is still important to notice the advantages of GMDH, which include handling dimensionality, preserving simple mathematical foundation and training, and providing excellent accuracy. On the other hand, GP models have more complicated mathematical foundation and they are more difficult to train, due to the need to optimize the model hyperparameters.

Finally, we can conclude that GMDH provides competitive and reliable performance as a surrogate model, when it is applied to low and high dimensional problems. The reader should notice that we applied one form of GMDH networks (the most common) in this study, which implies that other GMDH algorithms can provide different results (either better or worse).

4. Conclusions

Multi-layer GMDH networks are used in this study to perform UQ of reactor simulations. GMDH networks are considered one of the earliest deep learning methods. GMDH is used as a surrogate/metamodel in this study to replace high fidelity computer models with a metamodel to facilitate UQ tasks. Application of GMDH to a low dimensional problem of two-phase flow in a BWR channel is used, while application of nuclear data using 8-group homogenized cross-sections is used as a high-dimensional case. In both cases, GMDH networks show very good performance with Q_2 value reaches as large as 0.99 (maximum is 1.0). Comparison with other surrogate methods including GP and PCE demonstrates that GMDH has competitive performance with them at the low dimensional problem, and reliable performance at the high dimensional problem. In future work, the concept of deep GPs will be applied to UQ of nuclear reactor simulations. Deep GPs have an attractive features of handling high dimensional problems plus preserving the GP feature of prediction with confidence bounds. Additional applications of GMDH networks to multiphysics simulations as well as fault detection will be performed. Comparison of GMDH performance to other networks such as neural networks can also be done.

Appendix A. Supplementary data

Supplementary data to this article can be found online at https://doi.org/10.1016/j.net.2019.07.023.

References

[1] Y. LeCun, Y. Bengio, G. Hinton, Deep learning, Nature 521 (7553) (2015) 436.
[2] L. Deng, D. Yu, et al., Deep learning: methods and applications, Foundations and Trends® in Signal Processing 7 (3–4) (2014) 197–387.
[3] T. Poggio, H. Mhaskar, L. Rosasco, B. Miranda, Q. Liao, Why and when can deep-but not shallow-networks avoid the curse of dimensionality: a review, Int. J. Autom. Comput. 14 (5) (2017) 503–519.
[4] S.M. Erfani, S. Rajasegarar, S. Karunasekera, C. Leckie, High-dimensional and large-scale anomaly detection using a linear one-class SVM with deep learning, Pattern Recognit. 58 (2016) 121–134.
[5] E. Keogh, A. Mueen, Curse of dimensionality, in: Encyclopedia of Machine Learning, Springer, 2011, pp. 257–258.
[6] R.C. Smith, Uncertainty Quantification: Theory, Implementation, and Applications, vol. 12, Siam, 2013.
[7] A. Saltelli, M. Ratto, T. Andres, F. Campolongo, J. Cariboni, D. Gatelli, M. Saisana,

S. Tarantola, Global Sensitivity Analysis: the Primer, John Wiley & Sons, 2008.
[8] A.I. Forrester, A.J. Keane, Recent advances in surrogate-based optimization, Prog. Aerosp. Sci. 45 (1–3) (2009) 50–79.
[9] K.K. Vu, C. D'Ambrosio, Y. Hamadi, L. Liberti, Surrogate-based methods for black-box optimization, Int. Trans. Oper. Res. 24 (3) (2017) 393–424.
[10] J.D. Martin, T.W. Simpson, Use of kriging models to approximate deterministic computer models, AIAA J. 43 (4) (2005) 853–863.
[11] A. Marrel, B. Iooss, F. Van Dorpe, E. Volkova, An efficient methodology for modeling complex computer codes with Gaussian processes, Comput. Stat. Data Anal. 52 (10) (2008) 4731–4744.
[12] M.D. Buhmann, Radial Basis Functions: Theory and Implementations, vol. 12, Cambridge university press, 2003.
[13] S. Chan, A.H. Elsheikh, A machine learning approach for efficient uncertainty quantification using multiscale methods, J. Comput. Phys. 354 (2018) 493–511.
[14] B. Sudret, Global sensitivity analysis using polynomial chaos expansions, Reliab. Eng. Syst. Saf. 93 (7) (2008) 964–979.
[15] R. Tripathy, I. Bilionis, Deep UQ: learning deep neural network surrogate models for high dimensional uncertainty quantification, J. Comput. Phys. 375 (2018) 565–588.
[16] Y. Zhu, N. Zabaras, Bayesian deep convolutional encoder–decoder networks for surrogate modeling and uncertainty quantification, J. Comput. Phys. 366 (2018) 415–447.
[17] A.G. Ivakhnenko, The group method of data of handling; a rival of the method of stochastic approximation, Soviet Automatic Control 13 (1968) 43–55.
[18] A.G. Ivakhnenko, Polynomial theory of complex systems, IEEE Trans. Sys. Man Cybern. (4) (1971) 364–378.
[19] J. Schmidhuber, Deep learning in neural networks: an overview, Neural Netw. 61 (2015) 85–117.
[20] S.J. Farlow, The GMDH algorithm of ivakhnenko, Am. Statistician 35 (4) (1981) 210–215.
[21] A. Ivakhnenko, G. Ivakhnenko, The review of problems solvable by algorithms of the group method of data handling (GMDH), Pattern Recognition and Image Analysis C/C of Raspoznavaniye Obrazov I, Analiz Izobrazhenii 5 (1995) 527–535.
[22] A. Ivakhnenko, Heuristic self-organization in problems of engineering cybernetics, Automatica 6 (2) (1970) 207–219.
[23] X. Jia, Y. Di, J. Feng, Q. Yang, H. Dai, J. Lee, Adaptive virtual metrology for semiconductor chemical mechanical planarization process using GMDH-type polynomial neural networks, J. Process Control 62 (2018) 44–54.
[24] T. Kondo, J. Ueno, Multi-layered GMDH-type neural network self-selecting optimum neural network architecture and its application to 3-dimensional medical image recognition of blood vessels, Int. J. Innov. Comput. Inf. Control 4 (1) (2008) 175–187.
[25] M. Witczak, J. Korbicz, M. Mrugalski, R.J. Patton, A GMDH neural network-based approach to robust fault diagnosis: application to the damadics benchmark problem, Contr. Eng. Pract. 14 (6) (2006) 671–683.
[26] S. Dolati, N. Amanifard, H.M. Deylami, Numerical study and GMDH-type neural networks modeling of plasma actuator effects on the film cooling over a flat plate, Appl. Therm. Eng. 123 (2017) 734–745.
[27] X. Tian, V. Becerra, N. Bausch, T. Santhosh, G. Vinod, A study on the robustness of neural network models for predicting the break size in LOCA, Prog. Nucl. Energy 109 (2018) 12–28.
[28] T. Cong, G. Su, S. Qiu, W. Tian, Applications of ANNs in flow and heat transfer problems in nuclear engineering: a review work, Prog. Nucl. Energy 62 (2013) 54–71.
[29] M.-G. Park, H.-C. Shin, Reactor power shape synthesis using group method of data handling, Ann. Nucl. Energy 72 (2014) 467–470.
[30] F. Khoshahval, S. Yum, H.C. Shin, J. Choe, P. Zhang, D. Lee, Smart sensing of the axial power and offset in NPPs using GMDH method, Ann. Nucl. Energy 121 (2018) 77–88.
[31] B. Lu, B. Upadhyaya, Monitoring and fault diagnosis of the steam generator system of a nuclear power plant using data-driven modeling and residual space analysis, Ann. Nucl. Energy 32 (9) (2005) 897–912.
[32] H. Kim, M.G. Na, G. Heo, Application of monitoring, diagnosis, and prognosis in thermal performance analysis for nuclear power plants, Nucl. Eng. Technol. 46 (6) (2014) 737–752.
[33] H.R. Madala, A.G. Ivakhnenko, Inductive Learning Algorithms for Complex Systems Modeling, vol. 368, cRc press, Boca Raton, 1994.
[34] H. Akaike, A new look at the statistical model identification, IEEE Trans. Autom. Control 19 (6) (1974) 716–723.
[35] C.E. Rasmussen, Gaussian processes in machine learning, in: Advanced Lectures on Machine Learning, Springer, 2004, pp. 63–71.
[36] C. Lataniotis, S. Marelli, B. Sudret, UQlab user manual–kriging (Gaussian process modelling), Rep. UQLab-V0 (2015) 9–105.
[37] G. Blatman, B. Sudret, Adaptive sparse polynomial chaos expansion based on least angle regression, J. Comput. Phys. 230 (6) (2011) 2345–2367.
[38] R.G. Ghanem, P.D. Spanos, Stochastic finite element method: response statistics, in: Stochastic Finite Elements: A Spectral Approach, Springer, 1991, pp. 101–119.
[39] S. Marelli, B. Sudret, UQlab User Manual–Polynomial Chaos Expansions, Chair of Risk, Safety & Uncertainty Quantification, ETH Zürich, 0.9-104 edition.
[40] S. Marelli, B. Sudret, UQlab: a framework for uncertainty quantification in Matlab, in: Vulnerability, Uncertainty, and Risk: Quantification, and Management, Mitigation, 2014, pp. 2554–2563.
[41] B. Neykov, F. Aydogan, L. Hochreiter, K. Ivanov, H. Utsuno, F. Kasahara, E. Sartori, M. Martin, NUPEC BWR full-size fine-mesh bundle test (BFBT) benchmark, in: Tech. rep., Organisation for Economic Co-operation and Development(OECD), Nuclear Energy Agency, 2006.
[42] M.I. Radaideh, K. Borowiec, T. Kozlowski, Integrated framework for model assessment and advanced uncertainty quantification of nuclear computer codes under bayesian statistics, Reliab. Eng. Syst. Saf. 189 (2019) 357–377.
[43] X. Wu, T. Kozlowski, H. Meidani, K. Shirvan, Inverse uncertainty quantification using the modular Bayesian approach based on Gaussian process, Part 2: application to trace, Nucl. Eng. Des. 335 (2018) 417–431.
[44] N.V. Queipo, R.T. Haftka, W. Shyy, T. Goel, R. Vaidyanathan, P.K. Tucker, Surrogate-based analysis and optimization, Prog. Aerosp. Sci. 41 (1) (2005) 1–28.
[45] U.S.NRC, TRACE v5.840 Theory Manual: Fields Equations, Solution Methods, and Physical Models, US Nuclear Regulatory Commission, Washington, D.C., United States.
[46] S.M. Bowman, SCALE 6: comprehensive nuclear safety analysis code system, Nucl. Technol. 174 (2) (2011) 126–148.
[47] B.T. Rearden, M.A. Jessee, SCALE Code System, Tech. rep., Oak Ridge National Lab.(ORNL), 2018. Oak Ridge, TN (United States).
[48] M.I. Radaideh, S. Surani, D. O'Grady, T. Kozlowski, Shapley effect application for variance-based sensitivity analysis of the few-group cross-sections, Ann. Nucl. Energy 129 (2019) 264–279.
[49] N. Horelik, B. Herman, B. Forget, K. Smith, Benchmark for evaluation and validation of reactor simulations (BEAVRS), v1. 0.1, in: Proc. Int. Conf. Mathematics and Computational Methods Applied to Nuc. Sci. & Eng, 2013. Sun Valley, Idaho, United States, May 5-9.
[50] I.M. Sobol, Sensitivity estimates for nonlinear mathematical models, Math. Model. Civ. Eng. 1 (4) (1993) 407–414.
[51] G. Glen, K. Isaacs, Estimating sobol sensitivity indices using correlations, Environ. Model. Softw 37 (2012) 157–166.