UCL CDT DIS Note

25th April 2020

Draft version 0

Surrogate Modelling of the Tritium Breeding Ratio

Petr Mánek[a] and Graham Van Goffrier[a]

[a]University College London

The tritium breeding ratio (TBR) is an essential quantity for the design of modern and next-generation Tokamak nuclear fusion reactors. Representing the ratio between tritium fuel generated in breeding blankets at the boundary of the reactor, and fuel consumed during reactor runtime, the TBR depends in a complex manner on reactor geometry and material properties. We explored the training of surrogate models to produce a cheap but high-quality approximations for a Monte Carlo TBR model in use at the UK Atomic Energy Authority. We investigated possibilities for dimensional reduction on the parameter space of this model, and reviewed N classes of surrogate models for potential applicability. Here we present the performance and scaling properties of these surrogate models, the best of which demonstrated an accuracy of X and a mean prediction time of X, representing a relative speedup X with respect to the MC model. We further present a novel adaptive sampling algorithm, QASS, capable of interfacing with any of the individual studied models. Our preliminary testing on a toy TBR theory has demonstrated the efficacy of this algorithm for speeding up the surrogate modelling process.

Contents

1 Introduction

The analysis of massive datasets has become a necessary component of virtually all technical fields, as well as the social and humanistic sciences, in recent years. Given that rapid improvements in sensing and processing hardware have gone hand in hand with the data explosion, it is unsurprising that software for the generation and interpretation of this data has also attained a new frontier in complexity. In particular, simulation procedures such as Monte Carlo (MC) event generation can perform physics predictions even for theoretical regimes which are not analytically soluble. The bottleneck for such procedures, as is often the case, lies in the computational time and power which they necessitate.

Surrogate models, or metamodels, can resolve this limitation by replacing a resource-expensive procedure with a much cheaper approximation [1]. They are especially useful in applications where numerous evaluations of an expensive procedure are required over the same or similar domains, e.g. in the parameter optimisation of a theoretical model. The term "metamodel" proves especially meaningful in this case, when the surrogate model approximates a computational process which is itself a model for a (perhaps unknown) physical process [2]. There exists a spectrum between "physical" surrogates which are constructed with some contextual knowledge in hand, and "empirical" surrogates which are derived purely from the underlying expensive model.

In this internship project, in coordination with the UK Atomic Energy Authority (UKAEA) and Culham Centre for Fusion Energy (CCFE), we sought to develop a surrogate model for the tritium breeding ratio (TBR) in a Tokamak nuclear fusion reactor. Our expensive model was a MC-based neutronics simulation [3], itself a spherical approximation of the Joint European Torus (JET) at CCFE, which returns a prediction of the TBR for a given reactor configuration. We took an empirical approach to the construction of this surrogate, and no results described here are explicitly dependent on prior physics knowledge.

For the remainder of Section 1, we will define the TBR and set the context of this work within the goals of the UKAEA and CCFE. In Section 2 we will describe our datasets generated from the expensive model for training and validation purposes, and the dimensionality reduction methods employed to develop our understanding of the parameter domain. In Section 3 we will present our methodologies for the comparison testing of a wide variety of surrogate modelling techniques, as well as a novel adaptive sampling procedure suited to this application. After delivering the results of these approaches in Section 4, we will give our final conclusions and recommendations for further work.

1.1 Problem Description

Nuclear fusion technology relies on the production and containment of an extremely hot and dense plasma. In this environment, by design similar to that of a star, hydrogen atoms attain energies sufficient to overcome their usual electrostatic repulsion and fuse to form helium [4]. Early prototype reactors made use of the deuterium (^{2}H, or D) isotope of hydrogen in order to achieve fusion under more accessible conditions, but lead to limited success. The current frontier generation of fusion reactors, such as JET and the under-construction International Thermonuclear Experimental Reactor (ITER), make use of tritium (^{3}H, or T) fuel for further efficiency gain. Experimentation at JET dating back to 1997 [5] has made significant headway in validating deuterium-tritium (D-T) operations and constraining the technology which will be employed in ITER in a scaled up form.

However, tritium is much less readily available as a fuel source than deuterium. While at least one deuterium atom occurs for every 5000 molecules of naturally-sourced water, and may be easily distilled, tritium is extremely rare in nature. It may be produced indirectly through irradiation of heavy water (D_2O) during nuclear fission, but only at very low rates which could never sustain industrial-scale fusion power.

Instead, modern D-T reactors rely on tritium breeding blankets, specialised layers of material which partially line the reactor and produce tritium upon neutron bombardment, e.g. by

$$ {}^{1}_{0}\mathrm{n} + {}^{6}_{3}\mathrm{Li} \longrightarrow {}^{3}_{1}\mathrm{T} + {}^{4}_{2}\mathrm{He} \tag{1} $$

$$ {}^{1}_{0}\mathrm{n} + {}^{7}_{3}\mathrm{Li} \longrightarrow {}^{3}_{1}\mathrm{T} + {}^{4}_{2}\mathrm{He} + {}^{1}_{0}\mathrm{n} \tag{2} $$

where T represents tritium and ^{7}Li, ^{6}Li are the more and less frequently occurring isotopes of lithium, respectively. ^{6}Li has the greatest tritium breeding cross-section of all tested isotopes [4], but due to magnetohydrodynamic instability of liquid lithium in the reactor environment, a variety of solid lithium compounds are preferred.

The TBR is defined as the ratio between tritium generation in the breeding blanket per unit time and tritium fuel consumption in the reactor. The MC neutronics simulations previously mentioned therefore must account for both the internal plasma dynamics of the fusion reactor and the resultant interactions of neutrons with breeding blanket materials. Neutron paths are traced through a CAD model (e.g. Figure 1) of a reactor with modifiable geometry.

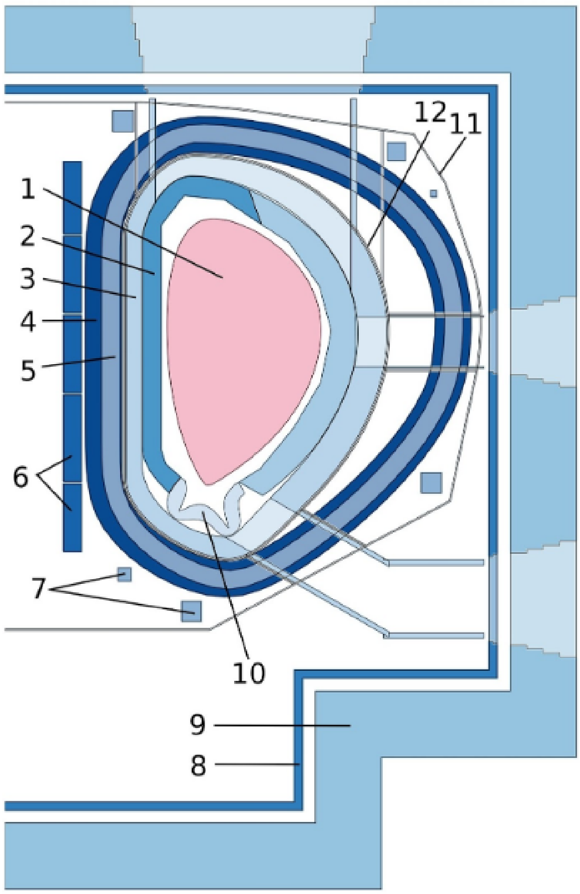


Figure 1: Typical single-null reactor configuration as specified by BLUEPRINT [6]: 1 — plasma, 2 — breeding blankets

The input parameters of the computationally-expensive TBR model therefore fall into two classes. Continuous parameters, including material thicknesses and packing ratios, describe the geometry of a given reactor configuration. Discrete categorical parameters further specify all relevant material sections, including coolants, armours, and neutron multipliers. One notable exception is the enrichment ratio, a continuous parameter denoting the presence of ^{6}Li. Our challenge, put simply, was to produce a TBR function which takes these same input parameters and approximates the MC TBR model with the greatest achievable accuracy.

2 Data Exploration

The initial step of our work is the study of the existing MC TBR model and its behaviour. Following the examination of its features (simulation parameters), we present efficient means of evaluating this model on large sets of points in high-performance computing (HPC) environment, preprocessing techniques designed to adapt collected datasets for surrogate modelling, and our attempts at feature space reduction to achieve the lowest possible number of dimensions.

2.1 Expensive Model Description

The MC TBR model that we understand to be the expensive function for our surrogate modelling is fundamentally a Monte Carlo simulation based on the OpenMC framework [7]. At input the software expects a set of 19 parameters, discrete and continuous, that are fully listed in Table 1. Following a brief period of time (usually on the order of tens of seconds), during which a fixed number of events is generated, the simulation outputs the mean and the standard deviation of the TBR aggregated over the simulated run. The former of these two we accept to be the output TBR value that is subject to approximation.

	Parameter Name	Acronym	Type	Domain	Description
Blanket	Breeder fraction†	BBF	Continuous	$[0, 1]$	TODO
	Breeder ^{6}Li enrichment fraction	BBLEF	Continuous	$[0, 1]$	
	Breeder material	BBM	Discrete	$\{Li_2TiO_3, Li_4SiO_4\}$	
	Breeder packing fraction	BBPF	Continuous	$[0, 1]$	
	Coolant fraction†	BCF	Continuous	$[0, 1]$	
	Coolant material	BCM	Discrete	$\{D_2O, H_2O, He\}$	
	Multiplier fraction†	BMF	Continuous	$[0, 1]$	
	Multiplier material	BMM	Discrete	$\{Be, Be_{12}Ti\}$	
	Multiplier packing fraction	BMPF	Continuous	$[0, 1]$	
	Structural fraction†	BSF	Continuous	$[0, 1]$	
	Structural material	BSM	Discrete	$\{SiC, eurofer\}$	
	Thickness	BT	Continuous	$[0, 500]$	
First wall	Armour fraction‡	FAF	Continuous	$[0, 1]$	
	Armour material	FAM	Discrete	$\{tungsten\}$	
	Coolant fraction‡	FCF	Continuous	$[0, 1]$	
	Coolant material	FCM	Discrete	$\{D_2O, H_2O, He\}$	
	Structural fraction‡	FSF	Continuous	$[0, 1]$	
	Structural material	FSM	Discrete	$\{SiC, eurofer\}$	
	Thickness	FT	Continuous	$[0, 20]$	

Table 1: Input parameters supplied to the MC TBR simulation in alphabetical order. Fractions marked with superscripts†‡ are independently required to sum to one.

In our work, we often reference TBR points or samples. These are simply vectors in the feature space generated by Cartesian product of domains of all features—parameters from Table 1.

Since most surrogate models that we employ assume overall continuous inputs, we take steps to unify our feature interface in order to attain this property. In particular, we eliminate discrete features by embedding each such feature into a finite multitude of continuous features using standard one-hot encoding. This option is available to us since discrete domains that generate our feature space are finite in cardinality and relatively small in size. And while it renders all features continuous, the modification comes at the expense of increasing the dimensionality of the feature space to 27 features in total. This is further discussed in Section 2.3.

2.2 Dataset Generation

While we deliberately make no assumptions about the internal properties of the MC TBR simulation, effectively treating it as a black box model, our means of studying its behaviour are limited to inspection of its outputs at various points in the feature space, and inference thereof. We thus require sufficiently large and representative quantities of samples to ensure statistical significance of our findings.

With grid search in high-dimensional domain clearly intractable, we selected uniform pseudo-random sampling to generate large amounts of feature configurations that we consider to be independent and unbiased. For evaluation of the expensive MC TBR model, we utilise parallelisation offered by the HPC infrastructure available at UCL computing facilities. To this end, we designed and implemented the Approximate TBR Evaluator—a Python software package capable of sequential evaluation of the multi-threaded OpenMC simulation on batches of previously generated points in the feature space. We deployed ATE at the UCL Hypatia cluster RCIF partition, which consists of 4 homogeneous nodes, each containing 40 cores. We completed three data generation runs, which are summarised in Table 2.

#	Samples	Batch division	t_{run}	$\bar{t}_{\text{eval.}}$ [s]	Description
0	100 000	100×1000	2 days, 23 h	7.88 ± 2.75	Testing run using older MC TBR version.
1	500 000	500×1000	13 days, 20 h	7.78 ± 2.81	Fully uniform sampling in the entire domain.
2	400 000	400×1000	10 days	7.94 ± 2.60	Mixed sampling, discrete features fixed.

Table 2: Parameters of sampling runs. Here, t_{run} denotes the total run time (including waiting in the processing queue), and $\bar{t}_{\text{eval.}}$ is the mean evaluation time of MC TBR (per single sampled point).

Skipping run zero, which was performed using older, fundamentally different version of the MC TBR software, and was thus treated as a technical proof-of-concept, we generated the total of 900 000 samples in two runs. While the first run featured fully uniform sampling of the unrestricted feature space, the second run used more elaborate strategy. Interested in further study of relationships between discrete and continuous features, we selected four assignments of discrete features (listed in Table 3) and fixed them for all points, effectively slicing the feature space into four corresponding subspaces. In order to achieve comparability between these *slices*, the remaining unassigned features were uniformly sampled only in the first slice, and reused in the other three.

Since some surrogate modelling methods applied in this work are not scale-invariant or perform suboptimally with arbitrarily scaled problems, all run results were standardised prior to further use. In this commonly used statistical procedure, features and regression outputs are independently scaled and offset to attain zero mean and unit variance.

	Discrete feature assignment						
Batches	BBM	BCM	BMM	BSM	FAM	FCM	FSM
0-99	Li_4SiO_4	H_2O	$Be_{12}Ti$	eurofer	tungsten	H_2O	eurofer
100-199	Li_4SiO_4	He	$Be_{12}Ti$	eurofer	tungsten	H_2O	eurofer
200-299	Li_4SiO_4	H_2O	$Be_{12}Ti$	eurofer	tungsten	He	eurofer
300-399	Li_4SiO_4	He	$Be_{12}Ti$	eurofer	tungsten	He	eurofer

Table 3: Selected discrete feature assignments corresponding to slices in run 2.

2.3 Dimensionality Reduction

Model training over high-dimensional parameter spaces may be improved in many aspects by carefully reducing the number of variables used to describe the space. For many applications, feature selection strategies succeed in identifying a sufficiently representative subset of the original input variables; however, all given variables were assumed to be physically relevant to the MC TBR model. Feature extraction methods, on the other hand, aim to identify a transformation from the parameter space which decreases dimensionality; even if no individual parameter is separable from the space, some linear combinations of parameters or nonlinear functions of parameters may be.

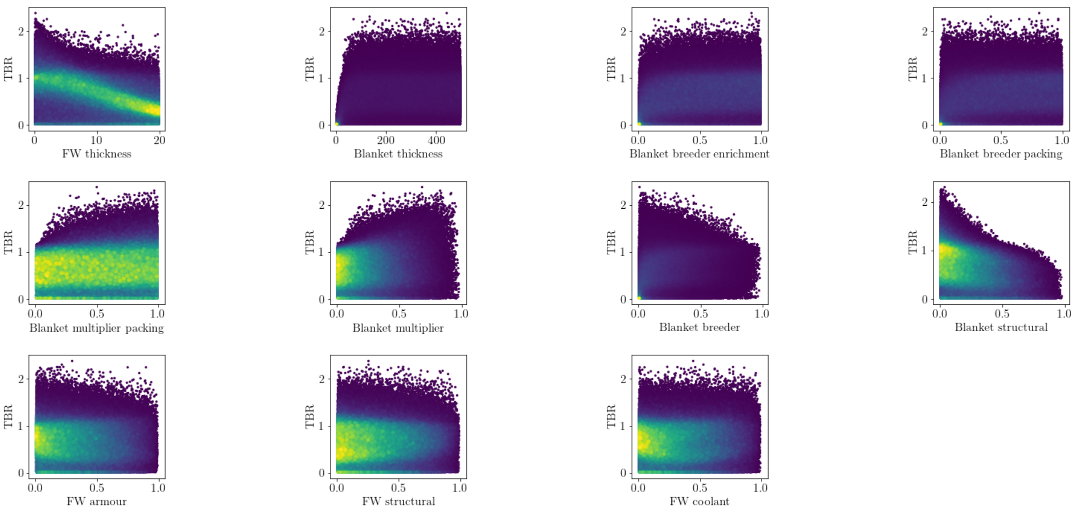


Figure 2: Marginalised dependence of TBR on the choice of continuous features.

2.3.1 Principal Component Analysis

To pursue linear feature extraction, principal component analysis (PCA) [8] was performed via SciKit Learn [9] on a set of 300 000 uniform samples of the MC TBR model.

Figure 3 shows the resultant cumulative variance of PCA-identified feature vectors. The similar share of variance among all features reveals irreducibility of the TBR model by linear methods.

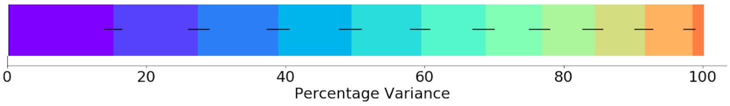


Figure 3: Cumulative variance for optimal features identified by PCA

2.3.2 Variogram Computations

Kriging is a geostatistical surrogate modelling technique which relies on correlation functions over distance (lag) in the feature space [10]. Although kriging performed poorly for our use case due to high dimensionality, these correlation measures gave insight into similarities between discrete-parameter slices of the data.

Figure 4 shows the Matheron semivariance [11] for three discrete slices with coolant material varied, but all other discrete parameters fixed. Fits [12] to the Matérn covariance model confirmed numerically that the coolant material is the discrete parameter with the greatest distinguishability in the MC TBR model.

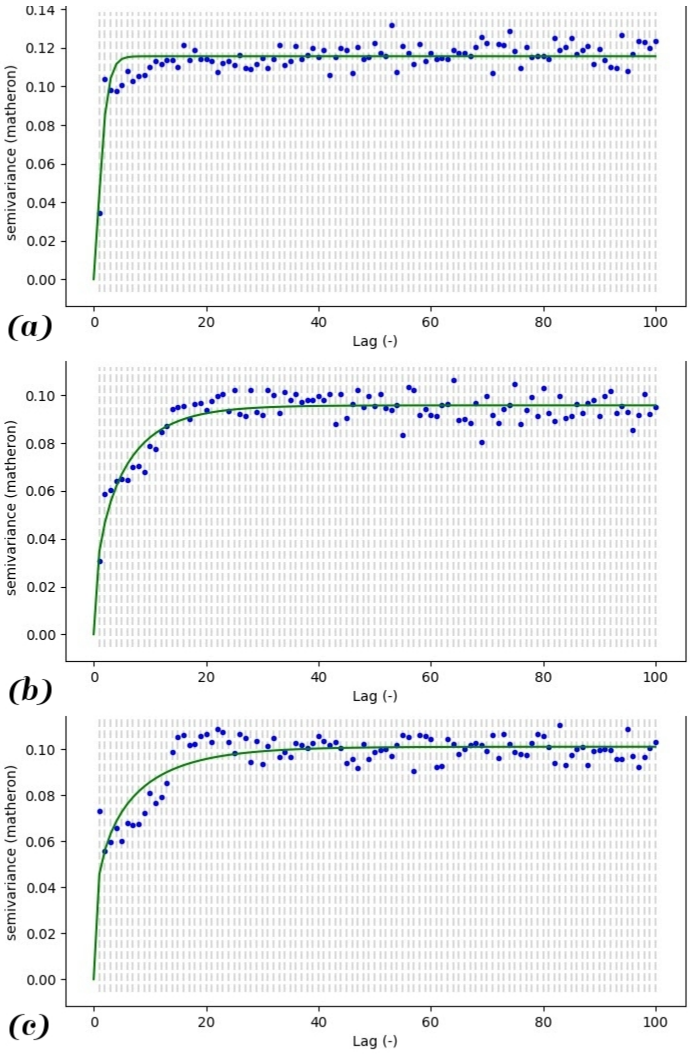


Figure 4: Semivariograms for MC TBR data with coolant materials: (a) He, (b) H_2O, (c) D_2O

2.3.3 Autoencoders

Autoencoders [13] are a family of approaches to dimensionality reduction driven by artificial neural networks (ANNs). Faced with many possible alternatives, we utilised a conventional autoencoder, which relies on a three-layer network that is trained to replicate the identity mapping. While it follows that the input and output layers of such network are sized to accommodate the analysed dataset, the hidden layer, also called *the bottleneck*, allows for variable number of neurons that represent a smaller subspace. By scanning over a range of bottleneck widths and investigating relative changes in the validation loss, we assess the potential for dimensional reduction.

In particular, we consider two equally-sized sets[1] of samples: (i) a subset of data obtained from run 1 and (ii) a subset of a single slice obtained from run 2. Our expectation was that while the former case

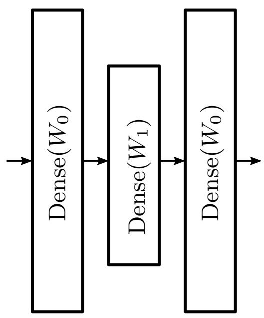


Figure 5: Autoencoder with input width W_0 and bottleneck width W_1.

[1] Each set contained 100 000 samples from batches 0-99 of the corresponding runs.

would provide meaningful insights into correlations within the feature space, the latter would validate our autoencoder implementation by analysing a set of points that are trivially reducible in dimensionality due to a number of fixed discrete features.

The results of both experiments are shown in Figure 6. Consistent with our motivation, in each plot we can clearly identify a constant plateau of low error in the region of large dimensionality followed by a point, from which a steep increase is observed. We consider this *critical point* to mark the largest viable dimensional reduction without significant information loss. With this approach we find that the autoencoder was able to reduce the datasets into a subspace of 18 dimensions in the first case and 10 dimensions in the second case.

Confirming our expectation that in the latter, trivial case the autoencoder should achieve greater dimensional reduction, we are inclined to believe that our implementation is indeed operating as intended. However, we must also conclude that in both examined cases this method failed to produce a reduction that would be superior to a naïve approach.[2] This is consistent with previous results obtained by PCA and kriging.

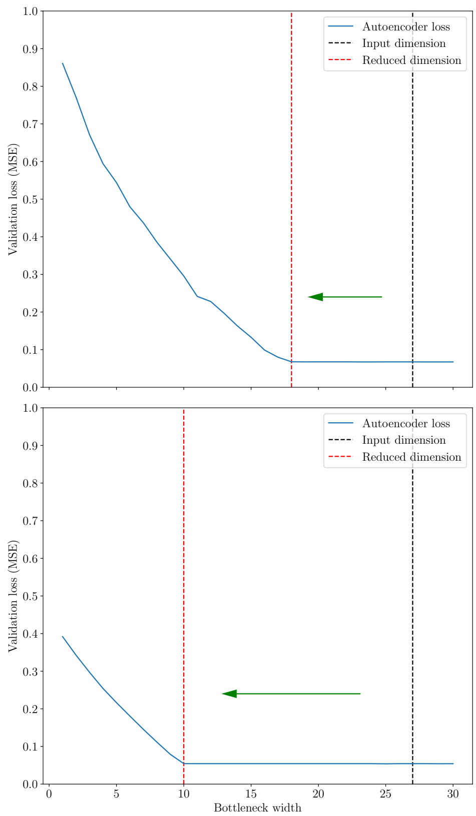


Figure 6: Autoencoder loss scan on the full feature space (top) and a single slice (bottom). Dimensional reduction is indicated by a green arrow.

3 Methodology

Assuming that data is appropriately treated to eliminate redundant features, we proceed to propose surrogate models and criteria used for their evaluation. The task all presented surrogates strive to solve can be formulated using the language of conventional regression problems. In this section, we focus on interpretation in the scheme of supervised and unsupervised learning.

Labeling the expensive MC TBR model $f(x)$, a surrogate is a mapping $\hat{f}(x)$ that yields similar images as $f(x)$. In other words, $f(x)$ and $\hat{f}(x)$ minimise a selected similarity metric. Furthermore, in order to be considered *viable*, surrogates are required to achieve expected evaluation time lower than that of $f(x)$.

In the supervised learning setting, we first gather a sufficiently large training set of samples $\mathcal{T} = \left\{\left(x^{(i)}, f\left(x^{(i)}\right)\right)\right\}_{i=1}^{N}$ to describe the behaviour of $f(x)$ across its domain. Depending on specific model class and appropriate choice of its hyperparameters, surrogate models $\hat{f}(x)$ are trained to minimise

[2] In both tested cases we can trivially eliminate 7 dimensions due to overdetermined one-hot-encoded features and 2 dimensions due to sum-to-one constraints. Furthermore, in the single slice case we may omit 7 additional dimensions due to artificially fixed features.

empirical risk with respect to $\mathcal{T}$ and a model-specific loss function $\mathcal{L}$, where empirical risk is defined as $R_{\text{emp.}}(\hat{f} \mid \mathcal{T}, \mathcal{L}) = \frac{1}{N} \sum_{i=1}^{N} \mathcal{L}\left(\hat{f}(x^{(i)}), f(x^{(i)})\right)$.

The unsupervised setting can be viewed as an extension of this method. Rather than fixing the training set $\mathcal{T}$ for the entire duration of training, multiple sets $\{\mathcal{T}_k\}_{k=0}^{K}$ are used, such that $\mathcal{T}_{k-1} \subset \mathcal{T}_k$ for all $k > 1$. The first set $\mathcal{T}_0$ is initialised randomly to provide a *burn-in*, and is repeatedly extended in epochs, whereby each epoch trains a new surrogate on $\mathcal{T}_k$ using the supervised learning procedure, evaluates its performance, and forms a new set $\mathcal{T}_{k+1}$ by adding more samples to $\mathcal{T}_k$. This permits the learning algorithm to condition the selection of new samples on the evaluation results in order to maximise improvement of surrogate performance over complex regions within the domain.

3.1 Metrics

Aiming to provide objective comparison of a diverse set of surrogate model classes, we define a multitude of metrics to be tracked during experiments. Following the motivation of this work, two desirable properties of surrogates arise: (i) their capability to approximate the TBR MC model well and (ii) their complexity. An ideal surrogate would maximise the former while minimising the latter.

Table 4 provides exhaustive listing and description of metrics recorded in the experiments. For regression performance analysis, we include a selection of absolute metrics to assess the approximation capability of surrogates, and set practical bounds on the expected accuracy of their predictions. In addition, we also track relative measures that are better-suited for model comparison between works as they maintain invariance with respect to the selected domain and image space. For complexity analysis, surrogates are assessed in terms of wall time elapsed during training and predicition. This is motivated by common practical use cases of our work, where models are trained and used as drop-in replacements for the expensive MC TBR model. Since training set sizes remain to be determined, all times are reported per a single sample. Even though some surrogates support acceleration by means of parallelisation, measures were taken to ensure sequential processing of samples to achieve comparability between considered models.[3]

To prevent undesirable bias in results due to training set selection, all metrics are collected in the scheme of k-fold cross-validation with a standard choice of $k = 5$. Herein, a sample set is subdivided into 5 disjoint folds which are repeatedly interpreted as training and testing sets, maintaining a constant ratio of samples between the two.[4] In each such interpretation experiments are repeated, and the overall value of each metric is reported as the mean across all folds.

3.2 Evaluation of Supervised Learning Surrogates

In our experiments, we evaluate and compare surrogates in effort to optimise against metrics described in Section 3.1. To attain meaningful and practically usable results, we require a suffi-

[3] The only exception to this are artificial neural networks, which are notoriously slow to train on conventional CPU architectures in serialised setting.

[4] Unless explicitly stated otherwise, we use 1 fold for testing and the $k-1$ remaining folds for training. This gives 80% to 20% train-test ratio.

Regression performance metrics	Mathematical formulation / description	Ideal value [units]	
Mean absolute error (MAE)	$\sum_{i=1}^{N} \lvert y^{(i)} - \hat{y}^{(i)} \rvert / N$	0	[TBR]
Standard error of regression S	$\mathrm{StdDev}_{i=1}^{N} \left\{ \lvert y^{(i)} - \hat{y}^{(i)} \rvert \right\}$	0	[TBR]
Coefficient of determination R^2	$1 - \sum_{i=1}^{N} \left(y^{(i)} - \hat{y}^{(i)} \right)^2 / \sum_{i=1}^{N} \left(y^{(i)} - \overline{y} \right)^2$	1	[rel.]
Adjusted R^2	$1 - (1 - R^2)(N-1)/(N-P-1)$	1	[rel.]
Complexity metrics			
Mean training time $\bar{t}_{\mathrm{trn.}}$	(wall training time of $\hat{f}(x))/N_0$	0	[ms]
Mean prediction time $\bar{t}_{\mathrm{pred.}}$	(wall prediction time of $\hat{f}(x))/N$	0	[ms]
Relative speedup σ	(wall evaluation time of $f(x))/(N\bar{t}_{\mathrm{pred.}})$	$\to \infty$	[rel.]

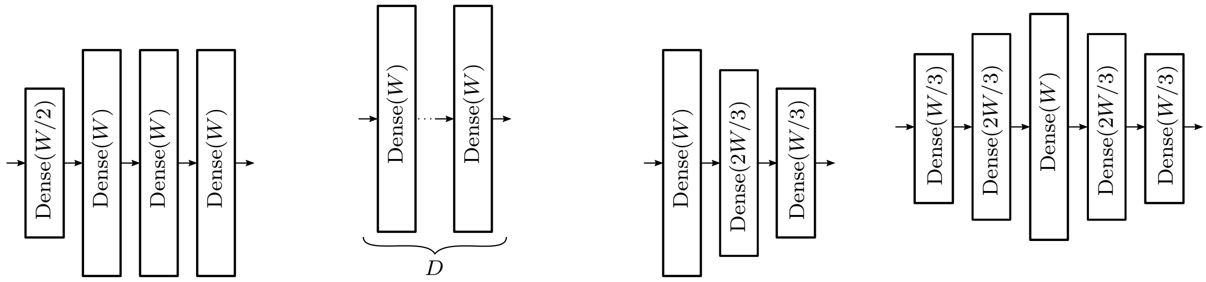
Table 4: Metrics recorded in supervised learning experiments. In formulations, we work with training set of size N_0 and testing set of size N, TBR values $y^{(i)} = f(x^{(i)})$ and $\hat{y}^{(i)} = \hat{f}(x^{(i)})$ denote images of the ith testing sample in the expensive model and the surrogate respectively. Furthermore, the mean $\overline{y} = \sum_{i=1}^{N} y^{(i)}/N$ and P is the number of input features.

ciently large and diverse pool of surrogate classes to draw from. This is described by the listing in Table 5. The presented selection of models includes basic techniques suitable for linear regression enhanced by the kernel trick or dimension lifting, methods driven by decision trees, instance-based learning models, ensemble regressors, randomized algorithms, artificial neural networks and mathematical approaches developed specifically for the purposes of surrogate modelling. For each of these classes, a state-of-the-art implementation was selected and adapted to operate with TBR samples.

Surrogate	Acronym	Implementation	Hyperparameters
Support vector machines	SVM	SciKit Learn [9]	3
Gradient boosted trees	GBT	SciKit Learn	11
Extremely randomized trees	ERT	SciKit Learn	7
AdaBoosted decision trees	AB	SciKit Learn	3
Gaussian process regression	GPR	SciKit Learn	2
k nearest neighbours	KNN	SciKit Learn	3
Artificial neural networks	ANN	Keras (TensorFlow) [14]	2
Inverse distance weighing	IDW	SMT [15]	1
Radial basis functions	RBF	SMT	3
Stochastic gradient descent	SGD	SciKit Learn	13
Ridge regression	RR	SciKit Learn	4
Kriging	KRG	SMT	4

Table 5: Considered surrogate model classes.

In some rows of Table 5, a single class in reality represents a family of fundamentally similar approaches. Good examples of this are kriging methods and neural networks. In the case of the former, multiple algorithms such as kriging based on partial least squares and gradient-enhanced kriging are considered. In the latter, a host of parametric graphs are defined to realise simplistic network architecture search (see Figure 7 for details). Discrimination between such options is considered an additional hyperparameter of the corresponding surrogate class.



(a) 1h3f(W) (b) df(D, W) (c) 3pyramid(W) (d) 5diamond(W)

Figure 7: Selected parametric neural network architectures. All layers except the last use ReLU activation. Prediction information flow is indicated by arrows. Connections between neurons are omitted for clarity.

3.2.1 Experiments

The presented surrogate candidates are evaluated in four experimental cases:

1. Hyperparameter tuning in simplified domain,
2. Hyperparameter tuning in full domain,
3. Scaling benchmark,
4. Competitive surrogate training.

The aim of the initial experiments is to use a relatively small subset of collected TBR samples to determine hyperparameters of considered surrogates. Since this process requires learning the behaviour of an unknown, possibly expensive mapping—here a function that assigns cross-validated metrics to a point in the hyperparameter domain—it in many aspects mirrors the primary task of this work with the notable extension of added utility to optimise. In order to avoid undesirable exponential slowdown in exhaustive searches of a possibly high-dimensional parameter space, Bayesian optimisation is employed as a standard hyperparameter tuning algorithm. We set its objective to maximise R^2 and perform 1000 iterations.[5]

In the first experiment, efforts are made to maximise the possibility of success in surrogates that are prone to suboptimal performance in discontinuous spaces. This follows the notion that, if desired, accuracy of such models may be replicated at large scale by decomposing the TBR domain into continuous subsets, and learning ensemble surrogate comprised of identical models that are trained on each subset independently, incurring exponential complexity penalty in the process. To this end, data are limited to a single slice from run 2, and discrete features are completely withheld from evaluated surrogates. This is repeated for each of the four available slices to investigate variance in behaviour under different discrete feature assignments.

The second experiment conventionally measures surrogate performance on the full feature space. Here, in extension of the previous case, surrogates work with samples comprised of discrete as well as continuous features.

[5] Tuning of each surrogate class was artificially terminated after 2 days. Instances which reached this limit are marked in the results.

The goal of the last two experiments is to exploit the information gathered by hyperparameter tuning for study of scaling properties. In the third experiment, 20 best-performing hyperparameter choices of each class (in R^2) are used to learn surrogates using progressively larger training sets. Following that, the fourth experiment attempts to produce competitive surrogates by re-training several selected well-scaling models on the largest available data set.

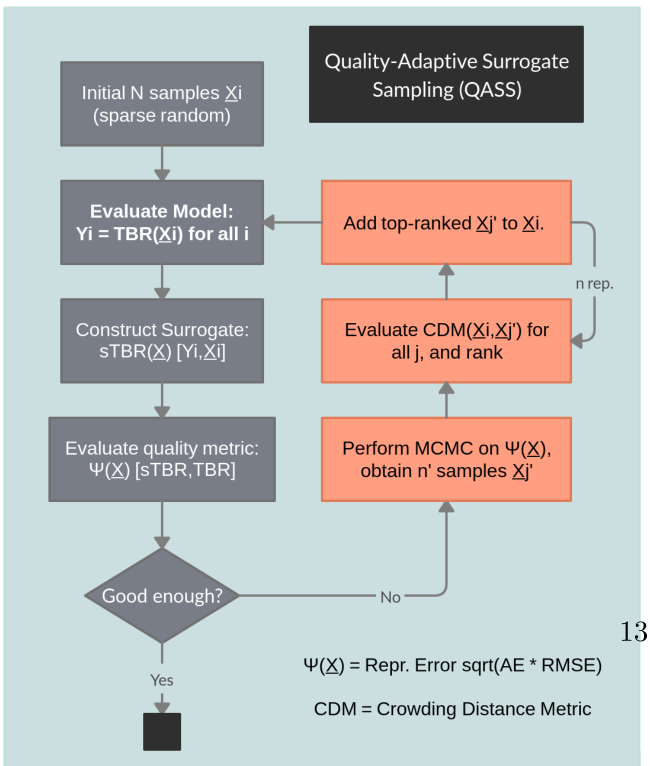
3.3 Adaptive Sampling

All of the surrogate modelling techniques studied in this project face a shared challenge: their accuracy is limited by the quantity of training samples which are available from the expensive MC TBR model. Adaptive sampling procedures can improve upon this limitation by taking advantage of statistical information which is accumulated during the training of any surrogate model. Rather than training the surrogate on a single sample set generated according to a fixed strategy, sample locations are chosen incrementally so as to best suit the model under consideration.

Adaptive sampling techniques are widespread in the literature and have been specialised for surrogate modelling. Garud's [16] "Smart Sampling Algorithm" achieved notable success by incorporating surrogate quality and crowding distance scoring to identify optimal new samples, but was only tested on a single-parameter domain. We theorised that a nondeterministic sample generation approach, built around Markov Chain Monte Carlo methods (MCMC), would fare better for high-dimensional models by more thoroughly exploring all local optima in the feature space. MCMC produces a progressive chain of sample points, each drawn according to the same symmetric proposal distribution from the prior point. These sample points will converge to a desired posterior distribution, so long as the acceptance probability for these draws has a particular functional dependence on that posterior value (see [17] for a review).

Many researchers have embedded surrogate methods into MCMC strategies for parameter optimisation [18, 19], in particular the ASMO-PODE algorithm [20] which makes use of MCMC-based adaptive sampling to attain greater surrogate precision around prospective optima. Our novel approach draws inspiration from ASMO-PODE, but instead uses MCMC to generate samples which increase surrogate precision throughout the entire parameter space.

We designed the Quality-Adaptive Surrogate Sampling algorithm (QASS, Figure 8) to iteratively increment the training/test set with sample points which maximise surrogate error and minimise a crowding distance metric (CDM) [21] in feature space. On each iteration following an initial training of the surrogate on N uniformly random samples, the surrogate was trained and AE calculated. MCMC was then performed on the error function generated by performing nearest-neighbor interpolation on these test error points. The resultant samples were



culled by 50% according to the CDM, and then the n highest-error candidates were selected for reintegration with the training/test set, beginning another training epoch. Validation was also performed during each iteration on independent, uniformly-random sample sets.

4 Results

TODO

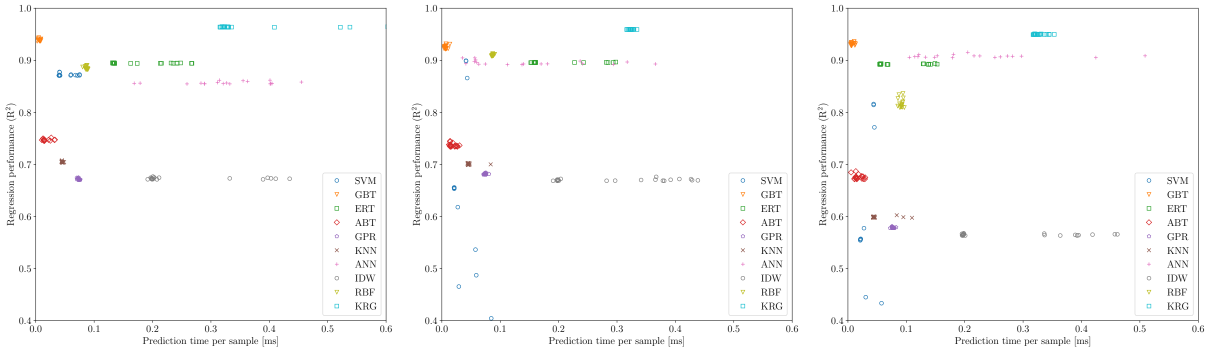
4.1 Evaluation of Supervised Learning Surrogates

We begin by evaluating a diverse set of surrogate classes that we proposed earlier. In particular, we aim to study considered models in terms of regression performance and evaluation complexity. Following Section 3.2.1, where we proposed four experiments that accomplish this task, we present and discuss our results in the next several sections.

4.1.1 Hyperparameter Tuning

The first two experiments perform Bayesian optimisation to maximise R^2 in cross-validated setting as a function of model hyperparameters. While in the first experiment we limit training and testing sets to the scope of four slices of the feature space, in the second experiment we lift this restriction.

The results displayed in Figure 9 indicate that in the first experiment, gradient boosted trees clearly appear to be the most accurate as well as the fastest surrogate class in terms of mean prediction time. Following that, we note that extremely randomised trees, support vector machines and artificial neural networks also achieved satisfactory results with respect to both examined metrics. While the remainder of tested surrogate classes does not exhibit problems in complexity, its regression performance falls below average.



(a) Run 2, batches 0-3 (b) Run 2, batches 100-103 (c) Run 2, batches 200-203

Figure 9: 20 best-performing surrogates per each considered class, plotted in terms of complexity (as $\bar{t}_{\text{pred.}}$) and regression performance (as R^2) on selected slices of run 2, evaluated in experiment 1.

TODO: multislice description

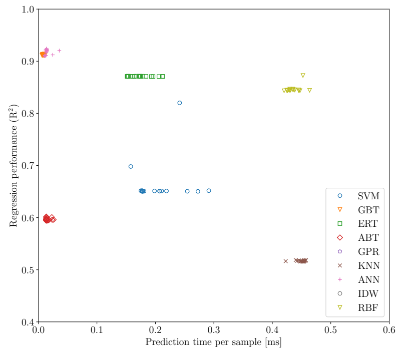
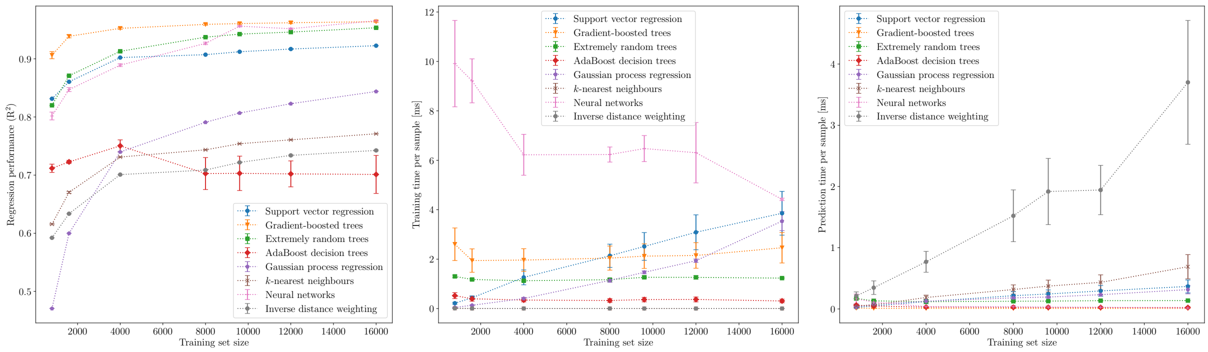


Figure 10: Results of experiment 2, plotted analogously to Figure 9.

4.1.2 Scaling Benchmark

In the next experiment we examine surrogate scaling properties correlating metrics of interest with progressively increasing training set size. The results shown in Figure 11 seem to suggest that in terms of regression performance, methods based on decision trees and artificial neural networks offer the best accuracy on larger sets overall. This is a curious extension of the previous findings, where gradient boosted trees were observed to significantly dominate over the rest of the examined methods. With increasing training set size, their relative advantage is clearly gradually diminished.



(a) Regression performance (as R^2) (b) Mean training time (c) Mean prediction time

Figure 11: Various metrics collected during experiment 3 (scaling benchmark) displayed as a function of training set size.

According to our experiment, the lowest mean training time is generally achieved by instance-based learning methods, which seem to offer near-constant scaling characteristics at the expense of significant performance increase later during prediction. Following that, we observe that the majority of tree-based methods also exhibit desirable properties. The notable exception here appear to be artificial neural networks, which are the only model to utilise parallelisation. As such, their constant synchronisation overhead presumably hinders performance on small training sets, producing misleading results when divided by the number of samples.

In terms of mean prediction time, all tested surrogates except previously mentioned instance-based learning methods scale exceptionally well. Tree-based generally models appear to perform the fastest.

4.1.3 Competitive Surrogate Training

TODO

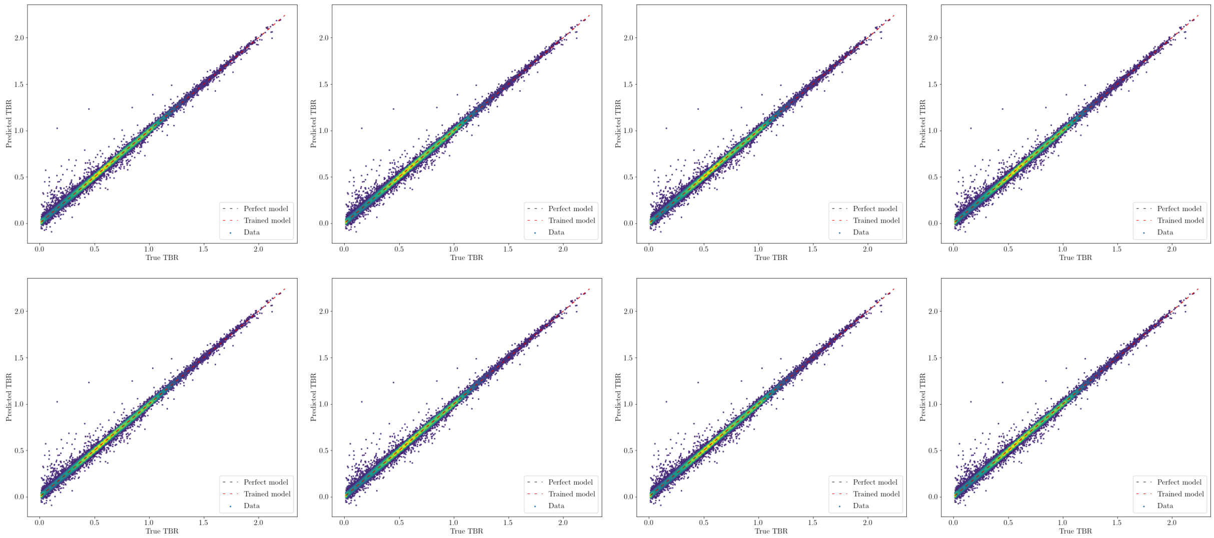


Figure 12: Regression performance of selected best-performing models trained in experiment 4.

4.2 Results of Adaptive Sampling

In order to test our QASS prototype, several functional toy theories for TBR were developed as alternatives to the expensive MC model. By far the most robust of these was the following sinusoidal theory with adjustable wavenumber parameter n:

$$\mathrm{TBR} = \mathrm{Mean}_{i \in C} \left[\frac{1 + \sin(2\pi n(x_i - 1/2))}{2} \right] \tag{3}$$

plotted in Figure 13 for $n = 1$ and two continuous parameters C. ANNs trained on this model demonstrated similar performance to those on the expensive MC model. QASS performance was verified by training a 1h3f(256) ANN on the sinusoidal theory for varied quantities of initial, incremental, and MCMC candidate samples. Although the scope of this project did not include thorough searches of this hyperparameter domain, sufficient runs were made to identify some likely trends.

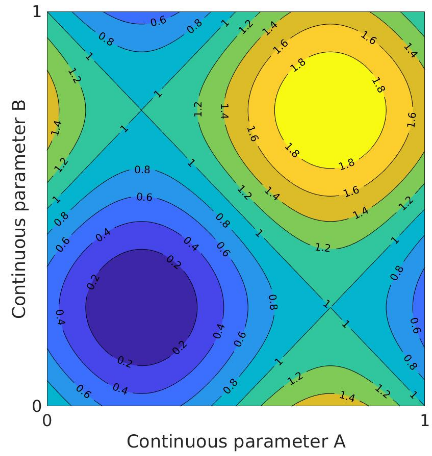


Figure 13: Sinusoidal toy TBR theory over two continuous parameters, wavenumber 1

An increase in MCMC candidate samples was seen to have a positive but very weak effect on final surrogate precision, suggesting that the runtime of MCMC on each iteration can be limited for increased efficiency. – Awaiting test results on initial sample quantity –. The most complex dynamics arose with the adjustment of sample increment, shown in Figure 14. For each tested initial sample quantity N, the optimal number of step samples was seen to be well-approximated by $\sqrt{N}$; the plotted error trends suggest that incremental samples larger than this optimum give slower model improvement on both the training and evaluation sets, and a larger minimum error on the evaluation set. This performance distinction is predicted to be even more significant when trained on the expensive MC model, where the number of sample evaluations will serve as the primary bottleneck for computation time.

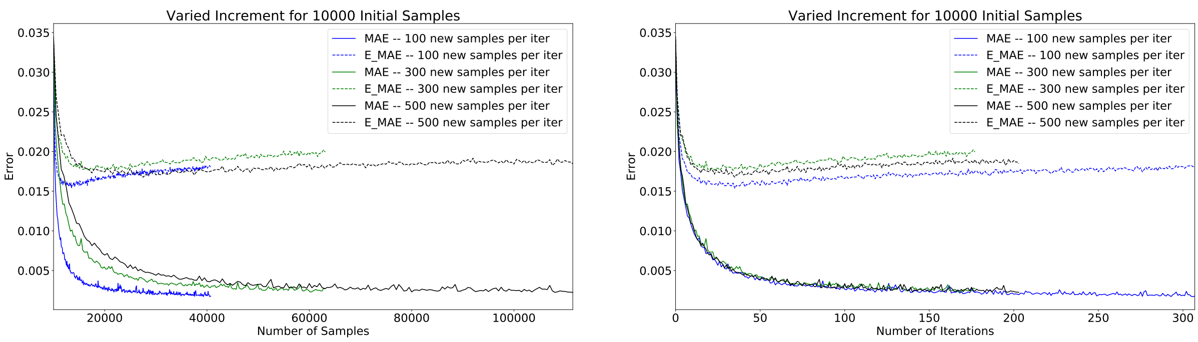


Figure 14: QASS absolute training error over total sample quantity (left) and number of iterations (right). MAE represents surrogate error on the adaptively-sampled training/test set, and E_MAE on the independent evaluation sets.

The plateau effect in surrogate error on the evaluation set, seen in Figure 14, was universal to all configurations and thought to warrant further investigation. At first this was suspected to be a residual effect of retraining the same ANN instance without adjustment to data normalisation; a "Goldilocks scheme" for checking normalisation drift was implemented and tested, but did not affect QASS performance. Schemes in which the ANN is periodically retrained were also discarded, as the retention of network weights from one iteration to the next was demonstrated to greatly benefit QASS efficiency. Further insight came from direct comparison between QASS and a baseline scheme with uniformly random incremental samples, shown in Figure 15.

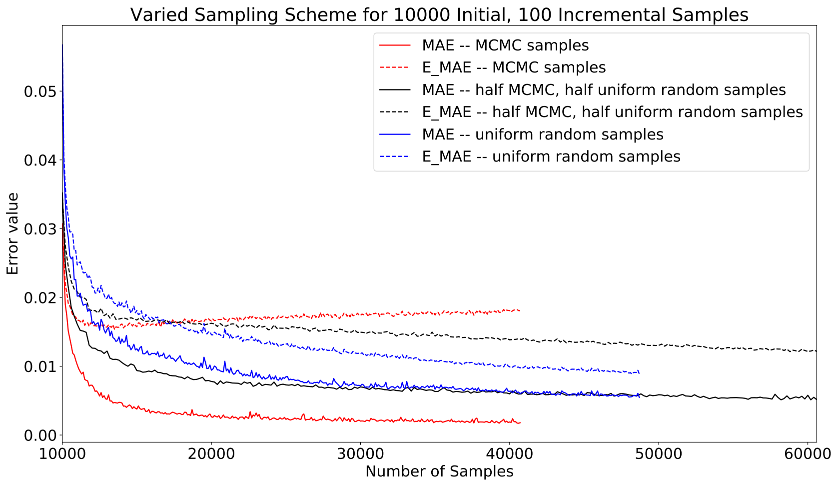


Figure 15: Absolute training error for QASS, baseline scheme, and mixed scheme

Such tests revealed that while QASS has unmatched performance on its own adaptively-sampled training set, it is outperformed by the baseline scheme on uniformly random evaluation sets. We suspected that while QASS excels in learning the most strongly peaked regions of the TBR theory, this comes at the expense of precision in broader, smoother regions where uniformly random sampling suffices. Therefore a mixed scheme was implemented, with half MCMC samples and half uniformly random samples incremented on each iteration, which is also shown in Figure 15.

5 Conclusion

Over the course of this internship project, we employed a broad spectrum of data analysis and machine learning techniques to develop fast and high-quality surrogate models for a MC TBR model in use at UKAEA. We generated 900,000 samples for training and test purposes, evaluated on this expensive MC model. We investigated possibilities for simplification of the parameter space, and concluded that no straightforward reduction was possible. After reviewing N surrogate models (fill in N), and examining their behaviour on (un)constrained feature space and scaling properties, we retrained some of the best-performing surrogates on the full parameter space. The optimum results obtained were an accuracy of X and a mean prediction time of X, representing a relative speedup X with respect to the MC model.

After a thorough review of the literature, we also developed a novel adaptive sampling algorithm, QASS, capable of interfacing with any of the individual studied models. Preliminary testing on a toy theory, qualitatively comparable to the MC TBR model, demonstrated the effectiveness of QASS and behavioural trends consistent with the design of the algorithm. *[Insert numerical results for QASS.]* Further optimisation over the hyperparameter space has strong potential to increase this performance, allowing for future deployment of QASS on the MC TBR model in coalition with any of the most effective identified surrogate models.

References

[1] Jacob Søndergaard. "Optimization Using Surrogate Models". PhD thesis. Technical University of Denmark, 2003.

[2] R.H. Myers and D.C. Montgomery. *Response Surface Methodology: Product and Process Optimization Using Designed Experiments.* 2nd. New York: John Wiley & Sons, 2002.

[3] Jonathan Shimwell. *collaboration.*

[4] F.A. Hernández and P. Pereslavtsev. "First principles review of options for tritium breeder and neutron multiplier materials for breeding blankets in fusion reactors". In: *Fusion Engineering and Design* 137 (Dec. 2018), pp. 243–256. ISSN: 0920-3796.

[5] M Keilhacker. "JET deuterium: tritium results and their implications". In: *Philosophical Transactions of The Royal Society A: Mathematical, Physical and Engineering Sciences* 357 (Mar. 1999), pp. 415–442.

[6] M. Coleman and S. McIntosh. "BLUEPRINT: A novel approach to fusion reactor design". In: *Fusion Engineering and Design* 139 (Feb. 2019), pp. 26–38. ISSN: 0920-3796.

[7] Paul K. Romano et al. "OpenMC: A state-of-the-art Monte Carlo code for research and development". In: *Annals of Nuclear Energy* 82 (2015). Joint International Conference on Supercomputing in Nuclear Applications and Monte Carlo 2013, SNA + MC 2013. Pluri- and Trans-disciplinarity, Towards New Modeling and Numerical Simulation Paradigms, pp. 90–97. ISSN: 0306-4549.

[8] Ian T Jolliffe and Jorge Cadima. "Principal component analysis: a review and recent developments". In: *Philosophical Transactions of the Royal Society A: Mathematical, Physical and Engineering Sciences* 374.2065 (Apr. 2016), p. 20150202.

[9] F. Pedregosa et al. "Scikit-learn: Machine Learning in Python". In: *Journal of Machine Learning Research* 12 (2011), pp. 2825–2830.

[10] Mohamed Amine Bouhlel and Joaquim Martins. "Gradient-enhanced kriging for high-dimensional problems". In: *Engineering with Computers* (Feb. 2018).

[11] Georges Matheron. "Principles of geostatistics". In: *Economic Geology* 58.8 (Dec. 1963), pp. 1246–1266. ISSN: 0361-0128.

[12] *Kriging Variogram Model.* URL: `https://vsp.pnnl.gov/help/Vsample/Kriging%7B%5C_%7DVariogram%7B%5C_%7DModel.htm` (visited on 20/04/2020).

[13] Jürgen Schmidhuber. "Deep learning in neural networks: An overview". In: *Neural Networks* 61 (2015), pp. 85–117. ISSN: 0893-6080.

[14] François Chollet et al. *Keras.* `https://keras.io`. 2015.

[15] Mohamed Amine Bouhlel et al. "A Python surrogate modeling framework with derivatives". In: *Advances in Engineering Software* (2019), p. 102662. ISSN: 0965-9978.

[16] Sushant Garud, Iftekhar Karimi and Markus Kraft. "Smart Sampling Algorithm for Surrogate Model Development". In: *Computers & Chemical Engineering* 96 (Oct. 2016).

[17] Jun Zhou, Xiaosi Su and Geng Cui. "An adaptive Kriging surrogate method for efficient joint estimation of hydraulic and biochemical parameters in reactive transport modeling". In: *Journal of Contaminant Hydrology* 216 (Sept. 2018), pp. 50–57. ISSN: 0169-7722.

[18] Wei Gong and Qingyun Duan. "An adaptive surrogate modeling-based sampling strategy for parameter optimization and distribution estimation (ASMO-PODE)". In: *Environmental Modelling & Software* 95 (Sept. 2017), pp. 61–75. ISSN: 1364-8152.

[19] Jiangjiang Zhang et al. "Surrogate-Based Bayesian Inverse Modeling of the Hydrological System: An Adaptive Approach Considering Surrogate Approximation Error". In: *Water Resources Research* 56.1 (Jan. 2020), e2019WR025721. ISSN: 0043-1397.

[20] Victor Ginting et al. "Application of the two-stage Markov chain Monte Carlo method for characterization of fractured reservoirs using a surrogate flow model". In: *Computational Geosciences* 15.4 (2011), p. 691. ISSN: 1573-1499.

[21] Antti Solonen et al. "Efficient MCMC for Climate Model Parameter Estimation: Parallel Adaptive Chains and Early Rejection". en. In: *Bayesian Anal.* 7.3 (2012), pp. 715–736. ISSN: 1936-0975.

[22] Jie Zhang, Souma Chowdhury and Achille Messac. "An adaptive hybrid surrogate model". In: *Structural and Multidisciplinary Optimization* 46.2 (2012), pp. 223–238. ISSN: 1615-1488.

Appendices

A Detailed Results of Supervised Models

		Regression performance				Complexity	
Surrogate class	#	MAE [TBR]	S [TBR]	R^2 [rel.]	$R^2_{\text{adj.}}$ [rel.]	$\bar{t}_{\text{trn.}}$ [ms]	$\bar{t}_{\text{pred.}}$ [ms]
GBT TODO	1234	1.23	1.23	1.23	1.23	1.23 ± 1.23	1.23 ± 1.23
GBT TODO	1234	1.23	1.23	1.23	1.23	1.23 ± 1.23	1.23 ± 1.23
GBT TODO	1234	1.23	1.23	1.23	1.23	1.23 ± 1.23	1.23 ± 1.23
GBT TODO	1234	1.23	1.23	1.23	1.23	1.23 ± 1.23	1.23 ± 1.23
GBT TODO	1234	1.23	1.23	1.23	1.23	1.23 ± 1.23	1.23 ± 1.23
GBT TODO	1234	1.23	1.23	1.23	1.23	1.23 ± 1.23	1.23 ± 1.23
GBT TODO	1234	1.23	1.23	1.23	1.23	1.23 ± 1.23	1.23 ± 1.23

Table 6: Results of experiment 1 (single slice hyperparameter tuning). Column # gives the number of Bayesian optimisation iterations. While regression performance is reported for the best instance per surrogate class, complexity is given in terms of mean and standard deviation over all tested instances.

		Regression performance				Complexity	
Surrogate class	#	MAE [TBR]	S [TBR]	R^2 [rel.]	$R^2_{\text{adj.}}$ [rel.]	$\bar{t}_{\text{trn.}}$ [ms]	$\bar{t}_{\text{pred.}}$ [ms]
GBT TODO	1234	1.23 ± 1.23	1.23 ± 1.23	1.23 ± 1.23	1.23 ± 1.23	1.23 ± 1.23	1.23 ± 1.23
GBT TODO	1234	1.23 ± 1.23	1.23 ± 1.23	1.23 ± 1.23	1.23 ± 1.23	1.23 ± 1.23	1.23 ± 1.23
GBT TODO	1234	1.23 ± 1.23	1.23 ± 1.23	1.23 ± 1.23	1.23 ± 1.23	1.23 ± 1.23	1.23 ± 1.23
GBT TODO	1234	1.23 ± 1.23	1.23 ± 1.23	1.23 ± 1.23	1.23 ± 1.23	1.23 ± 1.23	1.23 ± 1.23
GBT TODO	1234	1.23 ± 1.23	1.23 ± 1.23	1.23 ± 1.23	1.23 ± 1.23	1.23 ± 1.23	1.23 ± 1.23
GBT TODO	1234	1.23 ± 1.23	1.23 ± 1.23	1.23 ± 1.23	1.23 ± 1.23	1.23 ± 1.23	1.23 ± 1.23
GBT TODO	1234	1.23 ± 1.23	1.23 ± 1.23	1.23 ± 1.23	1.23 ± 1.23	1.23 ± 1.23	1.23 ± 1.23

Table 7: Results of experiment 2 (full feature space hyperparameter tuning), displayed analogously to Table 6. Here, regression performance metrics are reported as means and standard deviations of the best-performing models (in R^2) across 4 tested slices.

	Regression performance as R^2 [rel.] by cross-validation set size						
Surrogate class	1000	2000	5000	10 000	12 000	15 000	20 000
GBT TODO	1.23 ± 1.23	1.23 ± 1.23	1.23 ± 1.23	1.23 ± 1.23	1.23 ± 1.23	1.23 ± 1.23	1.23 ± 1.23
GBT TODO	1.23 ± 1.23	1.23 ± 1.23	1.23 ± 1.23	1.23 ± 1.23	1.23 ± 1.23	1.23 ± 1.23	1.23 ± 1.23
GBT TODO	1.23 ± 1.23	1.23 ± 1.23	1.23 ± 1.23	1.23 ± 1.23	1.23 ± 1.23	1.23 ± 1.23	1.23 ± 1.23
GBT TODO	1.23 ± 1.23	1.23 ± 1.23	1.23 ± 1.23	1.23 ± 1.23	1.23 ± 1.23	1.23 ± 1.23	1.23 ± 1.23
GBT TODO	1.23 ± 1.23	1.23 ± 1.23	1.23 ± 1.23	1.23 ± 1.23	1.23 ± 1.23	1.23 ± 1.23	1.23 ± 1.23
GBT TODO	1.23 ± 1.23	1.23 ± 1.23	1.23 ± 1.23	1.23 ± 1.23	1.23 ± 1.23	1.23 ± 1.23	1.23 ± 1.23
GBT TODO	1.23 ± 1.23	1.23 ± 1.23	1.23 ± 1.23	1.23 ± 1.23	1.23 ± 1.23	1.23 ± 1.23	1.23 ± 1.23

Table 8: Results of experiment 3 (scaling benchmark) in terms of R^2.

	Complexity as $\bar{t}_{\text{trn.}}$ [ms] by cross-validation set size						
Surrogate class	1000	2000	5000	10 000	12 000	15 000	20 000
GBT TODO	1.23 ± 1.23	1.23 ± 1.23	1.23 ± 1.23	1.23 ± 1.23	1.23 ± 1.23	1.23 ± 1.23	1.23 ± 1.23
GBT TODO	1.23 ± 1.23	1.23 ± 1.23	1.23 ± 1.23	1.23 ± 1.23	1.23 ± 1.23	1.23 ± 1.23	1.23 ± 1.23
GBT TODO	1.23 ± 1.23	1.23 ± 1.23	1.23 ± 1.23	1.23 ± 1.23	1.23 ± 1.23	1.23 ± 1.23	1.23 ± 1.23
GBT TODO	1.23 ± 1.23	1.23 ± 1.23	1.23 ± 1.23	1.23 ± 1.23	1.23 ± 1.23	1.23 ± 1.23	1.23 ± 1.23
GBT TODO	1.23 ± 1.23	1.23 ± 1.23	1.23 ± 1.23	1.23 ± 1.23	1.23 ± 1.23	1.23 ± 1.23	1.23 ± 1.23
GBT TODO	1.23 ± 1.23	1.23 ± 1.23	1.23 ± 1.23	1.23 ± 1.23	1.23 ± 1.23	1.23 ± 1.23	1.23 ± 1.23
GBT TODO	1.23 ± 1.23	1.23 ± 1.23	1.23 ± 1.23	1.23 ± 1.23	1.23 ± 1.23	1.23 ± 1.23	1.23 ± 1.23

Table 9: Results of experiment 3 (scaling benchmark) in terms of $\bar{t}_{\text{trn.}}$, displayed analogously to Table 8.

	Complexity as $\bar{t}_{\text{pred.}}$ [ms] by cross-validation set size						
Surrogate class	1000	2000	5000	10 000	12 000	15 000	20 000
GBT TODO	1.23 ± 1.23	1.23 ± 1.23	1.23 ± 1.23	1.23 ± 1.23	1.23 ± 1.23	1.23 ± 1.23	1.23 ± 1.23
GBT TODO	1.23 ± 1.23	1.23 ± 1.23	1.23 ± 1.23	1.23 ± 1.23	1.23 ± 1.23	1.23 ± 1.23	1.23 ± 1.23
GBT TODO	1.23 ± 1.23	1.23 ± 1.23	1.23 ± 1.23	1.23 ± 1.23	1.23 ± 1.23	1.23 ± 1.23	1.23 ± 1.23
GBT TODO	1.23 ± 1.23	1.23 ± 1.23	1.23 ± 1.23	1.23 ± 1.23	1.23 ± 1.23	1.23 ± 1.23	1.23 ± 1.23
GBT TODO	1.23 ± 1.23	1.23 ± 1.23	1.23 ± 1.23	1.23 ± 1.23	1.23 ± 1.23	1.23 ± 1.23	1.23 ± 1.23
GBT TODO	1.23 ± 1.23	1.23 ± 1.23	1.23 ± 1.23	1.23 ± 1.23	1.23 ± 1.23	1.23 ± 1.23	1.23 ± 1.23
GBT TODO	1.23 ± 1.23	1.23 ± 1.23	1.23 ± 1.23	1.23 ± 1.23	1.23 ± 1.23	1.23 ± 1.23	1.23 ± 1.23

Table 10: Results of experiment 3 (scaling benchmark) in terms of $\bar{t}_{\text{pred.}}$, displayed analogously to Table 8.

		Regression performance				Complexity		
Surrogate	Class	MAE [TBR]	S [TBR]	R^2 [rel.]	$R^2_{\text{adj.}}$ [rel.]	$\bar{t}_{\text{trn.}}$ [ms]	$\bar{t}_{\text{pred.}}$ [ms]	σ [rel.]
Model 1	TODO	1.23 ± 1.23	1.23 ± 1.23	1.23 ± 1.23	1.23 ± 1.23	1.23 ± 1.23	1.23 ± 1.23	$1234\times$
Model 1	TODO	1.23 ± 1.23	1.23 ± 1.23	1.23 ± 1.23	1.23 ± 1.23	1.23 ± 1.23	1.23 ± 1.23	$1234\times$
Model 1	TODO	1.23 ± 1.23	1.23 ± 1.23	1.23 ± 1.23	1.23 ± 1.23	1.23 ± 1.23	1.23 ± 1.23	$1234\times$
Model 1	TODO	1.23 ± 1.23	1.23 ± 1.23	1.23 ± 1.23	1.23 ± 1.23	1.23 ± 1.23	1.23 ± 1.23	$1234\times$
Model 1	TODO	1.23 ± 1.23	1.23 ± 1.23	1.23 ± 1.23	1.23 ± 1.23	1.23 ± 1.23	1.23 ± 1.23	$1234\times$
Model 1	TODO	1.23 ± 1.23	1.23 ± 1.23	1.23 ± 1.23	1.23 ± 1.23	1.23 ± 1.23	1.23 ± 1.23	$1234\times$
Model 1	TODO	1.23 ± 1.23	1.23 ± 1.23	1.23 ± 1.23	1.23 ± 1.23	1.23 ± 1.23	1.23 ± 1.23	$1234\times$

Table 11: Results of experiment 4 (competitive training) displayed analogously to Table 7. In addition, we also report relative speedup with respect to the MC TBR model with baseline X.

B Overview of Online Resources

TODO: point reader to various online repositories, briefly describe them

1. data repository
2. sampling repository
3. regression repository
4. hyperopt repository
5. docs repository