Surrogate Approaches for Aerodynamic Section Performance Modelling

Andy J. Keane[†] and Ivan I. Voutchkov
University of Southampton, Southampton, SO17 1BJ, England, United Kingdom.

Abstract

The use of surrogate models (response surface models, curve fits) of various types (radial basis functions, Gaussian Process models, neural networks, support vector machines, etc.) is now an accepted way for speeding up Design Search and Optimization (DSO) in many fields of engineering that require the use of expensive computer simulations, including problems with multiple goals and multiple domains. Surrogates are also widely used in dealing with Uncertainty Quantification (UQ) of expensive black-box codes where there are strict limits on the number of function evaluations that can be afforded in estimating the statistical properties of derived performance quantities. In this paper we compare and contrast a wide range of surrogate types on an aerodynamic section data set that allows for design variation, manufacturing uncertainty and damage in service, all solved with a high quality industrial strength Reynolds Averaged Navier Stokes (RANS) solver. We examine speed of training and model quality for different sizes of problem up to one where there are 26 input variables and nearly half a million CFD results in the available data set.

1. Introduction

When setting out to build a new surrogate the analyst is presented with a bewildering range of possible modelling methods: Fernández-Delgado *et al* [1] consider 77 regression methods belonging to 19 families across 82 data sets for example; the Matlab Statistics and Machine Learning Toolbox Regression Learner App contains 19 base methods, without considering the extensive range of neural network approaches available in their Deep Learning Toolbox (see documentation at www.mathworks.com). The discussion provided in such references and the documentation supplied with typical tools makes clear that it is very difficult to know, *a priori*, which method will work best for any given data set. Indeed the Regression Learning App includes the capability to try all 19 of its methods at a single button push, so as to carry out a full survey, and this is before one even begins to adapt the various control parameters peculiar to each method.

In this paper we explore the available range of methods in the Matlab toolboxes plus a few of our own codes, when applied to an aerodynamic section design problem where we have collected nearly 500,000 CFD results for differing geometries. While this study is nothing like so extensive as the review presented in [1] it does apply it to an industrial strength aerodynamics CFD code with a very large data set and it shows that those methods that work well across a very broad spectrum of problems may not be the best choice for aerodynamic robust design optimization problems.

The paper is laid out as follows: in section 2 we introduce a real two-dimensional CFD design case and show how this design can be modified by design or uncertainty and analysed with a RANS based CFD approach. In section 3 we apply all the methods in the well-known and easy to use Matlab Regression Learner App to data coming from this analysis chain. Then in section 4 we compare a number of more advanced surrogate approaches before moving on in section 5 to demonstrate how modern neural networks can effectively model hundreds of thousands of analysis results in a cost effective and reasonably accurate manner. In section 6 we apply these results to the problem of robust design optimization before making some notes on reproducibility in section 7 and drawing our conclusions in section 8.

2. A Simple Design Problem

We first introduce a practical but simple aerodynamic design problem: gas turbine blade section design. It is, of course, commonplace to use surrogate based design optimization methods to refine the shape of

[†] Professor of Computational Engineering; ajk@soton.ac.uk.

aerodynamic sections when designing new gas-turbines. Here we assess a range of surrogate modelling approaches on a gas-turbine aerodynamic section model introduced previously ([2] and [3]) which focused on the effects on gas turbine compressor blade performance of foreign object damage, erosion and uncertainty in manufacture, all at a single operating condition.

In the previous papers we compared and contrasted five approaches to robust design optimization for this problem. First, we simply optimized a baseline geometry for nominal performance using direct [4] and surrogate based search [5]. Then we employed a noisy phenotype [6] approach to allow for possible sensitivity in the mean performance but did not explicitly seek a trade-off between mean performance and robustness. Thirdly, we used the NSGAII multi-objective evolutionary search [7] to deliberately construct a trade surface (Pareto front) adopting both direct and surrogate assisted approaches. Fourthly, we used a surrogate based multi-objective expected improvement formulation [8] to construct the trade-off using single-objective search methods. Finally, in that study we used co-Kriging based surrogates, which were shown to give the most efficient approach to robust design improvement amongst the methods tested[‡]. These studies generated a little over 89,000 separate geometries and just over 70,000 successful CFD analyses, which form the initial data set we will here attempt to model with surrogates. We have further augmented this set with an optimal Latin hypercube design of experiments run over the problem to generate a further 500,000 runs of which around 80% were successful. To begin with we outline for convenience the problem being studied; complete details can found in the references.

Geometry Modification

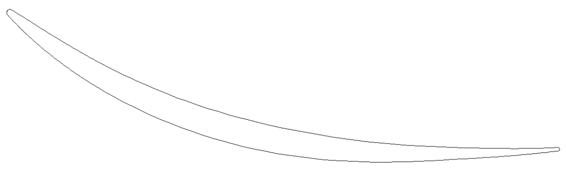
To carry out any form of automated aerodynamic section optimization some form of geometry modification is required. Here we adopt the Parametric Design And RApid Meshing (Padram) code (Shahpar and Lapworth [9]) that allows for a range of uncertainty studies that have been described elsewhere (Kumar *et al* [10]). In this process a base-line airfoil section shape is altered by the addition of a series of Hicks-Henne functions in various ways (Hicks and Henne [11]). Here this scheme is used to make four sets of changes to the base-line airfoil that allow for overall design improvement, modelling of manufacturing uncertainty, modelling of foreign object damage and modelling of flank erosion. Figures 1(a-f) illustrate the base-line airfoil section and each of these modification processes in turn. Note that all the changes are controlled such that the trailing edge form and angle are kept fixed as would be needed to ensure unchanged inflow angles for the next stage in the compressor. Ten Hicks-Henne functions are used for the design shape changes, distributed around the section, and this permits very significant alterations in geometry.

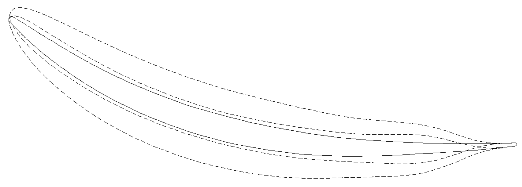
In the work that follows the normalised amplitude changes of these design variables are used. These are defined in such a way that a value of 0.5 generates the base geometry, with values less than this removing material from the section and those greater adding to it – full details may be found in the thesis of Kumar [12]. The other three sets of changes work in the same way but make much smaller shifts:

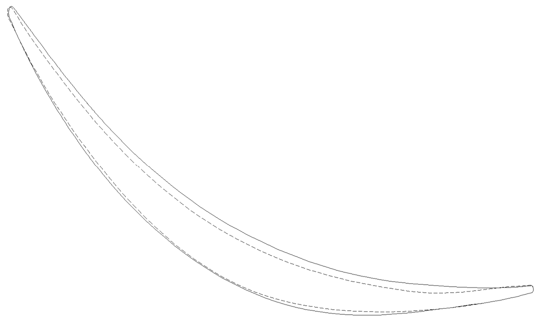
1. the manufacturing variability changes affect the entire section (again using ten Hicks-Henne functions distributed around the section, but now much smaller in scale) paying particular attention to the leading edge,
2. the foreign object damage changes are localised to the leading edge using a single Hicks-Henne function that can only remove material but can vary in position and shape as well as depth (a peak height setting of zero means no damage) and
3. the erosion changes are localised to the flank, again using a single Hicks-Henne function that can again vary in position and shape as well as depth and can only remove material (a peak height setting of zero means no damage) – it is applied where typical erosion events occur.

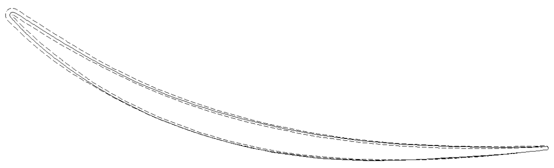
[‡] Co-Kriging of course requires multiple fidelities of analysis which are not considered in the present study.

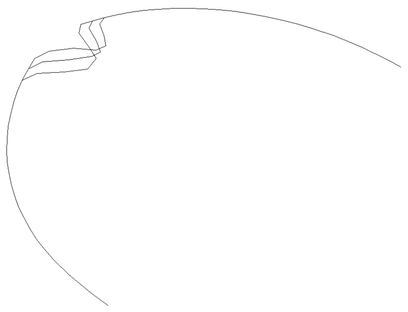
The modelling of the damage and erosion events is clearly far from accurate since the Hicks-Henne functions give smooth and continuous pockets whereas in reality these are typically irregular and sharp edged. However, as used here, the pockets still act to trip and disturb the flow in a way that is not unrealistic. Note that the flank erosion pockets are allowed to extend over a greater area of the blade than the damage pockets at the tip.


a)


b)


c)


d)


e)

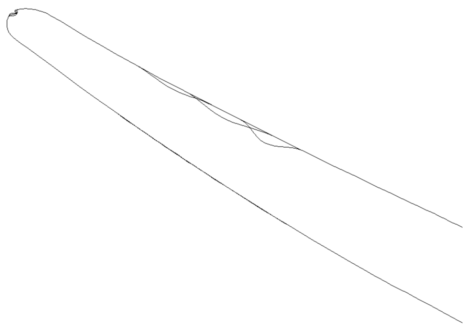
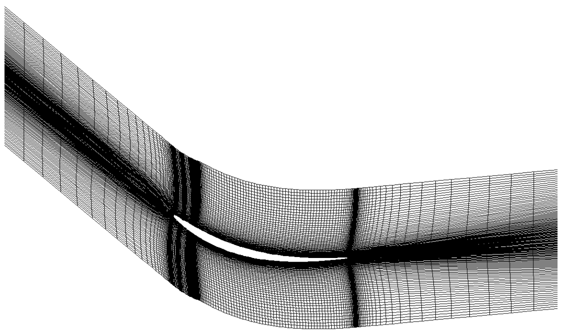
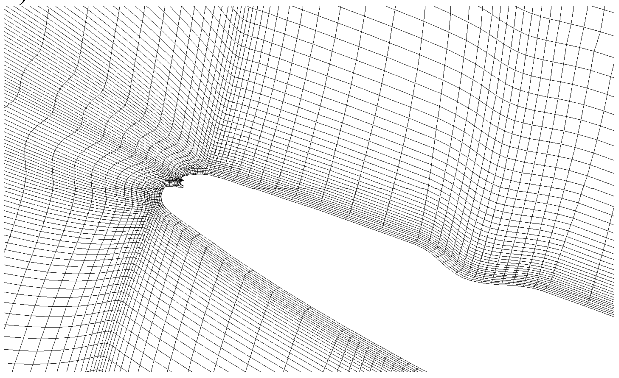

f)

Figure 1, a) base-line airfoil section; b) overall design variations maximum possible changes; c) typical design change (shown distorted for clarity); d) maximum range of changes aiming to model uncertainty in manufacturing processes; e) leading edge damage models; f) flank erosion models.


a)


b)

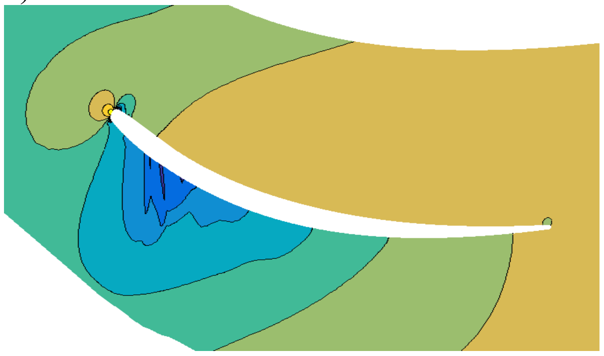

c)

Figure 2 – Padram meshes and resulting flow field; a) overall mesh, b) in way of leading edge and flank erosion and c) section profile and pressure field.

Analysis

The Padram geometry modification process also permits the automated adaptation of high quality "OCH" meshes around the airfoil section in a setting appropriate for application to gas-turbine blade design (here "OCH" refers to the shapes of the various mesh blocks, their being ring like, cup shaped or with legs). Figure 2 shows a typical mesh and the way that this mesh is distorted around the localised erosion geometries introduced when studying leading edge damage or flank erosion. Here the "O" mesh is four cells deep and slightly over 27,000 cells are used in total, along with the resulting flow field. The topology of this mesh is unchanged throughout the study presented here, so that any effects caused by re-meshing can be avoided. If geometry changes result in overly large distortions to the mesh the relevant run is aborted and appropriate action taken during subsequent surrogate modelling processes.

Having set up the mesh a full steady-state RANS solve can be carried out using the parallel version of the Hydra CFD code in around three minutes on a 12 core PC [13]. The section is run in isolation with boundary conditions of inlet temperature 290K, inlet total pressure 63400Pa, whirl angle -37.3° and outlet static pressure 52000Pa. Typical root mean square (RMS) flow residuals are 4.7E-9 and RMS turbulence residuals 2.4E-15. Here we use pressure losses as the main design goal. These are all normalized by that of the chosen base-line blade, to preserve confidentiality – it therefore has a normalized loss coefficient of unity. Full validation studies of this meshing and solution process are also presented by Kumar [12], where it is shown that the results achieved are reasonable for the kind of statistical work being presented here[§].

If this base-line blade is then made subject to uncertainties in leading edge damage, flank erosion and manufacturing it is possible to estimate the mean pressure loss and standard deviation in this loss using simple Monte-Carlo approaches. To do this 100 independent perturbations are made to each of the three sets of Hicks-Henne functions defining these three sets of uncertainties, using a standard LPτ space filling design of experiment (Statnikov and Matusov [14]) and averages taken. Such schemes generally give much faster convergence that pure random ones[**].

The perturbations made are significantly beyond the extremes in tolerance band typically found in manufacture or in service, but they provide a good measure of the resilience of a section design – such variations would be unlikely in practice. This calculation of course requires 100 times as much processing effort as finding the nominal performance, but provides a datum for comparison – here the normalised mean loss coefficient is 1.116 with a standard deviation of 0.115 (i.e., the mean loss is 12% worse than the deterministic calculation on the nominal base-line shape). Figure 3 shows the normalised convergence history of this calculation – note that since larger values of the perturbations almost inevitably lead to greater losses, the mean and standard deviation both rise until the 16 dimensional hyper-cube being sampled is fully explored by the pseudo random number sequence in use – here after around 50 calculations – thereafter the convergence is essentially unbiased.

Unfortunately the use of such a large sample size in design searches cannot generally be justified, particularly if three-dimensional analyses are being considered. The extensive use of this approach in the earlier papers did, however, generate a large and useful data set of results, albeit one that is not evenly spaced over the full search space. Of the 89,115 geometries initialled generated, 70,609 led to successful (i.e., fully converged) CFD runs[††]. Full flow field data is available for all the

[§] Note that using this mesh it is accepted that accurate modelling of the flow in the damage pockets cannot be achieved with this number of cells – in this study the intention is that these regions should act as trips to the flow as rough edge pockets would in practice. Clearly, detailed flow analysis around an eroded section that attempted to capture all of the flow physics would require a significantly increased mesh count with consequent increases in run time.

[**] It is also possible to design a sampling sequence specifically to match the problem being studied so as to accelerate the convergence of statistics – the so called quasi-Monte-Carlo approach.

[††] In some cases the significant geometry changes involved lead to unsteady flows that do not converge successfully.

successful runs. To augment this initial set of data a further sequence of 500,000 optimal Latin hypercube designs have been tested over the same variable ranges, adding 393,903 further converged results to our datasets.

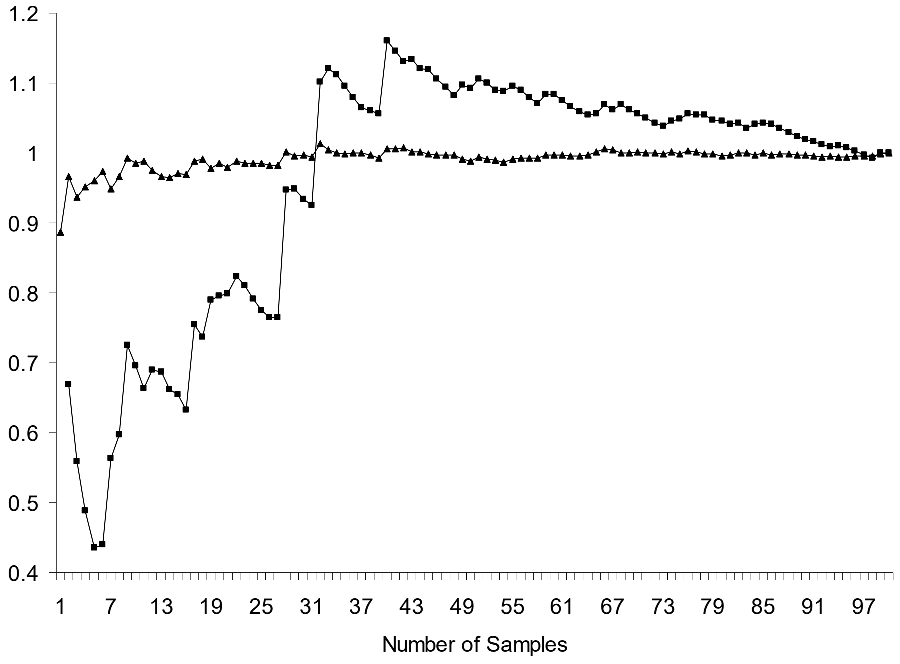


Figure 3 – convergence of averaging process used in assessing uncertainties in shape (triangles show normalised mean and squares normalised standard deviation).

It should, however, be noted that it is current practice when designing blades for aero-engines to work mainly with three-dimensional CFD models and over a range of operating points – the use of two-dimensional approaches at a single condition has been adopted here simply to limit the computational cost of the study.

3. Initial Surrogate Modelling Using Matlab Regression Learner App

To begin our study we simply loaded all the successful CFD results from our initial runs in one large table of 26+1 by 70,609 items into the Matlab Regression Learner App and asked it train all the available models, in parallel, with their default settings, without PCA filtering and using Holdout validation with 25% held out (note that this App does not currently include neural network models, so these are considered later on in this paper). The results from this initial study are presented in Table 1 while Figure 4 plots the RMSE of the fitted models against the reported training time on log scales. Note that these training times will vary depending on the computing facility used and are typically *not* very repeatable since the app tries to run as many of the methods at once as the available hardware will support – they merely serve to give order of magnitude estimates of the costs involved[‡‡]. Also we restrict our performance metrics here to RMSE and R^2 for simplicity – there are of course a great variety of performance metrics one could use, but these two are perhaps most familiar to the general reader. Apart from the very expensive stepwise linear model the training costs for the linear, tree and ensemble methods are all quite modest, those for the support vector machine models taking from tens of minutes to nearly two hours and those for the GPR methods tens of hours to days. Even then, by default, the GPR tools only use 2,000 points permed from the available data for training purposes and 10,000 for prediction[§§].

[‡‡] Here all runs are allocated 25 cores on a 32 core machine and make best use of these according to the method in use.

[§§] Note that although the GPR tools are using less data, the power of such methods often means they can be competitive despite working on data subsets – this is why they are the preferred approach of many workers when only limited amounts of data are available.

Method	Variant	#	Training Time (secs)	RMSE	R^2
Linear Regression	Linear	1	48.9	12.5%	0.48
	Interactions Linear	2	76.2	10.8%	0.61
	Robust Linear	3	25.6	12.6%	0.47
	Stepwise Linear	4	251370.0	10.8%	0.61
Tree	Fine Tree	5	37.0	11.0%	0.60
	Medium Tree	6	16.3	10.3%	0.65
	Coarse Tree	7	8.6	10.2%	0.66
Support Vector Machine	Linear SVM	8	1956.8	12.7%	0.46
	Quadratic SVM	9	3845.1	9.2%	0.72
	Cubic SVM	10	5679.9	8.5%	0.76
	Fine Gaussian SVM	11	4258.6	13.2%	0.42
	Medium Gaussian SVM	12	3954.5	8.0%	0.78
	Coarse Gaussian SVM	13	1428.9	9.5%	0.70
Ensemble	Boosted Trees	14	66.9	12.5%	0.48
	Bagged Trees	15	94.5	8.7%	0.75
Gaussian Process Regression	Squared Exponential GPR	16	40528.0	8.0%	0.78
	Matern 5/2 GPR	17	43907.0	7.8%	0.79
	Exponential GPR	18	140970.0	7.9%	0.79
	Rational Quadratic GPR	19	52298.0	7.9%	0.79

Table 1 – Surrogate training results for 70,609 CFD runs; initial Matlab Regression Learner App run.

Even so, it is apparent that getting a good balance between training time and predictive accuracy is not straightforward: some expensive methods such as the Fine Gaussian SVM give very poor results which are easily outperformed by the extremely fast to train Interactions Linear, Linear Regression model. Of the standard methods in the toolkit the Medium Gaussian SVM represents a good compromise. While the Gaussian Process Regression models offer slightly better fits they are really too expensive to train to justify the gains in accuracy offered. Of the cheap to train models, the Ensemble Bagged Tree is easily the best approach – it is also extremely widely used in the literature. Figure 5 shows the results for fitting the Matern 5/2 GPR which is the most accurate of the standard methods.

The main aim of the Regression Learner App is to provide a convenient initial assessment of possible modelling approaches. After it has been run the app will build customized modelling code for the user from any of its built in models, which can be further developed using a very wide range of options.

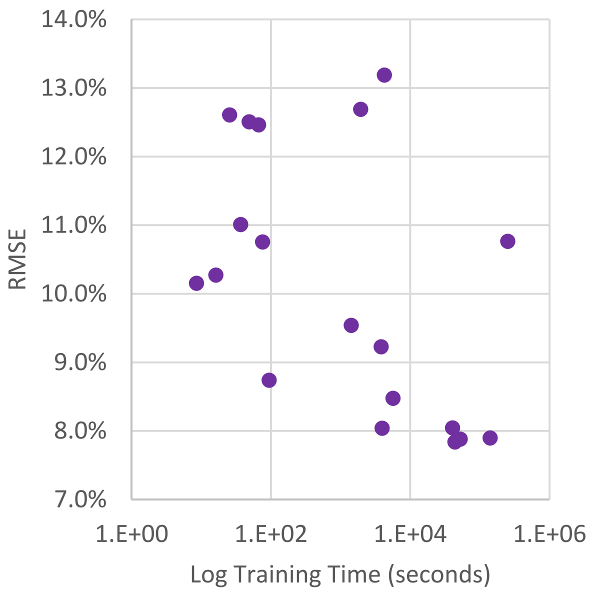


Figure 4 – Surrogate training results for 70,609 CFD runs; initial Matlab Regression Learner App run, see also Table 1.

These individual codes also permit a more detailed study of the performance of the various approaches. In particular they permit repeated runs in controlled conditions to more accurately assess speed of training and prediction performance. Here we carry forward just three of the 19 approaches that have promising performance in the initial survey: Medium Gaussian SVM, Bagged Trees and Matern 5/2 GPR. We rerun these three models by dividing the initial data set into two parts, using a random 75% permutation for training and then the remaining 25% to predict the performance.

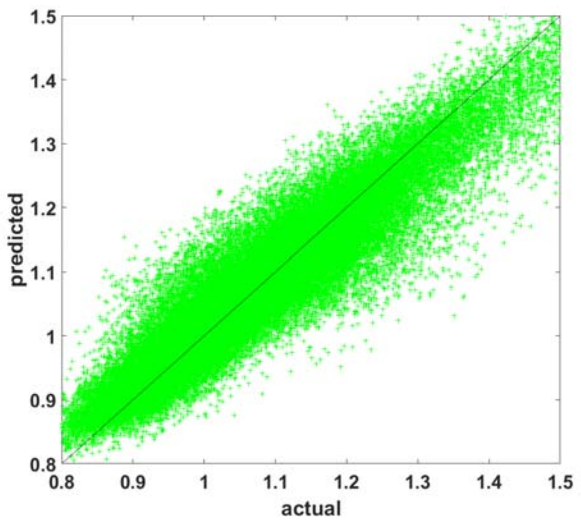


Figure 5 – Results for fitting the Matlab Matern 5/2 GPR surrogate model to 70,609 CFD runs with default settings.

The RMSE values for these three methods are essentially as before but now the individual training times can be more accurately measured, see Table 2 and Figure 6. It is also worth noting that some methods are more suitable for parallel computing options than others. For example by switching to the dedicated "TreeBagger" code within Matlab, significant parallel speed-ups can be achieved.

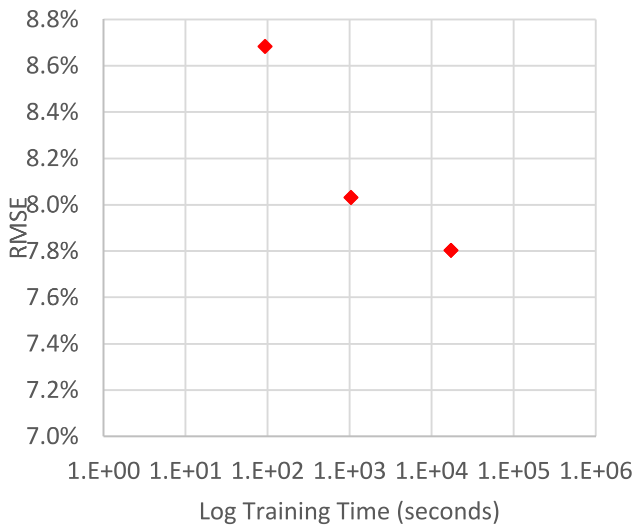


Figure 6 – Surrogate training results for 70,609 CFD runs from three individual Matlab methods run independently with default settings.

Method	Variant	#	Training Time (secs)	RMSE	R^2
Support Vector Machine	Medium Gaussian SVM	22	1037.1	8.0%	0.782
Ensemble	Bagged Trees	25	92.8	8.7%	0.748
Gaussian Process Regression	Matern 5/2 GPR	27	17150.2	7.8%	0.794

Table 2 – Surrogate training results for 70,609 CFD runs from three individual Matlab methods run independently with default settings.

4. Further Tuning and Other Surrogates

Having assessed the basic regression surrogates available in the Matlab App with default settings, we next turn to more specialized approaches. These either involve customizing the various parameters in some of the preceding methods or using alternative ones built from lower level Matlab capabilities, following logic derived from the literature. Here we consider five alternatives:

1. Medium Gaussian SVM with auto optimization of selected hyperparameters.
2. Bagged Trees but with variations to the default leaf and forest sizes.
3. Matern 5/2 GPR with auto optimization of selected hyperparameters.
4. A custom built neural network (NN) with five, six or eight hidden layers (the number of units in each hidden layer being 17, 13, 10, 6 and 3; 18, 15, 12, 9, 6 and 3 or 20, 17, 15, 12, 10, 7, 5 and 2, respectively. Training is carried out using the Levenberg-Marquardt approach using MSE as performance metric and up to 200 epochs of training, see the Matlab on-line Deep Learning Toolbox documentation at www.mathworks.com and [15].
5. A series of custom built Kriging GPR models with adjoint accelerated

optimization of all hyperparameters allowing various more complete data sets to be used in training and prediction (see [16] – differing sized subsets of the data are used for training and prediction).

6. A patched collection of the custom built Krigs based on a forest division of data (there are many ways of partitioning data for use with surrogates such as Krigs, see for example [17]).

The results from using these model are shown in Table 3 and Figure 7 (which includes the earlier results). For the three Matlab Learner App methods there is little change in predictive power and for the SVM and GPR models the attempt at tuning comes at high computational cost. Note that the patched Krig GPR is significantly faster than the direct Krig GPR models but this comes at the expense of worse predictive power: essentially given 26 dimensions the partitioning process is unable to identify well separated features in the data that can be used to sub-divide the problem.

Method	Variant	#	Running Time (secs)	RMSE	R^2
Support Vector Machine	Medium Gaussian SVM	32	49,547.6	8.0%	0.782
Ensemble	Bagged Trees	35	7,105.8	8.5%	0.757
Gaussian Process Regression	Matern 5/2 GPR	37	2,580,213.9	7.9%	0.792
Neural Network (mean of 50 runs)	17/13/10/6/3 NN	41	117.8	7.6%	0.805
	18/15/12/9/6/3 NN	42	157.9	7.6%	0.804
	20/17/15/12/10/7/5/2 NN	43	253.6	7.7%	0.800
OptimatV2 Kriging GPR	Krig 2,000trng/10,000pred	44	259,131.5	8.6%	0.750
	Krig 4,000trng/20,000pred	44a	1,174,735.4	8.0%	0.787
	Krig 4,000trng/50,000pred	44b	1,346,299.0	8.0%	0.786
	Patched Krigs	51	14,400.0	9.8%	0.682

Table 3 – Surrogate results for 70,609 CFD runs from tuned Matlab methods and other approaches, including time to predict on testing set.

In terms of accuracy versus cost nothing comes close to the power of the neural networks. This performance is possible because they have greater flexibility than the forest models, faster processing than the SVM approaches and are able to make effective use of all the available data, unlike the GPR models which, because of computational expense, even when using advanced adjoint schemes, are still forced to both train and evaluate with subsets of data or to partition the data across multiple models.

Note that although three levels of complexity are used in the NN models, their structures are relatively simple, with a tapering size for the hidden layers on progressing through the net, otherwise they use default Matlab settings. For this first set of data the simplest model works best, but as will be seen next, the deeper models become beneficial as more data is used for training.

5. Using Larger Data Sets

Because the Neural Network approach is so fast, we next consider using much larger data sets to make predictions. We therefore augment the original 70,609 runs with the results from 100,000, 200,000, 300,000, 400,000 and 500,000 optimal Latin hypercube samples over the 26 dimensional design space. These allow us to train with 149,473, 228,094, 306,826, 385,805 and 464,512 sets of converged CFD data, respectively (allowing for failed runs and including the initial 70,609 points).

Table 4 and Figure 8 (a, b) show these final sets of results for the three NN topologies considered (including those from Table 3). It is clear that the networks are able to gain useful benefit from the increasing amounts of data, showing an essentially inverse linear improvement in prediction error with data set size (i.e., RMSE is proportional to 1/data set size, converging to an absolute limit on RSME of 6.07% as the data set size tends to infinity). Note that as the data set size increases so the extra layers in the more complex networks come into their own.

Additionally, the running times tend to increase only weakly quadraticly in data set size. It is this kind of capability that has led to the recent explosion of interest in these methods in the field of machine learning.

NN Variant	# Data Points	#	Time (secs)	RMSE	R^2
17/13/10	70,609	41	117.8	7.60%	0.805
/6/3	149,473	41a	332.3	6.93%	0.826
	228,094	41b	733.1	6.61%	0.836
	306,826	41c	985.5	6.47%	0.840
	385,805	41d	1197.7	6.38%	0.843
	464,512	41e	1521.1	6.33%	0.844
18/15/12	70,609	42	157.9	7.63%	0.804
/9/6/3	149,473	42a	409.5	6.92%	0.826
	228,094	42b	1005.0	6.57%	0.838
	306,826	42c	1510.1	6.40%	0.843
	385,805	42d	1966.6	6.30%	0.846
	464,512	42e	2738.9	6.24%	0.848
20/17/15	70,609	43	253.6	7.69%	0.800
/12/10	149,473	43a	644.2	6.88%	0.827
/7/5/2	228,094	43b	1292.2	6.56%	0.838
	306,826	43c	2142.0	6.37%	0.845
	385,805	43d	3110.7	6.25%	0.849
	464,512	43e	4043.5	6.18%	0.851

Table 4 – Surrogate training results for Neural Network approaches with varying data set sizes, averaged over 50 runs in each case.

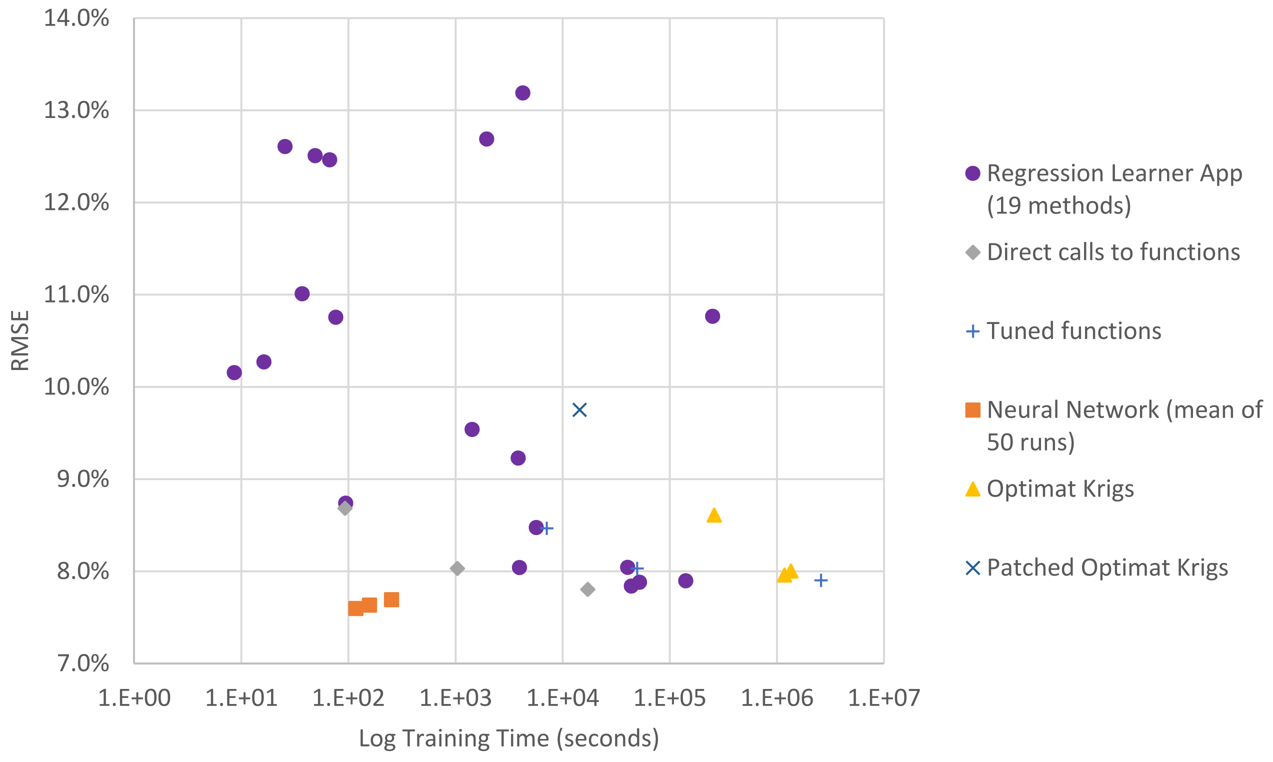
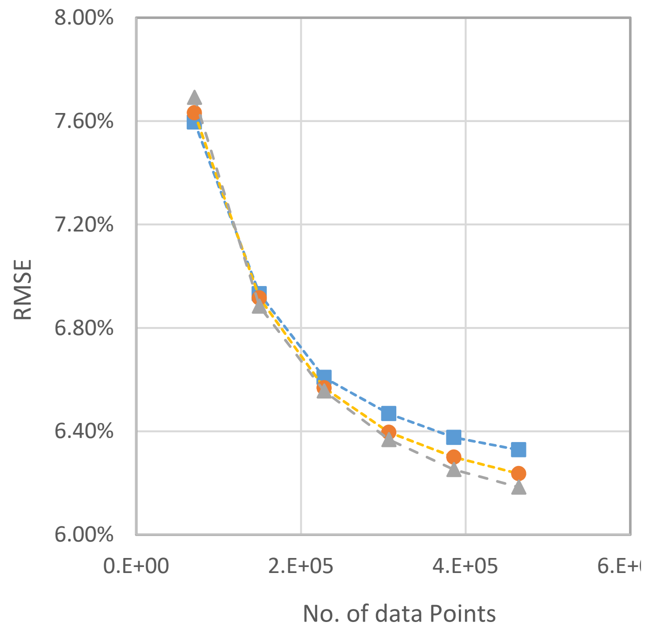


Figure 7 – Surrogate results for 70,609 CFD runs from all models.

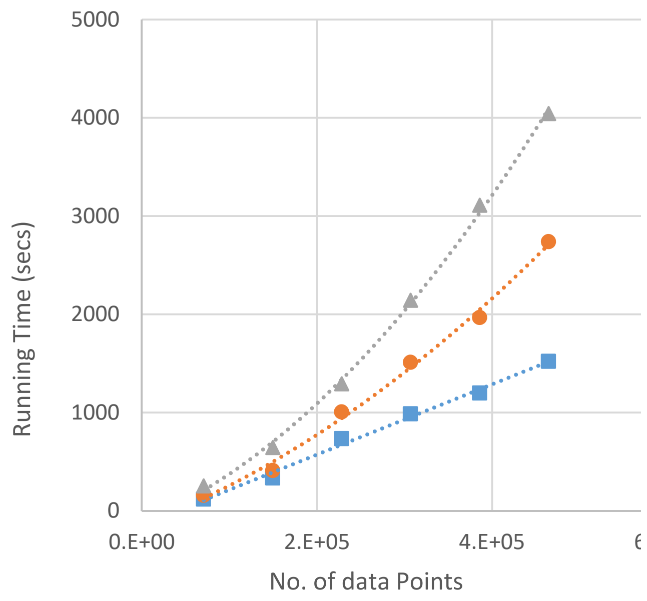
6. Using Surrogates to Support Robust Design

Although being able to accurately predict unseen test data is a robust measure of the overall quality of a surrogate it is also important to assess the results of surrogate construction on the analysis task at hand: here the problem of robust design optimization is considered. A surrogate can clearly be used in lieu of the CFD simulation either for design search and optimization (DSO) or uncertainty quantification (UQ) or both. For robust design optimization we use a combined approach (see for example the level one Kriging model of Dellino *et al* [18]).

In this process we set up a multi-objective search where we aim to optimize the mean behaviour of the aerodynamic section while at the same time reducing the variance in that performance (here the normalized stage pressure loss). At any point in the search we again use LPτ sampling (Statnikov and Matusov [14]) to carry out UQ across the noise variables in the available surrogate and then feed these into the classic NSGAII multi-objective DSO method (see Deb *et al* [7]) to try and find the Pareto front of mean loss versus standard deviation in loss. NSGAII is a long established global multi-objective search method that has a good track record of being able to establish high quality Pareto fronts.



a)



b)
Figure 8 – Surrogate training results for Neural Network approaches with varying data set sizes: a) RMSE, b) running time; five layer nets, squares; six layer nets circles, eight layer nets, triangles.

Clearly, if the surrogate can accurately model the underlying aerodynamic performance it will allow suitably accurate statistics to be computed, albeit at the cost of sampling multiple times. So to begin with, all our available data is made available to the surrogate construction and tuning process. Then 500 point low discrepancy sequence sampling over the noise variables is carried out on the surrogate, at any desired set of design variables, so as to return the desired predictions of mean and standard deviations of performance. These results are then used by the DSO method to find the best Pareto front of mean versus standard deviation for the given surrogate. Once the search is completed it is necessary to then confirm the statistics of any points on the final Pareto front with UQ applied directly to the CFD code, as the curse of dimensionality means that the combined surrogate will be unlikely to be able to supply completely accurate statistics directly in the 26 dimensional search space.

Figure 9 illustrates this process for the general case where no prior results are available – if an existing set of data exists then the initial Latin Hypercube step would be omitted. Whether or not additional CFD runs are added during the NSGAII search depends on the quality of the initial surrogate. If it does not give convincing RMSE and R^2 values then updates would be required to improve its quality (or perhaps an alternative surrogate might be considered)[***]. In either case a final set of CFD runs on the predicted results of interest is always advisable, noting that when working with mean and standard deviation (SD) this must involve sufficient UQ runs at *each* design of interest to provide convincing predictions.

Here we use our largest data-set of 464,512 converged CFD runs with the eight layer neural network to search for the estimated Pareto front, without updating with fresh CFD runs during the search and compare this to earlier results based on Optimat Krig GPR models (that because of computational cost have to be built on limited data subsets – here 4,000 in training and 10,000 in prediction).

[***] For example, our experience shows that if R^2 is less than 0.65, surrogate predictions will be of only limited value and further updating should be carried out or the model changed.

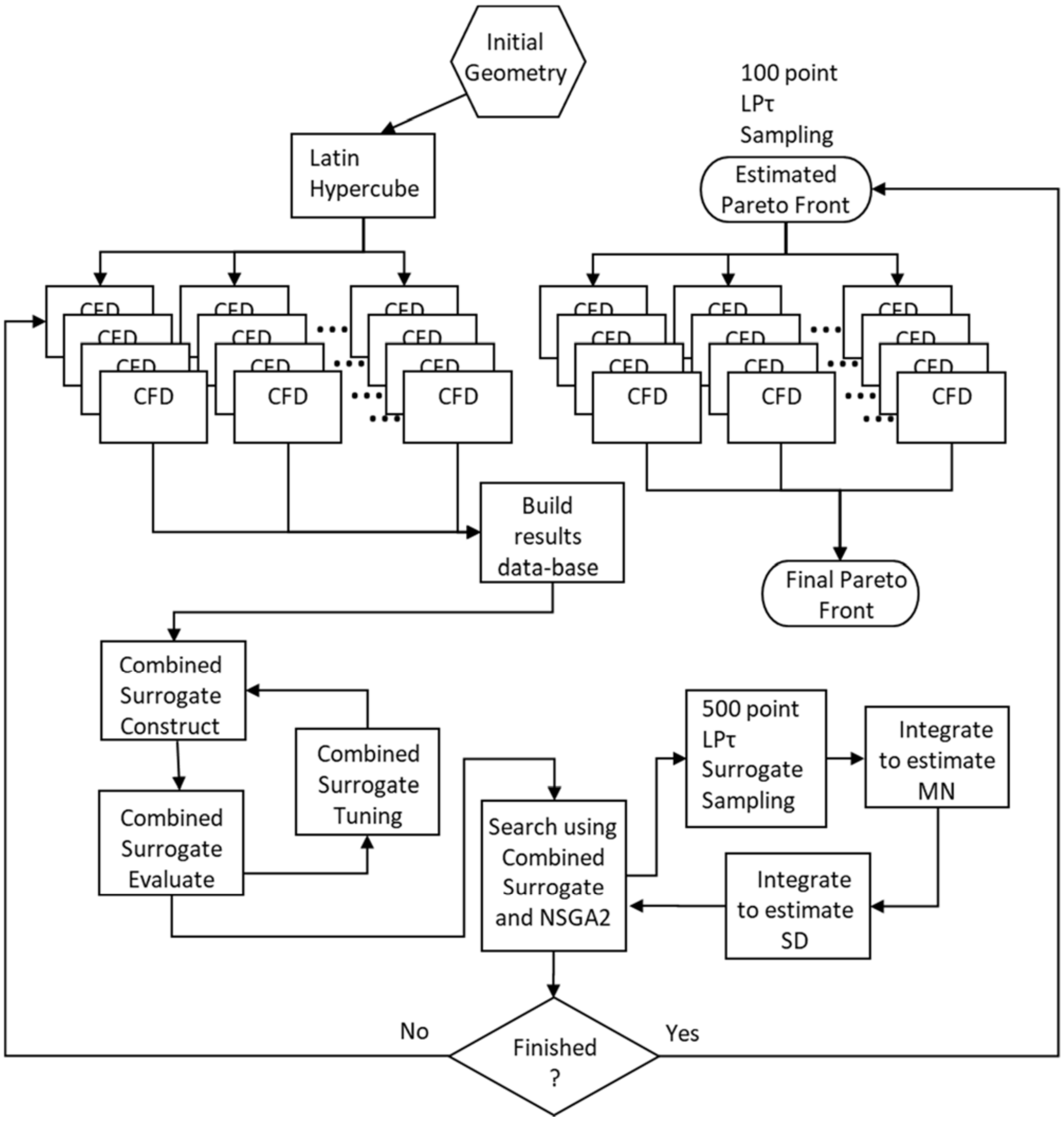


Figure 9 – Flow chart illustrating the update sequences used in combined surrogate approaches.

Figure 10 shows the resulting estimated fronts along with "true" fronts produced by carrying out 100 point LPτ UQ sampling at *each* point along both fronts. The "true" points are in each case joined to the equivalent NSGAII produced estimate from the surrogate by a dotted line – the lengths of these lines indicate the prediction errors for the individual design points. As can be seen, neither estimated front is a completely accurate predictor, however, there are regions of both fronts where predictions are reasonably good although there are significantly more of these for those produced using the neural network approach. It is also revealing that both estimated fronts suggest design performance that is unrealizable in practice, in the case of the neural network this is for low variance designs, while for the GPR it is for low mean and low variance designs.

The results from the neural network supported search clearly also appear to fall into distinct groups, with one group showing significantly better correlation between predicted and actual performance. This distinction can be further revealed by plotting out all the neural network derived designs using parallel coordinates along with the actual and predicted performance quantities, see Figure 11. The lines in this figure are coloured according to the length of the vector joining the predicted and actual locations in the Pareto front plot (i.e., the lengths of the dotted lines in Figure 10). Those designs where the errors are less than 0.05 are clearly distinct from the rest: for example design variable 9 is set to its maximum value for all these designs while for the rest it is at 60% or less; there is a similar clear distinction in design variables 8 and 10. These three variables control the rear half of the section pressure side and have all added material in that region. Conversely, variable 1 is set to its minimum value removing material just behind the leading edge on the suction side.

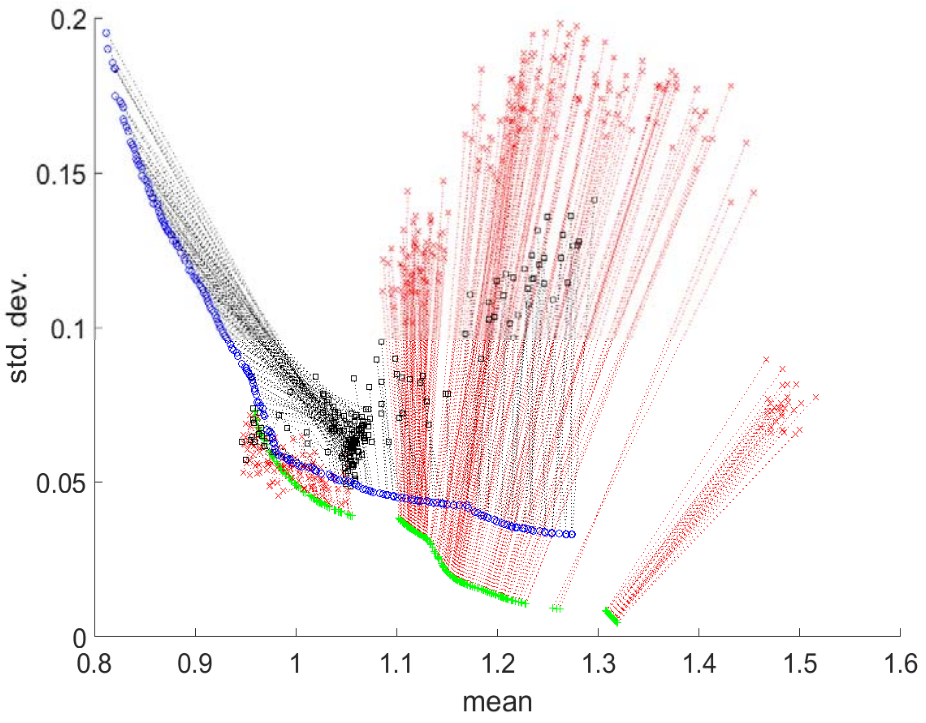


Figure 10 – Results of NSGAII multi-objective searches over surrogate models to establish Pareto fronts that trade mean performance versus standard deviation in that performance. Plus symbols are from the neural network surrogate while circles are for the Krig GPR surrogate. Crosses and squares are for designs validated by 100 point LPτ UQ sampling.

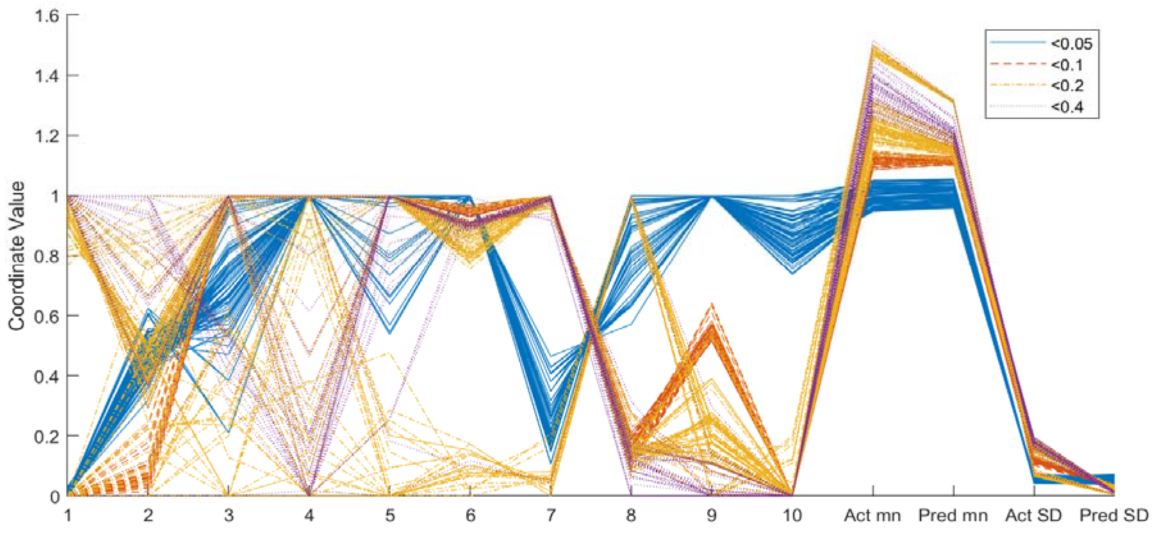


Figure 11 – Parallel co-ordinate plot of neural network derived Pareto front designs, patterned and coloured by length of error vectors (solid line < 0.05, dashed line < 0.1, chained line < 0.2, dotted line < 0.4).

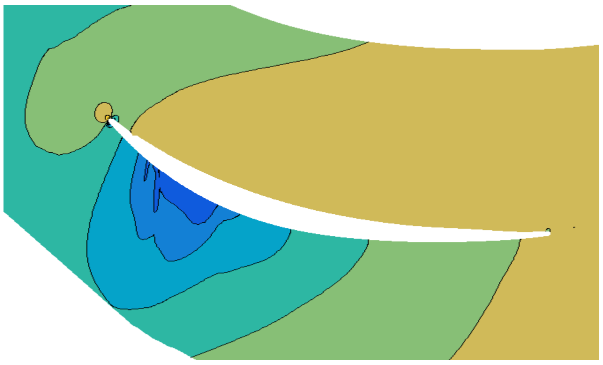


Figure 12 – Section profile and pressure field of design taken from knee of neural network designed Pareto front.

It is not clear why this grouping has occurred but it does suggest that there are certain settings to the design variable vector that will lead to designs with both good nominal performance and resilience to damage and which, moreover, are more readily modelled by the neural network surrogate. An example of one such geometry and its associated pressure field is presented in Figure 12. Note that as this design has been optimized at a single operating condition it has a much sharper leading edge, as would be expected, and as was noted before in [3]. More practical sections would be designed using multi-point approaches and would have less sharp leading edges.

It is also worth emphasizing that not only does the neural network approach produce significantly more designs with high quality "true" results; perhaps more importantly the entire neural net run took 4.8 hours while the Krig run took well over two weeks on the same hardware! In both cases the final evaluations using 100 point LPτ required tens of thousands of extra CFD runs for validation. When carrying out such design searches it is important to allow for this additional computational overhead when deciding how to allocate any available resources. It would also be possible to add these fresh validation runs into the data-bases used to produce the surrogates and then refine them in an update sequence and repeat the whole process – it is clear from Figure 10 that both surrogates could be improved, although Figure 8 suggests there are probably absolute limits on the accuracy that could be obtained with these methods on a problem of this kind.

7. A Note on Reproducibility

Note that in all the searches reported here some form of random number or pseudo random number sequence has been used. These are required because the global optimizers used in surrogate training and the searches used in NSGAII rely on random operations to work as does the creation of large Latin hypercubes. It is therefore the case that all of the results reported vary with different sequences of random numbers and so to draw completely definite conclusions some more thorough investigation of averaging would ideally be needed, even though hundreds of thousands of individual design calculations have been carried out – unfortunately when working with resource intensive RANS codes, as here, this is not possible even given the extensive computing power available in the cluster being used. It does however demonstrate the clear benefit of including variance measures in design optimization and various forms of surrogate modelling as a method for compensating for limited number of Monte-Carlo evaluations.

8. Conclusions

In this paper we have shown how advanced surrogate models can be used to support robust design optimization problems, both speeding up the search for good designs and supporting the process of design uncertainty quantification. Such approaches can be an order of magnitude or more faster than simple Monte-Carlo methods. We have shown that when large quantities of data are available then reasonably deep neural networks, with up to eight hidden layers, outperform many of the other models currently readily available in the literature. The point at which such approaches outperform more standard support vector and Gaussian process regression schemes would seem to be when there are more than a few tens of thousands of data points to construct surrogates from and that this is largely driven by the computational costs of training the surrogate models.

9. Acknowledgements

The use of Rolls-Royce plc codes and the financial support of the ATI under the GemINIDS grant are gratefully acknowledged.

10. References

[1] M. Fernández-Delgado, M. S. Sirsat, E. Cernadas, S. Alawadi, S. Barro and M. Febrero-Bande, “An extensive experimental survey of regression methods,” *Neural Networks, DOI: 10.1016/j.neunet.2018.12.010,* vol. 111, pp. 11-34, 2019.

[2] A. Keane, “Comparison of Several Optimisation Strategies for Robust Turbine Blade Design,” *J. Propulsion and Power, DOI: 10.2514/1.38673,* vol. 25, no. 5, pp. 1092-1099, 2009.

[3] A. Keane, “Use of Co-Kriging for Robust Design Optimization,” *AIAA Journal, DOI: 10.2514/1.J051391,* vol. 50, no. 11, pp. 2351-2364, 2012.

[4] D. E. Goldberg, Genetic Algorithms in Search, Optimization and Machine Learning, Addison-Wesley Longman Publishing Co., ISBN:0201157675, 1989.

[5] D. Jones, M. Schonlau and W. Welch, “Efficient global optimization of expensive black-box functions,” *Journal of Global Optimization, DOI: 10.1023/A:1008306431147,* vol. 13, pp. 455-492, 1998.

[6] S. Tsutsui and A. Ghosh, “Genetic Algorithms with a Robust Solution Searching Scheme,” *IEEE Transactions on Evolutionary Computation, DOI: 10.1109/4235.661550,* vol. 1, no. 3, p. 201–208, 1997.

[7] K. Deb, S. Agrawal, A. Pratap and T. Meyarivan, “A fast elitist non-dominated sorting genetic algorithm for multi-objective optimization: NSGA-ii,” *Lecture Notes in Computer Science, DOI: 10.1007/3-540-45356-3_83,* vol. 1917, p. 848–849, 2000.

[8] A. J. Keane, “Statistical Improvement Criteria for Use in Multiobjective Design Optimization,” *AIAA Journal, DOI: 10.2514/1.16875,* vol. 44, no. 4, pp. 879-891, 2006.

[9] S. Shahpar and L. Lapworth, “Parametric design and rapid meshing systems for turbomachinery optimization,” in *ASME GT-2003, DOI: 10.1115/GT2003-38698*, Atlanta, 2003.

[10] A. Kumar, A. Keane, P. Nair and S. Shahpar, “Robust design method of compressor fan blades against erosion,” *ASME Journal of Mechanical Design, DOI: 10.1115/DETC2006-99304,* vol. 128, p. 864–873, 2006.

[11] R. Hicks and P. Henne, “Wing design by numerical optimization,” *Journal of Aircraft, DOI: 10.2514/6.1977-1247,* vol. 15, p. 407–412, 1978.

[12] A. Kumar, Robust design methodolgies: application to compressor blades, PhD thesis, Southampton: University of Southampton, 2008.

[13] P. Moinier and M. Giles, “Preconditioned Euler and Navier-Stokes calculations on unstructured grids,” in *[48] Moinier, P. and Giles, M.B., “Preconditioned Euler and Navier-Stokes calculations on unstructured grids”, Proceedings of the 6th ICFD Conference on Numerical Methods for Fluid Dynamics*, Oxford, 1999.

[14] R. Statnikov and J. Matusov, Multicriteria Optimization and Engineering, New York: Chapman and Hall, ISBN 978-1-4615-2089-4, 1995.

[15] D. Marquardt, “An Algorithm for Least-Squares Estimation of Nonlinear Parameters,” *J. Applied Mathematics, DOI: 10.1137/0111030,* vol. 11, no. 2, p. 431–441, 1963.

[16] I. I. Voutchkov and A. J. Keane, “Multi-objective Optimization Using Surrogates in Computational Intelligence in Optimization,” in *Computational Intelligence in Optimization*, Y. T. a. C. Goh, Ed., Berlin, Springer-Verlag, ISBN 978-3-642-12774-8, DOI: 10.1007/978-3-642-12775-5_7, 2010, pp. 155-175.

[17] R. B. Gramacy and H. K. H. Lee, “Bayesian Treed Gaussian Process Models with an Application to Computer Modeling,” *J. American Statistical Association, DOI: 10.1198/016214508000000689,* vol. 103, no. 483, pp. 1119-1130, 2008.

[18] G. Dellino, J. Kleijnen and C. Meloni, “Robust optimization in simulation: Taguchi and Krige combined,” *Informs J. Comput., DOI: 10.1287/ijoc.1110.0465,* vol. 24, no. 3, p. 471–484, 2012.