Chapter 13
Evaluation of Surrogate Modelling Methods for Turbo-Machinery Component Design Optimization

Gianluca Badjan, Carlo Poloni, Andrew Pike and Nadir Ince

Abstract Surrogate models are used to approximate complex problems in order to reduce the final cost of the design process. This study has evaluated the potential for employing surrogate modelling methods in turbo-machinery component design optimization. Specifically four types of surrogate models are assessed and compared, namely: neural networks, Radial Basis Function (RBF) Networks, polynomial models and Kriging models. Guidelines and automated setting procedures are proposed to set the surrogate models, which are applied to two turbo-machinery application case studies.

Keywords Surrogate models · Neural networks · Turbo-machinery

13.1 Introduction

This paper is based on a MSc thesis in Mechanical Engineering (University of Trieste) [1]. The main author was a research student at the University and is now employed as an Aerodynamics Methods Engineer at ALSTOM Power UK, facilitating tailored applied research collaboration between the University and the Company. The aim of the research was to evaluate the possibility of employing surrogate modelling methods for turbo-machinery component design optimization. The main idea behind these methods is to replace expensive to compute physical models with surrogate models, in order to speed up the entire design optimization process. These surrogate models

G. Badjan (✉) · A. Pike · N. Ince
ALSTOM Power Ltd, Newbold Road, Rugby CV21 2NH, UK
e-mail: gianluca.badjan@power.alstom.com

A. Pike
e-mail: andrew.pike@power.alstom.com

N. Ince
e-mail: nadir.ince@power.alstom.com

C. Poloni
University of Trieste, Via Valerio 8, 34127 Trieste, Italy
e-mail: poloni@units.it

D. Greiner et al. (eds.), *Advances in Evolutionary and Deterministic Methods for Design, Optimization and Control in Engineering and Sciences*, Computational Methods in Applied Sciences 36, DOI 10.1007/978-3-319-11541-2_13

are required to be cheap to compute and easy to use, whilst providing an adequate representation of the real problem. Four different surrogate models were employed in this study: Feed-Forward Backpropagation Neural Networks (FFBP NN), Radial Basis Function (RBF) Networks, Kriging models and polynomial models.

In the first part of this paper the surrogate modelling methods will be summarized, providing guidelines and automated procedures for their setting. Surrogate models will subsequently applied to two turbo-machinery case studies.

13.2 Neural Networks

The FFBP NNs employed in this study are multilayer networks with a single hidden layer. FFBP NNs are characterized by a very complex setting process, due to the high number of parameters to be set and their multi-modal performance function [2]. A critical choice for the neural network is the number of hidden neurons, since it determines the "flexibility" of the model. This parameter is usually chosen directly by the user. Unfortunately, the complexity of the modelled process is usually unknown, and FFBP NNs are tested for different architectures in order to find the best fitting for a particular dataset. This "Trial and Error" procedure is very time consuming, making it desirable to automate the setting of neural networks.

There are two main approaches to design a FFBP NN in an automatic fashion [1]:

- Constructive Methods
- Pruning Methods

The pruning methods appear to be the most convincing, since they allow a more tailored neural networks setting than the constructive methods. Two pruning techniques are evaluated in this paper: the Optimal Brain Surgeon (OBS) and the MATLAB *trainbr* algorithm.

13.2.1 Optimal Brain Surgeon

The OBS algorithm is a pruning technique developed by Hassibi [3] and implemented by Noorgard [4]. Each pruning session returns a certain number of pruned (partially connected) networks, one for each OBS algorithm iteration. Consequently, a neural network must be chosen according to some criteria, which are provided by Noorgard [4].

In addition, a new criteria was introduced by Badjan [1]:

$$\text{Balanced Valid. Error} = \text{Valid. Error} + \|\text{Valid. Error} - \text{Train. Error}\|_2 \qquad (13.1)$$

called *balanced validation error*, which takes into account both the error on the estimation subset and on the validation subset (note that Valid. and Train. error are ≥ 0 by definition).

According to this criteria, FFBP NNs with a low validation error that show similar performances on both the estimation and the validation subsets are preferred to the other networks.

13.2.2 MATLAB trainbr

In MATLAB *trainbr* [13], the following cost function is implemented:

$$MSE_{reg} = \alpha MSW + \beta MSE \tag{13.2}$$

where MSE is the mean square error, MSW is the sum of the squares of the network weight and biases, α and β are regularization parameters.

Minimizing Eq. (13.2) leads to lower values of the network weights and biases, making the network response smoother and less prone to overfit. In fact, assigning low values to the free parameters may be viewed as equivalent to pruning the neural network.

13.2.3 Dynamic Threshold Neural Networks

The Dynamic Threshold Neural Network (DTNN) was originally proposed by Chiang and Fu [5] for pattern recognition purposes, but it was also successfully applied to function approximation problems by Pediroda [6] and Poloni et al. [7]. The DTNN was designed to employ *Static Threshold Quadratic Sigmoidal Neurons* in the hidden layer and *Dynamic Threshold Quadratic Sigmoidal Neurons* in the output layer.

This network configuration produces outputs in the range [0, 1], which is adequate for pattern recognition purposes, but it could represent a limitation for function approximation purposes. In the view of the Authors, having an output range limited between two fixed values implies that the training-set contains both the minima and the maxima of the objective function. If the training-set targets are normalized between [0, 1], then other new input configurations will always produce target values included between [0, 1]. An example may illustrate the concept: considering a training-set with the maximum objective function value 12 and the minimum objective function value -5, after the data normalization 12 will correspond to 1 and -5 to 0; if there is a maxima (or minima) somewhere in the input domain with a value 15 (-7), then the corresponding output will be again 1 (0). It can be noticed that even if the objective value is saturated, the input configuration might represent the true maxima (minima). In any case, no robust analysis could be performed using the

surrogate model on that point, since it would not be possible to approximate in a proper way the shape of the objective function in the saturated zone.

For these reasons, in this study it was decided to rearrange the architecture of the DTNN, employing the Dynamic Threshold Quadratic Sigmoidal neurons directly in the hidden layer and the standard linear transformation in the output layer, thereby removing the output limits.

The setting process of the DTNN presents some differences with respect to classic FFBP NNs. In fact, fewer neurons are generally required to fit a dataset, since they have a higher approximation capability than the neurons of classic FFBP NNs [5]. It was therefore decided to use a constructive methodology to train this type of network.

The proposed setting process for DTNNs consists in:

1. Set the DTNN with n hidden neurons.
2. Train the network m times from different initial configurations.
3. Check the performance on the validation subset.
4. Set a new DTNN with $n + 1$ hidden neurons.
5. Train the new network a couple of times and check the performance on the validation subset.
6. If the validation error increases stop the procedure, otherwise go to point 4.

13.3 RBF Networks

In this paper, Gaussian, multiquadrics and inverse-multiquadrics functions were chosen to build RBF Networks, since they have a shape parameter σ used to control the domain of influence of the radial basis function. There are various strategies in the literature for selecting an appropriate value for the shape parameter σ. The *leave-one-out* (LOO) error is a well known criteria for setting RBF Networks. However, the computational cost can be very high, of order $O(N^4)$, which becomes prohibitively expensive even for problems of modest size. Fortunately, Rippa [8] proposed a technique to reduce the computational cost of the LOO metric to $O(N^3)$, which was here implemented.

Based on the LOO error, an iterative procedure to select the optimal shape parameter for interpolating RBF Network is proposed in this paper:

1. Initialize σ to 1 and evaluate the LOO error.
2. Set σ_{new} to 0.5 and evaluate the corresponding LOO error.
3. If LOO σ_{new} < LOO σ then $\sigma = \sigma_{new}$ and $\sigma_{new} = \sigma/a$, otherwise $\sigma = \sigma_{new}$ and $\sigma_{new} = \sigma * b$.
4. Evaluate the LOO error for the RBF Network set with σ_{new}.
5. Repeat the procedure from point 3 until the maximum number of iterations is reached.
6. Return the RBF Network which scored the minimum LOO error.

The parameters a, b can be set by the user, determining how much σ is increased or decreased at each iteration. In addition, there is also the possibility to vary these parameters during the iterations, to gradually reduce or increase the step size of the shape parameter. The Authors suggest to set $a = 1.5$ and $b = 1.8$. The maximum number of iterations should take into account the time required to solve a single LOO measure. However, 20 iterations should be an appropriate number for the majority of the problems.

RBF Networks can also perform a regression of the data, introducing the regularization parameter λ in a similar way as for the Kriging model [9]. Keane and Nair [10] suggest to set λ to the variance of the noise in the response data, but since this information in usually unknown. The remaining option is to add it to the list of parameters to be estimated. In this study, both the shape parameter σ and the regularization parameter λ were searched throughout their domain using a Genetic Algorithm (GA). Suitable upper and lower bounds for the search of λ are 10^{-6} and 1 respectively [10].

13.4 Polynomial Models

Polynomial models can be applied to multi-dimensional problems taking into account interaction terms [9]. In this paper, optimal values for global and interaction orders are found by applying cross-validation.

13.5 Kriging Models

Kriging models are powerful methods based on Gaussian processes. They can perform either interpolation or regression of data. In this paper, Kriging models are set via maximizing the marginal likelihood function [9].

13.6 Assessment Criteria for Surrogate Models

If the observational data are abundant, a randomly selected subset (Hastie et al. [11] recommend around the 25 % of the total $\boldsymbol{x} \rightarrow y$ pairs) should be set aside for model testing purposes. These observations must not be touched during the previous stages, as their sole purpose is to allow us to evaluate the testing error (based on the difference between the true and approximated function values at the test sites) once the model has been built. Standard assessment criteria for surrogate models are Normalized Root Mean Square Error (NRMSE) and Coefficient of Determination (r^2). According to [9], good surrogate models should have NRMSE $<10\,\%$ and $r^2 > 0.8$.

Furthermore, a new criteria is introduced in this paper, called RANKING [1], the aim of which is to assess the capability of surrogate models to replicate the trend expressed by the underlying function.

The RANKING is evaluated using the following procedure:

1. Sort the true solutions of a particular dataset in ascending order.
2. Check if the corresponding approximated solutions increase their values monotonically.
3. A score of 1 is given to the solutions that increase step by step, referring to the previous highest value (absolute RANKING).
4. Finally the score is divided by the number of points in the dataset and multiplied by 100.

A numerical example may illustrate the steps:

- Assuming the following true solutions y: [12, 43, 2, 33, 30, 31] and the surrogate model approximations $\widehat{y}$: [10, 45, 3, 32, 35, 31].
- Now sorting the true solutions y in an ascending order: [2, 12, 30, 31, 33, 43] with the corresponding original index: [3, 1, 5, 6, 4, 2].
- Then the corresponding approximation $\widehat{y}$ will be: [3, 10, 35, 31, 32, 45].
- The scores for each point are [1, 1, 1, 0, 0, 1] and their sum is 4, it should be noticed that this metric is done on the absolute ascending order.

$$\text{RANKING} = \frac{4}{6}100 = 67\,\% \tag{13.3}$$

The higher the value of the RANKING, the better the surrogate model can follow the underlying response trend. However, this criteria in isolation is insufficient to determine the overall accuracy of the model.

13.7 Optimization Case Studies

Two optimization case studies were chosen to evaluate the application of surrogate models in turbo-machinery component design optimization:

- Mono-objective optimization of the operating conditions of a turbine cascade.
- Mono-objective optimization of a turbine labyrinth seal.

The optimization procedure consisted in two consecutive steps:

1. *Global search* of the optima, over all the design space.
2. *Local search* of the optima, refining the result obtained from the global search.

This optimization strategy combines both robustness and accuracy.

In particular, a *Genetic Algorithm* was chosen as the global optimizer, since it is a robust and reliable algorithm widely used in optimization [7, 12]. The subsequent

local optimization was done using the *Sequential Quadratic Programming* (SQP) method [13].

13.7.1 Turbine Cascade Case

The performance of a steam turbine cascade [14] was analyzed for different operating conditions employing an ALSTOM in-house CFD code.

The design space was defined by three input variables:

- Incidence Angle
- Inlet Total Pressure (for adjusting Mach Number)
- Fluid Viscosity (for adjusting Reynolds Number)

The objective of the optimization was to maximize the efficiency of the turbine profile. A dataset of 150 points was obtained running an Optimized Latin Hypercube DOE. Each simulation took about three minutes on a PC (Quad Core CPU running @ 2.66 Ghz, 3.25 GB RAM). Afterwards, the dataset was normalized in the range $[-1, 1]$ and randomly split into a training-set of 120 points and a test-set of 30 points.

13.7.1.1 Performance of Surrogate Models

Different setting approaches were adopted for each type of surrogate model. Five FFBP NNs were created using the *trainbr* algorithm and the OBS technique. It is worth reminding that FFBP NNs have a multi-modal performance function, therefore finding the best network configuration for a particular dataset usually requires to train the network from different initial weights/biases configurations. The same concept applies to the OBS technique, since the setting of the first oversized neural network influences the results of the subsequent pruning process. The validation subset was the 20 % of the training-set. The DTNN was built finding the optimal number of hidden neurons via "trial and error" procedure. Eventually, five DTNNs were created with the optimal architecture. As described for neural networks, five polynomial models were built using cross-validation with 10 subsets, in order to investigate how the random splitting affects the setting process. The same global order and interaction order were obtained for all five models, confirming that cross-validation is a robust procedure to set polynomial models. The interpolating RBF Networks were built only once, using the iterative procedure previously described in Sect. 13.3, with 20 steps. However, the setting procedure for regressive RBF Networks was different. In fact, these models were tuned using the GA, which was set with a population of 30 individuals and 30 generations. In this case, five regressive RBF Networks were built for each basis function, resulting in broadly similar performance. Finally, five Kriging models were also built. The likelihood function employed to build the Kriging models was optimized setting the GA with a population of 50 individuals and 100 generations. The adopted setting configuration produced almost

Table 13.1 Surrogate models performance on the blade test-set

	Test RMSE (%)	Test r^2	Test RANKING (%)
FFBP NN OBS	0.505	0.9996	90.00
FFBP NN *trainbr*	3.069	0.9848	56.67
DTNN	0.712	0.9993	93.33
RBF G	1.908	0.9943	73.34
RBF IM	1.668	0.9965	73.34
RBF M	1.803	0.9965	76.67
Reg. RBF G	1.740	0.9954	73.34
Reg. RBF IM	1.620	0.9962	73.34
Reg. RBF M	1.501	0.9967	73.34
Kriging	1.888	0.9960	66.67
Reg. Kriging	1.034	0.9983	76.67
Polynomial	1.389	0.9967	83.34

G Gaussian, *IM* inverse-multiquadrics and *M* multiquadrics

Table 13.2 Optimized and validated results for the blade study case

	Incidence angle (deg)	Tot. inlet pressure (*bar*)	Fluid viscosity (Ns/m^2) (10^{-6})	Optimized solution (−)	Validated solution (−)
FFBP NN OBS	−27.4056	190.10	1.8	0.93319	0.93378
FFBP NN *trainbr*	−8.4097	188.06	1.8	0.93115	0.93360
DTNN	−24.2300	189.31	1.8	0.93330	0.93383
RBF G	3.3989	191.63	1.8	0.93213	0.93304
RBF IM	−32.2721	189.11	1.8	0.93247	0.93376
RBF M	−32.0508	189.01	1.8	0.93250	0.93377
Reg. RBF G	−23.5060	190.59	1.8	0.93240	0.93375
Reg. RBF IM	−22.8310	190.87	1.8	0.93219	0.93372
Reg. RBF M	−32.5122	188.99	1.8	0.93267	0.93376
Polynomial	−14.2838	190.56	1.8	0.93389	0.93364
Reg. Kriging	−28.9457	189.05	1.8	0.93409	0.93382

The maximum efficiency in the DOE dataset is 0.93099, *G* Gaussian, *IM* inverse-multiquadrics and *M* multiquadrics

identical models, the small differences were related to the GA obtaining only the neighbourhood of the maximum as opposed to maximum of the likelihood function.

For each type of surrogate model, only the best performing model was chosen for the comparison summarized below.

It can be noticed from Table 13.1 that all the models performed very well, with low values for the NRMSE and high values for r^2 and RANKING.

Once the surrogate models were built, then it was possible to use them to evaluate all the other input configurations required by the optimizer algorithm. The constraints

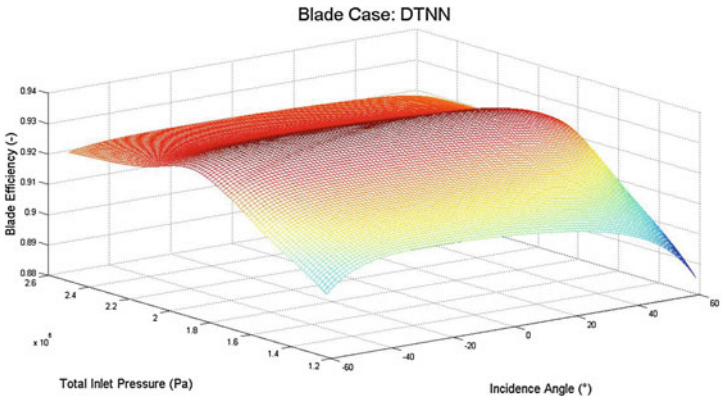

Fig. 13.1 Blade case—DTNN

of the optimization were represented by the design space boundaries, which were defined by the highest and lowest values of each input variable in the DOE.

As can be seen in Table 13.2, all the surrogate models gave very similar validated solutions. In particular, the DTNN produced the best results, which was chosen to plot a graphic representation of the problem, fixing the viscosity to its lowest value in the dataset (see Fig. 13.1). It is also interesting to note that the DTNN had the best RANKING score on the test-set. However, almost the same validated solution was obtained with the Regressive Kriging, which was also definitely far quicker to set than DTNNs and FFBP NNs. Polynomial model and RBF Networks were also quicker and easier to set than neural networks. In the opinion of the authors, the simplicity of the setting process should be always considered in the assessment of surrogate modelling methods. In practical applications, a quick-to-set surrogate model should be preferred to other models with time consuming and non-robust setting processes, especially when the results are almost the same, as in this case.

Finally, it should be considered that the efficiency improvements obtained from the initial DOE were small from the numerical point of view, but very important in engineering design.

13.7.2 Turbine Seal Case

The leakage of a labyrinth seal of the high-pressure stage of a steam turbine was evaluated via CFD simulations, which were performed with the commercial code ANSYS FLUENT [15]. Seven geometric parameters were originally chosen in order to determine the key variables for prediction of leakage, such as fin height, thickness, angle, etc. These input variables were screened using full-factorial DOEs and Pareto Charts (based on polynomial regression). Finally four top parameters were selected to be included in the surrogate modelling and subsequent model based optimization (minimization) of seal leakage:

- a_1, Angle parameter
- a_2, Angle parameter
- L_1, Length parameter
- L_2, Length parameter

A full-factorial DOE of 5 levels per variable (resulting in $5^4 = 625$ points) was originally planned to investigate the problem, but some simulations failed due to technical issues in the CFD solver, obtaining a reduced dataset composed of 517 points. Each simulation took about eight minutes on a PC (12 Core CPU running @ 2.92 Ghz, 24 GB RAM). The dataset was randomly split into a training-set of 414 points and test-set of 103 points. The surrogate models were built adopting the same methodology employed for the turbine cascade case.

The subsequent optimizations were run setting the GA with 100 individuals and 30 generations, and allowing a maximum of 30 iterations for the local optimizer. As for the turbine cascade case, the design space boundaries were defined by the highest and lowest values of each variable in the DOE.

All the values shown in the tables and pictures regarding the seal case were normalized in the range $[-1, 1]$, for the purpose of protecting commercially sensitive information.

13.7.2.1 Performance of Surrogate Models

Table 13.3 shows that the surrogate models did not perform very well in this case, scoring high NRMSE values and low r^2 values. Also the scores for the RANKING were very low. In the opinion of the authors, the RANKING criteria should be applied to the cases where surrogate models perform very similarly, as for the blade case.

Table 13.3 Surrogate models performance on the seal test-set

	Test NRMSE (%)	Test r^2	Test RANKING (%)
FFBP NN OBS pruned	10.490	0.7554	5.83
FFBP NN *trainbr*	10.181	0.7704	9.71
DTNN	11.903	0.6871	5.83
RBF G	13.339	0.6072	8.74
RBF IM	11.880	0.6924	8.74
RBF M	11.530	0.7103	7.77
Reg. RBF G	11.246	0.7267	8.74
Reg. RBF IM	11.420	0.7164	7.77
Reg. RBF M	11.544	0.7093	7.77
Polynomial	11.573	0.7108	7.77
Kriging	14.199	0.5573	9.71
Reg. Kriging	10.749	0.7504	7.77

G Gaussian, *IM* inverse-multiquadrics and *M* multiquadrics

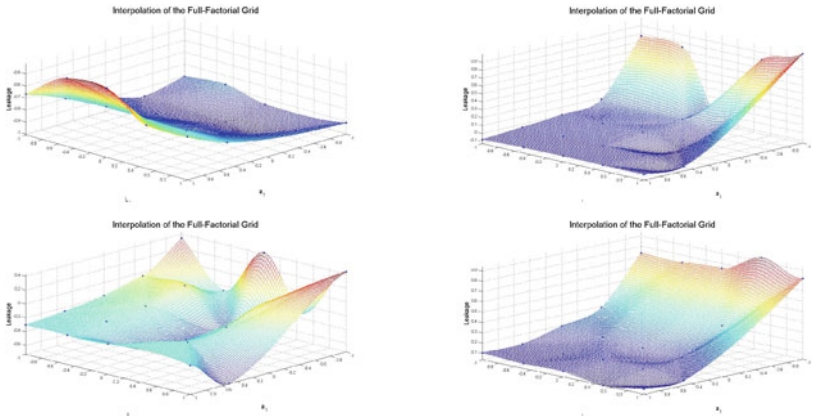

Fig. 13.2 Full factorial cubic spline interpolations: a_2 and L_2 are fixed to different values

The validated solutions were generally worse than the "best solution" contained in the DOE (which was equal to -1 following normalization). However, very small reductions of the leakage were obtained with RBF Networks (surrogate model based optimal solution -1.001, validated CFD code solution -1.0067), but not enough to consider the optimization a success.

It was also found that most of the optimized solutions were found for the lowest value of a_2 and L_2. Recalling that a generic full-factorial DOE consists in a multi-dimensional grid, Fig. 13.2 was generated fixing some variables and using MATLAB cubic spline interpolation [13].

As can be seen from Fig. 13.2, the underlying function shows very different scenarios varying the values of the same fixed variables, making it difficult to be modelled even with a full-factorial DOE of 517 points. This behavior is probably due to the fact that the input variables are highly correlated. However, it should be noted that the cubic spline interpolation does not correspond to the true function, which is obviously unknown, and the underlying function might be even more complex.

In addition, Figs. 13.3 and 13.4 show a comparison between some surrogate models and the corresponding full factorial cubic spline. It is clear from Fig. 13.3 that

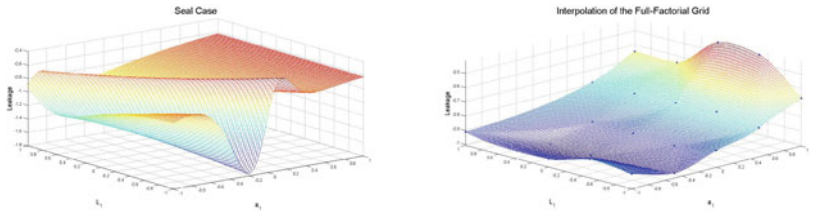

Fig. 13.3 FFBP NN OBS pruned versus full factorial cubic spline interpolation, $a_2 = -1$ and $L_2 = -1$

FFBP NNs pruned with OBS overfitted the data. In fact, it appeared that the high flexible structure of FFBP NNs can fit the data in a large variety of ways (i.e. with very different configurations for weights/biases), generating approximations with good values for NRMSE and r^2 but also with very strange shapes. On the contrary, the FFBP NN trained with *trainbr* gave a good representation of the problem, as can be seen in Fig. 13.4. In fact, the *trainbr* algorithm increases the level of regression of neural networks, making them smooth and less prone to overfit [16]. RBF Networks and Kriging showed less flexibility than neural networks, since their structure is directly anchored to the points in the dataset.

After these observations, it was decided to adopt a different strategy for the seal case, aimed to obtain an optimized solution similar to the best solution contained in the full-factorial DOE of 517 points, but using less CFD computations.

13.7.2.2 Further Investigation with Alternative Datasets

An Optimized Latin Hypercube DOE of 100 points was run with the objective to gather information over all the design space using less points. Again, some points failed to produce a result, obtaining a reduced dataset composed of 94 points. Surrogate models were built with the new dataset, employing all the points as a training-set.

The new validated solutions were not better than the solutions obtained in the previous optimization. However, it can be noticed that the majority of the regressive models gave again the optimized solutions for the lowest value of a_2 and L_2.

It was therefore decided to run a further Optimized Latin Hypercube DOE of 50 points with a_2 and L_2 fixed to their lowest value, in order to reduce the dimensionality of the problem. Finally a dataset composed of 47 points was obtained. The surrogate models were built employing all the 47 points as the training-set.

As can be seen in Table 13.4, the Kriging model and the Regressive RBF Gaussian Network improved the best solution contained in the first dataset of 517 points. Thus, the computational budget was reduced from 600 points to 150 points. Unfortunately the FLUENT solver failed to converge at the optimized solution of the Regressive Kriging.

In addition, a further full-factorial DOE of 400 points (20 levels per variable) was run fixing a_2 and L_2 to their lowest value, in order to investigate in detail the

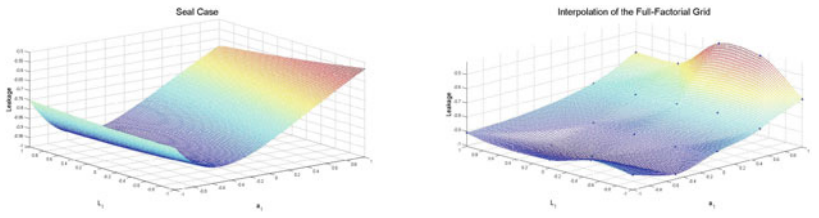

Fig. 13.4 FFBP NN *trainbr* versus full factorial cubic spline interpolation, $a_2 = -1$ and $L_2 = -1$

Table 13.4 Optimized and validated results for the seal study case using the dataset composed of 47 points

	a_1 (−)	L_1 (−)	Optimized solution	Validated solution
OBS pruned	−0.9350	−0.4219	−1.2282	−0.8986
trainbr	−0.5230	0.0042	−1.0246	−0.9818
DTNN	−0.7167	0.0274	−1.0073	−0.9966
RBF G	−0.7374	0.6752	−1.0303	−1.0013
RBF IM	−0.5981	−0.0644	−1.0210	−0.9943
RBF M	−0.6270	−0.0109	−1.0195	−1.0188
Reg. RBF G	−0.7383	0.6797	−1.0292	−1.0196
Reg. RBF IM	−0.5976	−0.0651	−1.0210	−0.9982
Reg. RBF M	−0.3396	0.1970	−1.1035	−0.9693
Kriging	−0.6989	0.5896	−1.1035	−1.0286
Reg Kriging	−0.5494	1	−1.0054	N/A
Polynomial	−0.7461	0.7949	−1.0454	−0.9824

Leakage normalized w.r.t. the first dataset, *G* Gaussian, *IM* inverse-multiquadrics and *M* multi-quadrics

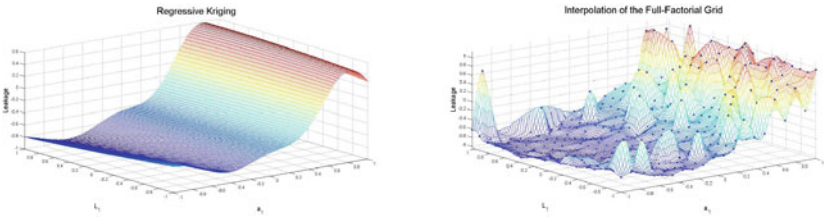

Fig. 13.5 Regressive Kriging versus FF interpolation, DOE 50 points

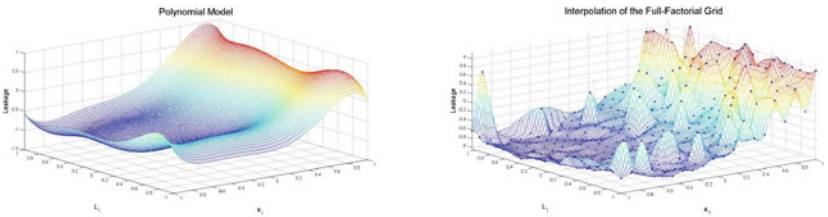

Fig. 13.6 Polynomial model versus FF interpolation, DOE 50 points

morphology of the underlying function. The final dataset was composed of 384 points (with 16 points failing to converge).

As can be seen in Figs. 13.5 and 13.6, regressive models developed with 47 points were capable to detect the general trends of the real problem, which was highly irregular with many peaks (see the full factorial cubic spline interpolations). In particular,

the Polynomial model was able to define the "borders" of the underlying function very well. On the contrary, FFBP NN pruned with OBS and DTNN overfitted the data.

In engineering design, the visualization of a problem is extremely useful, since it can provide an indication of promising regions that may yield a robust optimal design. Flat zones with stable performance will be preferred to peaky zones, where small variations of the input variables lead to high variation of the output. For example, the blue valley in Fig. 13.6 represents a stable zone, where small variations of a_1 and L_1 do not particularly affect the leakage. In fact, the geometric parameters defining a labyrinth seal are subject to manufacturing tolerances. Thus, it is clear that an important aspect in industrial design is managing the uncertainties, to find solutions which are insensitive to the stochastic fluctuations of the parameters (*Robust Design*) [10, 17].

Summarizing, the seal case was significantly more challenging than the blade case. FFBP NNs pruned with OBS and DTNNs performed poorly, overfitting the surface. Instead, the FFBP NNs trained with *trainbr* were able to detect the main trends of the underlying function. However, other surrogate models such as RBF Networks, Kriging and Polynomial, gave better results with less training. In particular, the Kriging model produced the best numerical result and the Polynomial model gave the best representation.

In addition, it appeared that a good strategy in optimization assisted by surrogate models may consist in:

1. Run a small global DOE, according to the available computational budget.
2. Build regressive surrogate models and visualize the problem where possible.
3. Validate the optimized solutions.
4. Evaluate the possibility of reducing the dimensionality of the problem, or at least to define a small promising zone in the domain.
5. Run a reduced/local DOE, according to the available computational budget.
6. Validate the new optimized solutions.

13.8 Conclusion

This paper has demonstrated the utility of Surrogate Models in turbo-machinery design optimization. In the first instance, different surrogate modelling methods should be used when dealing with unknown problems, in order to find the model that best fits a particular dataset. In addition, surrogate models should be assessed on the basis of their ease of configuration. From this point of view, FFBP NNs and DTNNs present too many drawbacks to be considered a valid methodology in turbo-machinery component design optimization. They did not show any clear advantage compared to other methodologies in terms of accuracy, but their setting process presented many issues. However, neural networks are widely applied in control engineering and signal processing, where their flexibility represents a benefit in modelling of dynamic systems.

Finally for the considered case studies, Kriging models were assessed as being the most promising surrogate model among those evaluated in this paper, combining high performance with a relatively easy setting process.

References

1. Badjan G (2013) Evaluation of surrogate modelling methods for turbo-machinery component design optimization. MSc Thesis in Mechanical Engineering, University of Trieste, Italy
2. Haykin S (2005) Neural networks: a comprehensive foundation. Pearson Education, Hamilton
3. Hassibi B, Stork DG (1993) Optimal brain surgeon and general network pruning. IEEE Int Conf Neural Netw 169:293–299
4. Noorgard M (2000) Neural network based system identification toolbox. Technical Report 00-E-891, Technical University of Denmark
5. Chiang CC, Fu HC (1994) The classification capability of a dynamic threshold neural network. Pattern Recogn Lett 15:409–418
6. Pediroda V (2001) Utilizzo e sviluppo di tecniche "Soft-Computing" per lo studio e la progettazione di macchine a fluido. Ph.D. thesis in Chemical and Energy Technologies, University of Udine, Italy
7. Poloni C, Giurgevich A, Onesti L, Pediroda V (2000) Hybridization of a multi-objective genetic algorithm, a neural network and a classical optimizer for a complex design problem in fluid dynamics. Comput Methods Appl Mech Eng 186:403–420
8. Rippa S (1999) An algorithm for selecting a good value for the parameter *c* in radial basis function interpolation. Adv Comput Math 11:193–210
9. Forrester A, Sóbester A, Keane A (2008) Engineering design via surrogate modelling. Wiley, University of Southampton
10. Keane A, Nair P (2005) Computational approaches for aerospace design: the pursuit of excellence. Wiley, New York
11. Hastie T, Tibshirani R, Friedman J (2001) The elements of statistical learning. Springer, New York
12. Goldberg DE (1989) Genetic algorithm in search, optimization and machine learning. Addisons-Wesley
13. MATLAB (2008) User guide. The MathWorks Inc.
14. Perdichizzi A, Dossena V (1993) Incidence angle and pitch-chord effects on secondary flows downstream of a turbine cascade. J Turbomach 115:383–391
15. ANSYS FLUENT (2008) User guide. ANSYS, Inc.
16. Demuth H, Beale M (2000) Neural network toolbox: for use with MATLAB. User's guide, Version 4, The MathWorks Inc.
17. Pediroda V, Poloni C (2006) Approximation methods and self organizing map techniques for MDO problems. Department of Mechanical Engineering, University of Trieste, Italy