Computers and Chemical Engineering 68 (2014) 220–232

Contents lists available at ScienceDirect

Computers and Chemical Engineering

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

Adaptive sequential sampling for surrogate model generation with artificial neural networks

John Eason, Selen Cremaschi*

Department of Chemical Engineering, The University of Tulsa, 800 South Tucker Drive, Tulsa, OK 74104, USA

ARTICLE INFO

Article history:
Received 29 January 2014
Received in revised form 3 April 2014
Accepted 20 May 2014
Available online 7 June 2014

Keywords:
Adaptive sampling
Space-filling design
Artificial neural networks
Surrogate models

ABSTRACT

Surrogate models – simple functional approximations of complex models – can facilitate engineering analysis of complicated systems by greatly reducing computational expense. The construction of a surrogate model requires evaluation of the original model to gather the data necessary for building the surrogate. Sequential sampling procedures are proposed for determining and minimizing the required number of samples for efficient global surrogate construction. In this paper, two new adaptive sampling algorithms – one purely adaptive and one combining adaptive and space-filling characteristics – are proposed and compared to a purely space-filling approach. Our analysis suggests a mixed adaptive sampling approach for constructing surrogates for systems where the behavior of the underlying model is unknown. Results of the case study, optimization of carbon dioxide capture process with aqueous amines, revealed that the mixed adaptive sampling algorithm may reduce the required sample size by up to 40% compared to a purely space-filling design.

1. Introduction

Complex mathematical models have become an integral part of engineering with the rise of powerful computing tools. However, design space exploration, optimization, and sensitivity analysis of systems defined by these models are difficult because of the high computational cost of each simulation or model evaluation. Because such engineering analysis usually requires many model evaluations, working directly with complex models may become prohibitively expensive. This problem becomes especially evident for systems involving multi-scale processes such as biomass gasification process, which requires the consideration of gasification kinetics (Lang et al., 2009, 2011) or algae growth in two-phase packed columns, which requires the consideration of hydrodynamics effects (Smith et al., 2012, 2013).

A surrogate model (a.k.a. metamodel or reduced-order model) can be thought of as a "regression" to a set of data, where the data is a set of input–output pairings obtained by evaluating a black-box model of the complex system (Caballero and Grossmann, 2008). Surrogate models may be classified into two general categories based on their purpose: local and global models (Crombecq et al., 2009).

The popular response surface methodology (RSM) is an example of optimization using local, polynomial surrogate models. RSM sequentially fits local first and second order regression models to a small region of the overall search space, and use these regression models to search for directions to improve the objective function and to locate the optimum. The local regression models only provide information about the impact of inputs on the outputs within the small region, and are generally discarded once they are used for determining the improvement direction. RSM has been successfully used in many applications, see (Baş and Boyacı, 2007) for a review of use in chemical and biochemical processes. However, it has several drawbacks. RSM lacks the ability to model functions that cannot be approximated by first or second order responses, and the application of RSM to high-dimensional problems is sometimes problematic (Wan et al., 2005).

A global surrogate model is a function that approximates the system across the design space. If coupled with global optimization algorithms, the solution obtained using these models can be shown to be global within the accuracy of the surrogate models and the bounds of the data used to generate them. Although additional effort is required to construct a global surrogate model, including possibly more samples in suboptimal regions, these models may be retained for sensitivity analysis and further exploration of the design space. The algorithms developed in this paper are focused on this global surrogate modeling approach. Two popular global surrogate models are kriging interpolators and artificial neural networks (including radial basis function networks) (Henao and

* Corresponding author. Tel.: +1 918 631 3422; fax: +1 918 631 3268.
E-mail addresses: selen-cremaschi@utulsa.edu, selen.aydogan@gmail.com (S. Cremaschi).

http://dx.doi.org/10.1016/j.compchemeng.2014.05.021
0098-1354/

Maravelias, 2010). Throughout this paper, "underlying function" is used to refer to the complex model that is approximated with a surrogate. Kriging interpolators fit a spatial correlation function to a data set consisting of input–output pairs obtained by evaluating the underlying function. These data points are used in a linear interpolation with weights derived from the spatial correlation function (Kleijnen, 2009). Kriging interpolators have been used in few examples in process synthesis and optimization of chemical processes (Caballero and Grossmann, 2008; Davis and Ierapetritou, 2009). One disadvantage of kriging interpolators is the lack of available software suitable for engineering applications and potential numeric problems with a poorly conditioned sample matrix. Artificial neural networks (ANNs) are mathematical models consisting of interconnected simple processing units known as neurons (Basheer and Hajmeer, 2000). The parameters of a network include weights and biases, and the operation performed at each neuron (transfer function). The problem of fitting these parameters is known as "training" the network. One disadvantage of ANNs is the need for providing the number of neurons, which must be sufficiently high to capture functional behavior but not so high as to cause overfitting. In addition, the behavior of the neural network may exhibit undesirable oscillatory behavior between sample points. This may be somewhat offset by using a regularization term during fitting. In this work, we use Bayesian regularization, which derives regularization weights from Bayesian statistics. The ANNs have been applied to many chemical engineering problems, including the optimization of nylon-6,6 polymerization and acetic anhydride production (Nascimento et al., 2000), the modeling and superstructure optimization of biodiesel production (Fahmi and Cremaschi, 2012; Yuste and Dorado, 2005), and the optimization of Fischer–Tropsch synthesis (Fernandes, 2006). Henao and Maravelias (2010, 2011) present a superstructure optimization framework using ANNs as surrogate models, with specific examples of maleic anhydride production from benzene.

Artificial neural networks are selected as the surrogate model in our work due to their previous successful use in chemical engineering, applicability to higher dimensional problems, ease of use, simplicity to construct, and availability in several existing software packages. To construct ANN's, or any surrogate model, one needs to collect input–output data from the original (i.e., underlying) model. The number and location of these data points are important for the overall accuracy of the surrogate model and controlling the computational expense of its construction. The goal of this paper is to propose sample generation procedures that facilitate efficient construction of accurate ANN surrogate models for the overall design space, and to investigate their performances through numerical experiments with challenge functions. We present two adaptive sequential sample generation algorithms, which scale well to problems with high dimensions, and compare their performances to each other and to a purely space-filling sampling approach (*incremental* Latin hypercube Sampling, Nuchitprasittichai and Cremaschi, 2013) using a set of known challenge functions. Based on our numerical experiments with the challenge functions, the performance of the selected algorithm is compared to a purely space-filling sampling approach (Nuchitprasittichai and Cremaschi, 2013) for locating the design variables and operating conditions that minimize the cost for amine-based carbon dioxide (CO_2) capture process. Our results suggest that an adaptive sampling approach is required for accurately modeling strong nonlinearities and for reducing the number of underlying function evaluations to yield similar surrogate model accuracies when compared to a purely space-filling sampling approach.

The paper is organized as follows: Next section reviews the past sampling approaches for global surrogate model construction. Then, we give an overview of *incremental* Latin hypercube Sampling (*iLHS*) algorithm (Nuchitprasittichai and Cremaschi, 2013) followed by the details of our proposed two adaptive sample generation algorithms. The numerical experiments with known challenge functions for comparing these three algorithms are discussed next. Finally, the selected algorithm is applied for a case study of optimization of CO_2 capture with aqueous amines, followed by conclusions and future recommendations.

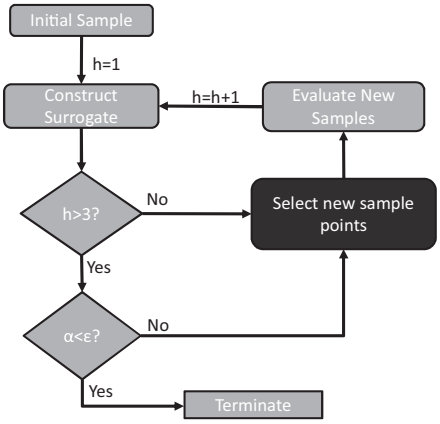

Fig. 1. Surrogate construction algorithm with sequential sampling approach.

2. A brief literature review on sampling approaches for global surrogate construction

Traditionally, global surrogate models have been constructed as follows (Gorissen et al., 2007): an experimental design is selected, and a test matrix is constructed containing the input sets generated according to the experimental design. The underlying model is evaluated at each input set of the test matrix to obtain the corresponding outputs. A surrogate model is constructed, fit to the collected input–output data, and its performance is assessed. Experimental designs typically chosen for computational experiments include factorial design (grid sampling), Latin hypercube sampling, and orthogonal arrays (Queipo et al., 2005; Simpson et al., 2001). If the performance of the model is deemed unsatisfactory, the gathered data and the surrogate model are discarded, and the whole process begins again with a new, larger sample size from the experimental design stage. This approach, referred to as the "one-shot" design (Crombecq et al., 2009), was used in most chemical engineering studies, which used neural networks (Fahmi and Cremaschi, 2012; Fernandes, 2006; Henao and Maravelias, 2010, 2011; Yuste and Dorado, 2005). However, when the underlying function is complex and time consuming to evaluate, the experimental design becomes very important in efforts to reduce the number of underlying function evaluations, and hence, the computational cost of building the surrogate. There are two central questions that should be addressed at the experimental design stage: how many samples are necessary, and where should these samples be located?

Sequential sampling determines the minimum necessary sample size by beginning with a small sample size and slowly adding data until the surrogate model performance reaches some desired measure of accuracy. By systematically increasing the sample size while maintaining previous sampled points, sequential methods have been shown to require fewer observations, i.e., function evaluations, than one-shot procedures, hence reducing the total time required for data collection (Ghosh and Sen, 1991; Peng et al., 2004; Provost et al., 1999). Fig. 1 gives a general algorithm for constructing surrogate models using sequential sampling. An initial set of input data and the corresponding function evaluations, typically generated using a grid or Latin hypercube design, are used to construct the first surrogate model. Then, the additional sample points are

selected based on some objective (discussed in the coming paragraphs). The underlying function is evaluated at these new points. These input–output pairs are added to the total data set, and all the data is used to construct a new surrogate model. A metric, α, that defines the relative improvement in the surrogate model's performance is checked against a target value, ε. When α falls below the target value (i.e., the performance of the surrogate model is deemed satisfactory), the algorithm terminates and returns the last surrogate model specifics. The performance metric, α, typically quantifies the relative improvement observed in a measure of the prediction error of the surrogate model. Hence, if the addition of new sample points does not improve the performance of the surrogate model, the sample size is assumed to be adequate. The counter variable, h, accounts for the simple fact that at least two iterations (i.e., $h = 3$) are required before the relative improvement metric, α, can be calculated. The performance of sequential sampling algorithms depends on obtaining an appropriate sample distribution, i.e., selecting the appropriate sample locations at each iteration. We focus on this step of the algorithm (highlighted in in dark Fig. 1) in this paper.

There are two goals that may be used for determining the location of new sample points. First, the new samples should be far away from existing points to avoid redundant samples. This goal is referred to as the space-filling objective. On the other hand, one would like to also add new sample points in the regions of the design space that are more "interesting," i.e., more nonlinear, discontinuous or inconsistent with the general trends of the underlying function. This goal is known as adaptive sampling or active learning objective. The contrast between these two approaches in terms of the quality of fit is known as the "exploration versus exploitation" problem (Crombecq, 2011), and is illustrated with an example in Fig. 2. In Fig. 2a, the underlying function has been sampled at evenly spaced intervals. A surrogate model developed using these sample points and the corresponding function evaluations would not accurately reproduce the minimum of the underlying function and would miss the peak completely. Fig. 2b shows where the new samples might be generated if a purely space-filling strategy were followed. With this new sample set, the existence of another nonlinear region (i.e., the peak) is discovered. However, with limited number of sample points in the nonlinear regions, the specific shape and extrema in these regions remain unknowns. By contrast, an adaptive sampling approach (exploitation) would focus on generating more samples in the already-identified nonlinear regions and would provide a better representation of that region (Fig. 2c). However, the adaptive sampling approach would completely miss the nonlinear regions that were previously not identified.

The most well-known sampling methods are space-filling techniques. Grid sampling (factorial designs) is frequently used in one-step designs. They could be used in sequential sampling approaches by refining the grids; however, the required number of samples increases exponentially at each iteration. For example, if a two dimensional 9×9 grid sample (81 sample points) was not sufficient to construct an accurate surrogate model, at the next iteration, an additional 208 sample points, i.e., underlying function evaluations, would be required to form a new, evenly spaced 17×17 grid that can retain the previous 81 points. Therefore, grid sampling is not an efficient approach in sequential sampling methods. The *incremental* Latin hypercube sampling algorithm (Nuchitprasittichai and Cremaschi, 2013) provides a space-filling Latin hypercube design at each iteration. This algorithm is briefly discussed in the next section. Techniques have been proposed using Delaunay triangulations (Davis and Ierapetritou, 2010) and their dual structure, Voronoi tessellations (Crombecq et al., 2011). These structures subdivide the sample space based on the position of existing samples, and identify large volumes without a sample point. While effective in problems with two or three dimensions, the process of constructing Delaunay triangulations or Voronoi tessellations becomes prohibitively expensive with higher dimensions. Low discrepancy sequences, such as the Halton and Sobol sequences, have been proposed for space-filling sequential sampling (Giunta et al., 2003). However, Chi et al. (2005) discuss the poor space coverage characteristics of these approaches in high dimensions. Even in low dimensions, Crombecq et al. (2011) demonstrated that low-discrepancy sequences performed worse than competing space-filling approaches, including Voronoi cell sampling, Latin hypercube, and their own optimization-based approaches, in terms of their space-filling qualities estimated by average nearest neighbor distance or average nearest projected distance (distance between points when projected down to any one dimension). The optimization-based approaches determine the location of each successive sample point by maximizing the nearest neighbor distance or the projected distance. Because the optimization becomes computationally expensive as the number of dimensions increases, Crombecq et al. (2011) proposed a "Monte-Carlo-based" approach as an alternative. The Monte-Carlo-based method selects new sample points from a large set of randomly generated proposed points on the basis of the desired objective function. The authors concluded that this approach provided the best performance for the required computational time, especially as the number of dimensions increases.

Adaptive sampling approaches for constructing global surrogates generally rely on the prediction uncertainty of the surrogate models to generate the new sample points. Many studies have developed methods using the variance predictor and statistics of kriging models to construct adaptive sampling schemes, the most famous of which is the Efficient Global Optimization algorithm (Jones et al., 1998). Kleijnen and Van Beers (2004) used the statistical resampling method of jackknifing to provide an alternative measurement of the uncertainty of kriging interpolator surrogate predictions. Devabhaktuni and Zhang (2000) developed an adaptive sampling algorithm for constructing and training artificial neural networks for microwave modeling. The algorithm partitions the search space into 2^n regions of equal volume where n is the number of input variables (i.e., dimensions) and generates sample points using a central composite face-centered distribution in the partition with the highest average error. Only regions with the greatest error are split into 2^n regions in consecutive iterations. This process is repeated until a satisfactory surrogate model is constructed. Although the algorithm is simple and efficient, for problems with high dimensions (a large number of input variables), the number of function evaluations required to sample each new set of partitions (2^n) is problematically large, resulting in oversampling. A final proposed method for adaptive sampling is local linear approximations, as used in the LOcal Linear Approximations of the system (LOLA) – Voronoi algorithm (Crombecq, 2011). This method tries to approximate the gradient at a proposed point by selecting a set of nearby sampled points and fitting a first order surface. These points are selected using a multi-objective optimization function, trying to minimize the distance from the proposed point but maximize the distance from each other. The set of optimization problems that must be solved for each proposed point makes this approach computationally expensive as dimension and total sample size increase.

3. The sample generation algorithms

This section presents three sample generation algorithms we tested for efficient global surrogate model construction using ANNs. All three algorithms determine the locations of the new sample points within the search space in a sequential sampling

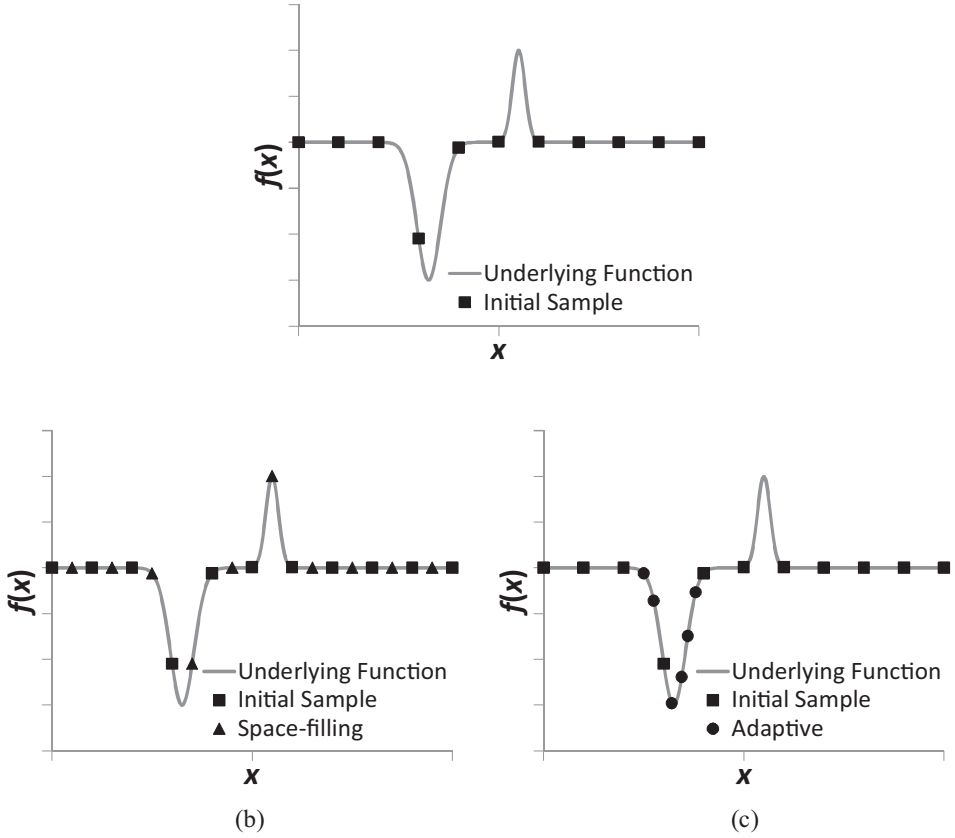

Fig. 2. Exploration vs. exploitation problem: (a) underlying function with initial sample, (b) space-filling sampling, (c) adaptive sampling (Crombecq, 2011).

framework (highlighted portion of Fig. 1). The first algorithm is a purely space filling algorithm and was developed previously within our group (Nuchitprasittichai and Cremaschi, 2013). The second and third algorithms have adaptive or/and space-filling characteristics, and they are contributions of this paper.

3.1. Incremental Latin hypercube sampling (iLHS) algorithm

The *iLHS* algorithm provides a purely space-filling design at each sequential step for any number of samples added (Nuchitprasittichai and Cremaschi, 2013). It was developed as an extension of the Latin hypercube sampling approach, which is an efficient method for sampling from a multidimensional distribution. In Latin hypercube sampling (LHS), each variable of the d variables in the sample space is divided into n equal probability intervals, where n is the number of samples that will be generated. A sample point is selected from each interval. This yields n coordinates for each input variable. Drawing one coordinate randomly from each input variable, a sample point is obtained. Repeating without replacement, n sample points in d dimensional space is generated. The ease of implementation at high dimensions has made LHS method very popular (Jin et al., 2001). Because of the characteristics of LHS approach, the number of sample points must be known in advance in order to construct the sample intervals. If a new point were added randomly, some intervals may become overpopulated and the resulting design will not satisfy the conditions for a Latin hypercube design. The *iLHS* algorithm overcomes this shortcoming of the LHS. Given any arbitrary new sample size, the *iLHS* algorithm divides the variables into equal probability intervals according to the new sample size, discards the overlapping samples within the new intervals, and fills the remaining empty intervals preserving the Latin hypercube design of the sample points at each iteration.

Without the *iLHS*, all sample points would have to be discarded to create a larger Latin hypercube design. The *iLHS* algorithm allows reuse of some previous underlying function evaluations. Conversely, it would be best if all previous function evaluations could be reused. The following proposed two sampling algorithms reuse all previous sample points, thus minimizing the number of underlying function evaluations required to increase the sample size.

3.2. Pure adaptive sampling algorithm

The pure adaptive sampling algorithm generates an estimate of surrogate-model-prediction variance to determine the next set of sample points. The variance is estimated using the jackknifing method, as used by Kleijnen and Van Beers for kriging (Kleijnen and Van Beers, 2004). Jackknifing is a statistical technique that may be used to estimate the variance of a parameter using subsamples of the total parent sample (Wolter, 2007). K subsamples are created by leaving out K disjoint sets (of size $1/K$th of the parent sample) from the parent sample. The parameter of interest is determined for each subsample. The variance of the values from the subsamples is an estimate of the variance of the parameter for parent sample. In the pure adaptive algorithm, the parameter of interest is the prediction from the surrogate model. The jackknifing method is used to estimate the variance of the output (i.e., the prediction) at any given point. The main hypothesis of this sampling algorithm is that sample points where the predictions of the surrogate models (trained using the subsamples generated according to jackknifing method) do not agree, i.e., points where the estimate of the prediction variance is high, are the points that should be added to the sample set. We postulate that as new samples are added, the prediction variance throughout the design space would be reduced, and the overall accuracy of the surrogate model over the design space will improve. The pure adaptive sampling algorithm is given in Fig. 3.

$F[X]$ = neural network trained using data set X
S = Set of existing sample points and the corresponding outputs
For i = 1 to K
    $X_i = 1/K^{th}$ of S where $X_i \backslash X_{\neq i} = \varnothing$
    Train $F_i[S \backslash X_i]$
end for
P = Set of proposed points, randomly generated for all $p \in P$
    For i = 1 to K
        PredictionSet$_i$ = $F_i[p]$
    end for
    $\hat{\sigma}_p^2 \leftarrow$ variance of PredictionSet
end for
Select points p with greatest values of $\hat{\sigma}_p^2$

Fig. 3. Pure adaptive sampling algorithm.

The first step is to divide the existing sample set (S) into K subsets, each containing one Kth of the existing sample set. Next, K neural networks are trained with the existing sample set (S) where one of the K subsets is omitted. During training, a different subset is omitted each time a neural network is trained. A large set of proposed sample points, P, is randomly generated. The variance of the predictions from the set of networks is calculated at each $p \in P$, and the points with the highest values of this estimated predictor variance, $\hat{\sigma}_p^2$, are added to the overall sample set as the new sample points.

3.3. Mixed adaptive sampling algorithm

A third sampling algorithm, termed mixed adaptive sampling, was developed using a combination of space-filling and adaptive approaches. The mixed adaptive sampling algorithm, given in Fig. 4, uses a sample-point quality parameter η, which considers the Euclidean distance between the suggested sample point and its nearest neighbor, and the estimated predictor variance (pure adaptive sampling algorithm). For a proposed point p, "nnd_p" stands for the nearest neighbor distance, and $\hat{\sigma}_p^2$ is the estimated predictor variance. The sample-point quality parameter for each proposed point p, η_p, is calculated as the weighted sum of the estimated predictor variance and the nearest neighbor distance, each normalized relative to their corresponding highest values in the set of proposed sample points. The maximum possible value of sample-point quality parameter is 2. A sample point with $\eta_p = 2$ would indicate that this point has both the highest estimated predictor variance (and hence the highest uncertainty in the surrogate model predictions) and the largest nearest neighbor distance (and hence it is the point that is farthest away from any other point in the set). The proposed points with the highest values of η_p are selected as new sample points.

4. Comparison of the algorithms – numerical experiments

The algorithms were compared based on their performance to replicate the behavior of three challenge functions: the

$F[X]$ = neural network trained using data set X
S = Set of existing sample points $\{s_1, s_2, \ldots s_N\}$ and the corresponding outputs
For i = 1 to K
    $X_i = 1/K^{th}$ of S where $X_i \backslash X_{\neq i} = \varnothing$
    Train $F_i[S \backslash X_i]$
end for
P = Set of proposed points, randomly generated
For all $p \in P$
    For i = 1 to K
        PredictionSet$_i$ = $F_i[p]$
    end for
    $\hat{\sigma}_p^2 \leftarrow$ variance of PredictionSet
    $nnd_p \leftarrow$Euclidean distance between p and s_1
    For j = 2 to N
        Y = Euclidean distance between p and s_j
        If $Y < nnd_p$
            $nnd_p \leftarrow Y$
    end for
end for
For all $p \in P$
    $\eta_p = nnd_p / \max_P(nnd_p) + \hat{\sigma}_p^2 / \max_P(\hat{\sigma}_p^2)$
end for
Select points p with greatest values of η_p

Fig. 4. Mixed adaptive sampling algorithm.

2-Dimensional Shekel function on a domain of $[0,10]^2$, the Ackley function on a domain of $[-2,2]^2$, and MATLAB's Peaks function on a domain of $[-3,3]^2$. The graphical representations of these challenge functions are given in Fig. 5, and the equations of the functions may be found in Supplementary Materials. The neural network toolbox in MATLAB was used to construct the one-hidden-layer feed-forward networks, and the networks were trained using Bayesian regularization backpropagation algorithm. The tangent sigmoid transfer function was used for each neuron.

The chosen model evaluation technique was cross-validation. In order to demonstrate the efficacy of a neural network, typically some data is withheld while the network is trained, and then used as validation data on the final network. When data generation is difficult or expensive, cross-validation is an approach for maximizing the utility of a finite data set (Forrester and Keane, 2009). The data set is randomly divided into K equal sized subsets. K neural networks are trained, leaving out one subset each time. The left-out subset is used as the validation data. The sum of squared error is computed for each point when it is in the validation subset. The sum of squared cross-validation error (*SSCVE*) for the full data set is used as an indicator of the fit, such that in Fig. 1, α represents the relative improvement in *SSCVE*.

The sufficient number of neurons for each challenge function was determined by observing network performance trained to a dense grid sample as the number of neurons varied. This allowed us to focus only on the impact of sampling algorithms on the accuracy of the generated neural networks. The optimal

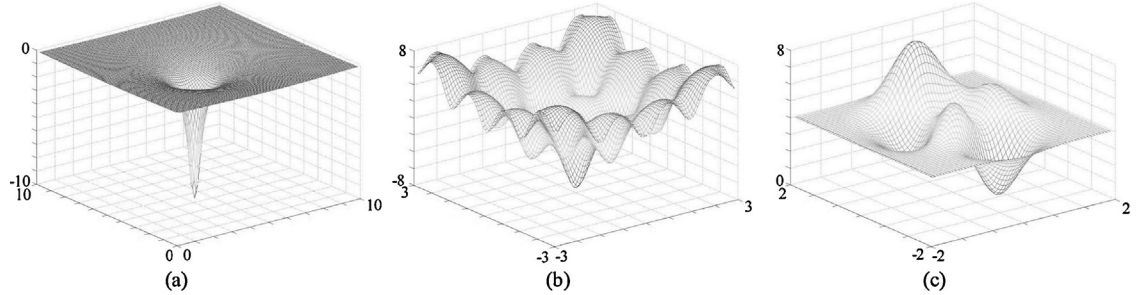

Fig. 5. Graphs of challenge functions: (a) 2-dimensional Shekel function on a domain of $[0,10]^2$; (b) Ackley function on a domain of $[-2,2]^2$; (c) MATLAB's Peaks function on a domain of $[-3,3]^2$.

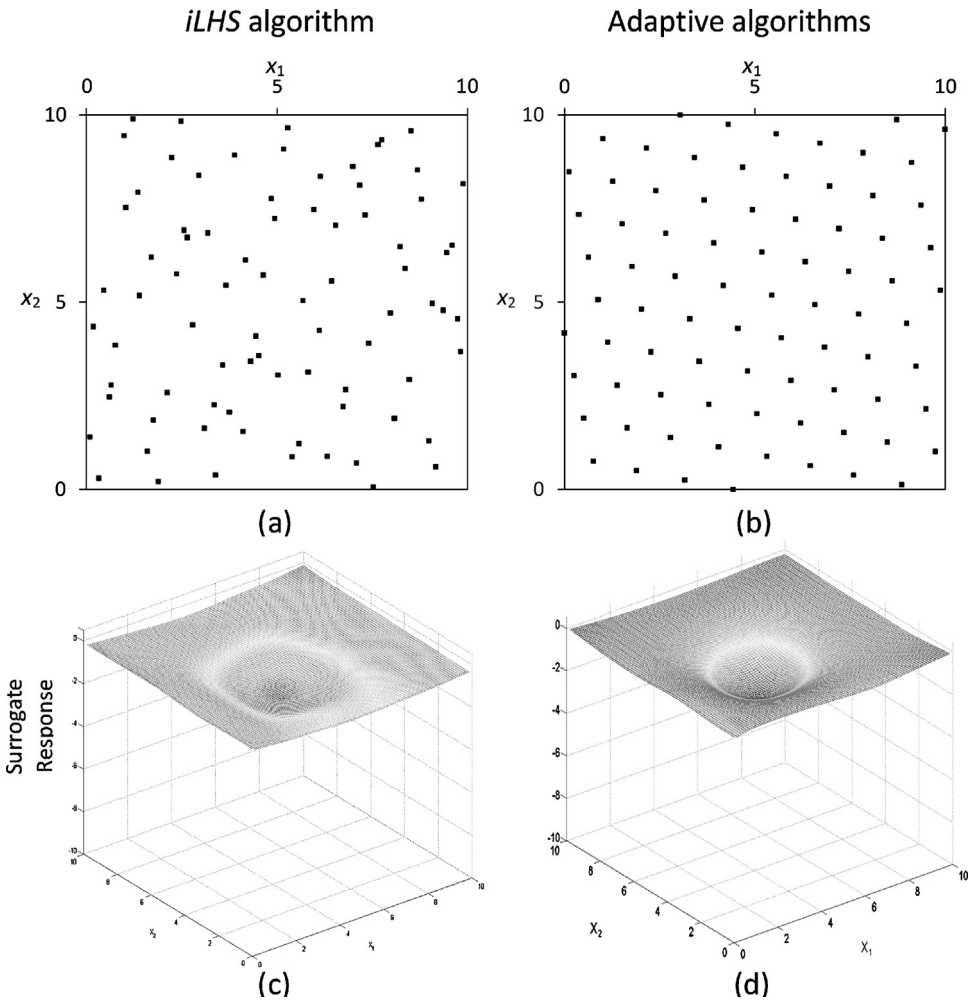

Fig. 6. Sample points (a and b) and the responses of the neural networks trained to these data (c and d) at the first ($h = 1$) iteration of the *iLHS* and (pure and mixed) adaptive sampling algorithms.

number of neurons was determined as the lowest number of neurons at which the *SSCVE* of the neural networks for each function was stabilized: at 12 neurons, 14 neurons, and 15 neurons for the Shekel, Ackley, and Peaks functions, respectively. Based on these architectures, an initial sample size of 80 points was chosen to provide sufficient degrees of freedom to prevent overfitting (a network with two inputs and 15 neurons has 61 parameters). In the numerical experiments, the algorithms were run up to 10 iterations, a point at which it was clear that the cross-validation errors had stabilized for all methods. The sample size was increased geometrically with the geometric factor β equal to 0.3. In other words, at each iteration, the number of new samples is 0.3 times the current sample size. In general, increasing sample sizes geometrically is a more efficient approach than increasing them arithmetically because it avoids high numbers of iterations if the initial sample size estimate were too small (Provost et al., 1999).

The differences between the performances of the sampling algorithms were most pronounced for the Shekel function. Therefore, these results will first be covered in detail, followed by brief discussions of the results for the Ackley and Peaks functions in the sections below. The graphs of the sample point locations and the corresponding surrogate model response at each increment of the sample size for each function are provided in Supplementary Materials. Each algorithm was used to "Select new sample points" (Fig. 1) in a sequential sampling scheme for ten iterations.

4.1. Results for the Shekel function

Fig. 6 gives the 80-point Latin hypercube designs on a domain of $[0,10]^2$ used as the initial designs for the *iLHS* (Fig. 6(a)) and adaptive (Fig. 6(b)) sampling algorithms. Fig. 6(a) displays the randomly generated Latin hypercube design, and Fig. 6(b) is the pre-optimized one for 80 points for two dimensions obtained from (Dam et al., 2009). Based on our experiments, choosing a pre-optimized design as an initial sample results in large unexplored spaces when used with the *iLHS* algorithm, the problem was not observed in adaptive sampling schemes. If a pre-optimized design were not available for a given sample size and dimension, a randomly generated one should suffice, nevertheless when available, these designs provide a good initial exploration of the design space for adaptive sampling approaches. The plot of the predictions made by the neural networks trained using these first data sets are given in Fig. 6(c) and (d). The Shekel function, shown in Fig. 5(a), is nearly linear across much of the selected domain of $[0,10]^2$ with the exception of a strongly nonlinear dip centered at (4, 4). Comparing Fig. 6(c) and (d) with Fig. 5(a), it is clear that the surrogate model fails to capture the sharpness of the nonlinearity, and does not come close to representing the magnitude of the actual minimum.

As the sample size increases according to *iLHS* algorithm, the predictions of the surrogate model (i.e., surrogate-model response) slowly improve, but these improvements are only marginal. Fig. 7(a) shows the final sample set with 1130 samples after 10

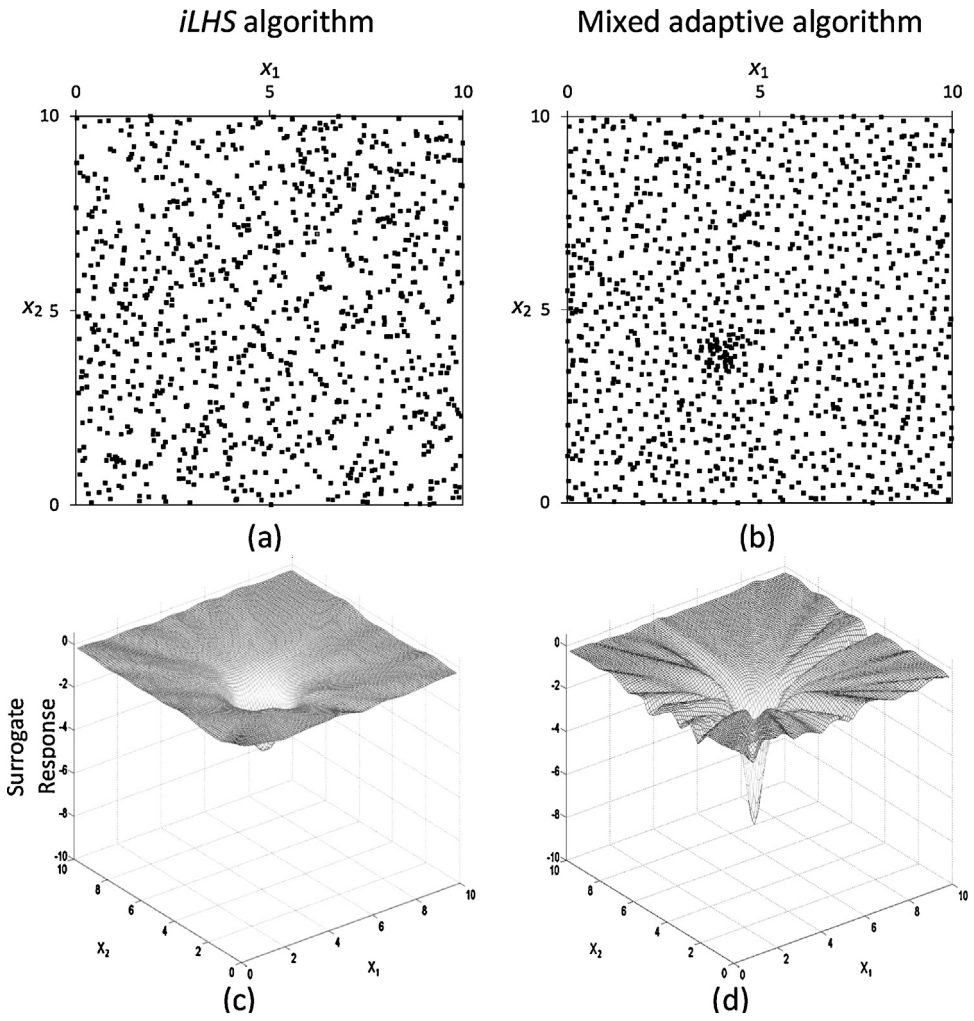

Fig. 7. Sample points (a and b) and the response of the neural networks trained to these data (c and d) at the 10th ($h = 10$) iteration of the *iLHS* and mixed adaptive sampling algorithms.

iterations of the *iLHS* algorithm (at $h = 10$). The surrogate-model response of the neural network trained using this data is shown in Fig. 7(c). This response reveals that the neural network fails to capture the highly nonlinear behavior of the Shekel function even with a large sample size when trained to a set of data that is only space-filling. This final neural network predicts a minimum of −3.56 at (4, 4), whereas the actual minimum is −10.54.

The sample sets and the resulting surrogate responses for the second ($h = 2$), the fourth ($h = 4$) and the last ($h = 10$) iterations of the pure adaptive algorithm are presented in Fig. 8. The response curve at second iteration demonstrates how adding more points in the nonlinear region results in more accurate representations of the nonlinearity. The minimum of the surrogate model trained at the second iteration is −7.04, much closer to the actual minimum of the Shekel function. The surrogate model trained to the initial Latin hypercube had a large prediction variance in the nonlinear region of the Shekel function centered at (4, 4). Therefore, at iteration two when the adaptive sampling algorithm added 30 points, 28 of the 30 fell within a two by two square centered at (4, 4) (Fig. 8, $h = 2$). The higher density of sample points in the nonlinear region resulted in a more accurate representation of that region. However, the predictions of the surrogate model at the linear regions of the Shekel function deteriorated (Fig. 8, $h = 2$). Training neural networks is an optimization problem with an objective to minimize the error between the network predictions and the original evaluations of the underlying function. Therefore, accuracy is favored in regions with a greater density of samples. The form of the neural network function also imposes restrictions, and the waves shown in the linear region are typical of neural network responses.

At iteration three, the pure adaptive algorithm discovers that the area of greatest prediction uncertainty of the surrogate model is now in the linear region near the origin. After sampling in this region and training the new surrogate models, the dip of the Shekel function is absent in the surrogate response surface (Table 2 – Row 3 in Supplementary Materials). Iteration four adds points on the opposite corner of the input space near (10, 10) as shown in Fig. 8, $h = 4$. This sample set still fails to train a network that captures the nonlinearity. The nonlinear behavior (dip) of the Shekel function is re-captured by the surrogate model trained to the sample set of iteration five. The response surface of the surrogate model at this iteration is similar to that of Fig. 8, $h = 2$ (Table 2 – Row 2 in Supplementary Materials). During this iteration, 38 points are added around the dip and 32 points are added scattered throughout the overall domain (in the two previous iterations, no points were added around the dip). During iteration six, there were only two samples added within a two by two square around the minimum, resulting in a sharper but shallower dip in the response surface of the surrogate. The sample points added during iteration seven increases the depth of the nonlinear region predictions (of the surrogate model) again toward the actual minimum (similar to the response in iteration two), and this behavior and the resulting

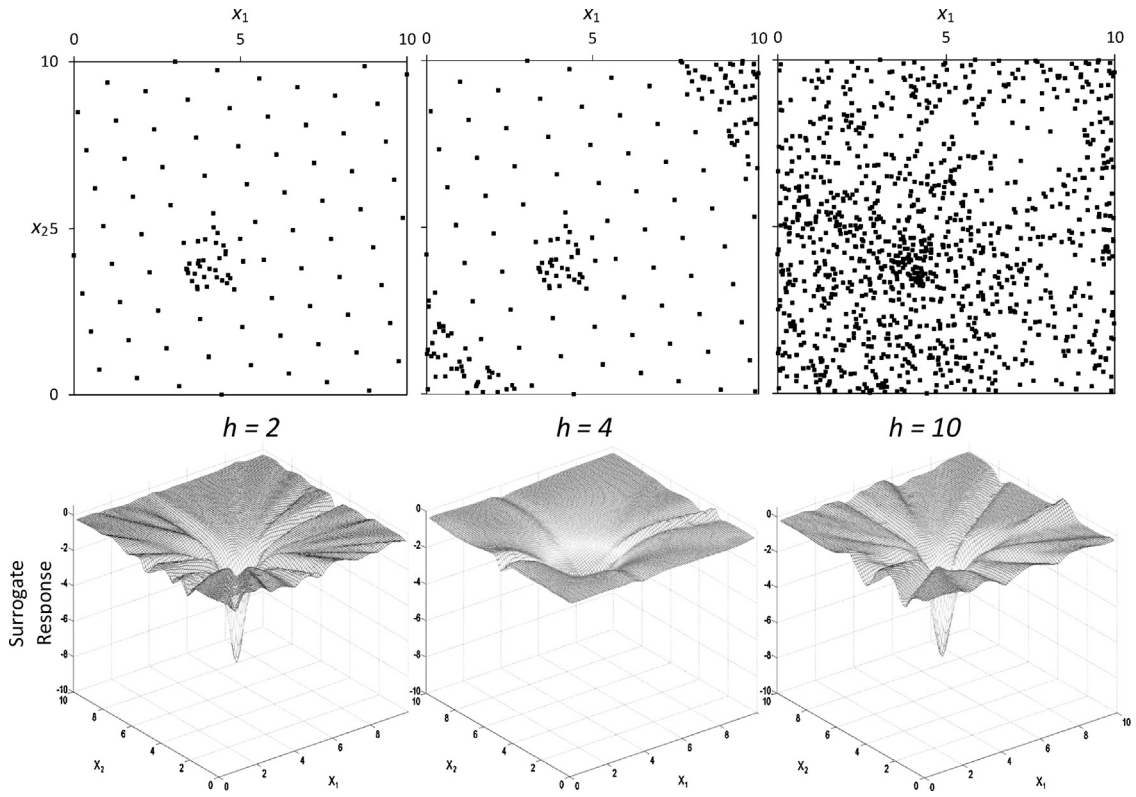

Fig. 8. Sample points (top graphs) and the response of the neural networks trained to these data (bottom graphs) at the second ($h = 2$), the fourth ($h = 4$) and the 10th ($h = 10$) iteration of the pure adaptive sampling algorithm.

response surface of the surrogate model stays essentially same from iteration eight to 10.

The 1130 sample points generated by the pure adaptive sampling algorithm and the response of the surrogate model trained using these sample points for iteration 10 are given in Fig. 8, $h = 10$. The minimum of this surrogate model is −6.70, significantly better than −3.56, the prediction of the surrogate model trained to 1130 sample points generated after 10 iterations of the *iLHS* algorithm. Both surrogate models does not capture the actual minimum of −10.54, however the degree of nonlinearity that we were able to represent using the adaptive sampling algorithm increased due to the sample selection process.

A close examination of the samples at iteration 10 of the adaptive sampling algorithm (Fig. 8, $h = 10$, top graph) reveals that there are still regions that have not been explored beyond the initial sample, for example unit square centered around (8, 8). Fig. 9 is a plot of the density of the samples collected by all three algorithms (*iLHS*, pure adaptive, and mixed adaptive) in a one by 10 unit slice of the sample space as we slide this slice along the *x*-axis. As desired, the sample density is maximum in the region of nonlinearity centered at $X_0 = 4$ for the pure adaptive algorithm. However, around $X_0 = 8$, the density reaches a minimum of 5.5 samples per unit area for the same algorithm. For example, unit squares centered at (8, 8) or (8.32, 4) contain one sample point, and there are no samples in a unit square centered at (6.44, 8). By contrast, the density of the samples generated by the *iLHS* algorithm, by design, maintains a nearly constant value of 11.3 samples per unit area (Fig. 9). This illustrates the disadvantage of a purely adaptive approach: there is no incentive to explore new areas of the input space. As such, it is possible that a spike or dip or a discontinuity in the underlying function may completely be missed by the sampling algorithm.

As can be seen from the densities plotted in Fig. 9, the mixed adaptive sampling algorithm provides a compromise between the pure adaptive and purely space-filling *iLHS* sampling algorithms. At the final iteration, the maximum density is lower than the one of the pure adaptive sampling algorithm, and the overall search space was covered comparable to the coverage obtained by *iLHS* algorithm in terms of sample density. The mixed adaptive sampling algorithm was less aggressive in generating sample points at the nonlinear region of the Shekel function compared to the pure adaptive sampling algorithm at initial iterations. At iteration two, the mixed adaptive sampling algorithm selected 24 points to add within the two by two square centered at the minimum of the Shekel function as opposed to 28 for the pure adaptive algorithm. Because of this slight difference, the surrogate models trained to sample points of iterations two and three does not display the nonlinear behavior of the Shekel function, and starting iteration four, the dip of the Shekel function appears slowly. The surrogate model trained to the sample points at iteration six results in a fairly accurate representation (similar to the one generated at iteration 10 of the pure adaptive algorithm) of the Shekel function. However, at iteration seven, because of the space-filling objective, more samples were added in the linear region of the Shekel function than within nonlinear region, and the dip is lost with the minimum changing from −7.04 at iteration six to −1.51 at iteration seven. Only three points out of 90 were added around the minimum of the Shekel function in this iteration as compared to 31 points added by the pure adaptive sampling algorithm for the same iteration. Iteration eight added four sample points in the nonlinear region, which is enough to improve the dip to a minimum of −6.45. In iteration nine, 12 points were added at the nonlinear region of the Shekel function, and the nonlinearity is observed in the response of the surrogates trained using this data set as well; this behavior of the surrogate predictions is maintained at iteration 10. The minimum of the final neural network trained at iteration 10 is −7.04. This value is comparable to the minimum of the surrogate trained using the samples generated by the pure adaptive sampling algorithm (−6.70) and the actual minimum of −10.54 of the Shekel function. The plots of sample points generated

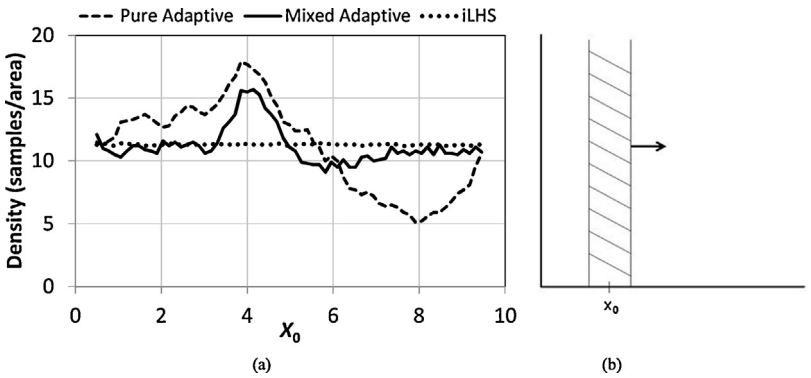

Fig. 9. (a) The density of the samples generated (calculated as samples/unit area centered at X_0 with one unit width) by all algorithms at iteration 10, (b) The illustration that shows how the density of the samples centered at point X_0 is calculated.

by the mixed adaptive sampling algorithm at iteration 10 and the corresponding surrogate response trained using these points are shown in Fig. 7(b) and (d). Note the density at the location of the dip at (4, 4) (Fig. 7(b)), but unlike the pure adaptive sampling algorithm, the rest of the design space has been rather evenly sampled. This behavior is quantified by the density plots in Fig. 9. The mixed adaptive sampling algorithm provides a compromise between the pure adaptive sampling and purely space-filling *iLHS* algorithms. At the final iteration, the maximum density around the minimum is slightly lower than the one observed by the pure adaptive sampling algorithm, however opposed to pure adaptive sampling algorithm, we do not observe any regions with large drops in the density.

4.2. Results for the Ackley function

The Ackley function (Fig. 5(b)) was chosen as a test function for its many local extrema. For a function with many extrema, exploration of the input space is more important say compared to a function that behaves like the Shekel function. For iterations one through four, all sampling algorithms resulted in surrogate responses that were smooth, mostly convex curves with a minimum located near (0, 0). The local minima and maxima of the original Ackley function were not present in any of the surrogate responses for these iterations partly due to the lack of data and partly because the surrogate complexity is penalized in our chosen neural network training algorithm (Bayesian regularization). The fluctuations in the data were treated as noise by the training algorithm until enough data became available, at which point, minimizing the error between the surrogate predictions and the data started to outweigh the penalty term of the more complex, oscillatory behavior of the surrogate response in the objective. The oscillatory behavior of the original Ackley function was first captured by the surrogates trained to sample sets generated by the *iLHS* and mixed adaptive sampling algorithms at iteration six, where there were 360 data points. The pure adaptive sampling algorithm, however, failed to produce a good surrogate model at this sample size. The prediction surfaces of the surrogate models trained using 360 sample points at iteration six generated using all three algorithms are shown in Fig. 10. Because the pure adaptive sampling algorithm leaves many regions of the space unsampled, the local extrema are still ignored by the neural network trained to this data set (Fig. 10) at this sample size. The *iLHS* algorithm has collected enough samples throughout the oscillations, and the neural network trained using this data set begins to exhibit the oscillatory behavior (Fig. 10). The neural network trained using the data generated according to the mixed adaptive sampling algorithm has also begun to represent the extrema, and the adaptive portion of the sampling algorithm generates more samples around the local extrema, which has resulted in more clearly defined minima and maxima in the surrogate response (Fig. 10), fairly closely approximating the actual function behavior (Fig. 5(b)).

4.3. Results for the Peaks function

The Peaks function is the smoothest of the three challenge functions. Based on our results, the impact of sampling method on the quality of fit was negligible. Fig. 11 shows the surrogate models trained using a sample size of 200 points generated after four iterations for each sampling approach. The similarities of these responses and the overall quality of the surrogate models indicate that the sampling procedure becomes less important for simpler functions.

Space-filling designs are important to ensure that the generated samples cover the overall design space, and, hence, the surrogate model trained using the samples considers the quality of fit across the design space. The Ackley function, with many local extrema, is a good example of a system where the space-filling characteristic of the sampling algorithm is important, and where a purely adaptive approach is clearly outperformed by algorithms with space-filling characteristics. Our observations in this section suggest using a mixed adaptive sampling approach, because of its versatility in all three challenge functions, whenever the form of the underlying function is truly unknown, whether the function has many extrema like the Ackley function or few, strongly nonlinear regions like the Shekel function.

5. Case study – CO_2 capture with aqueous amines

In this case study, the sequential sampling methodology with mixed adaptive sampling algorithm is used to develop a surrogate model that predicts the net present cost of CO_2 capture in U.S. dollars per ton of CO_2 recovered using a conventional chemical absorption/desorption process with aqueous amines. The developed surrogate model is then incorporated into a mixed integer nonlinear program (MINLP) to locate the input variables that yields the minimum net present CO_2 capture cost (Eq. (1)). The MINLP was modeled in GAMS version 23.5.2 and solved using BARON version 9.0.6. In Eq. (1), the function $y(\boldsymbol{x})$ represents the surrogate model that estimates the CO_2 capture cost as a function of the input vector $\boldsymbol{x}$.

$$\begin{aligned} &\text{Minimize} \quad y(\boldsymbol{x}) \\ &\text{Subject to} \quad \boldsymbol{X_{lo}} \le \boldsymbol{x} \le \boldsymbol{X_{up}} \end{aligned} \tag{1}$$

The input variables, i.e., the components of vector $\boldsymbol{x}$, for the surrogate are the stripper inlet temperature, the amine concentration, the number of absorber and stripper stages, the reboiler duty, and the solvent circulation rate. The output, y, is the net present cost per ton of CO_2 recovered considering both capital and operating

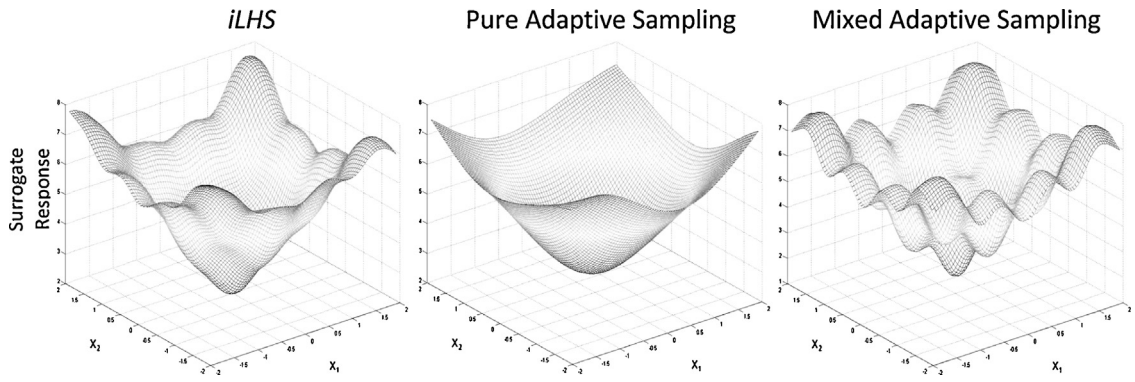

Fig. 10. The response of the neural networks trained at the sixth ($h = 6$) iteration of the *iLHS*, the pure adaptive sampling, and the mixed adaptive sampling algorithms (Ackley function).

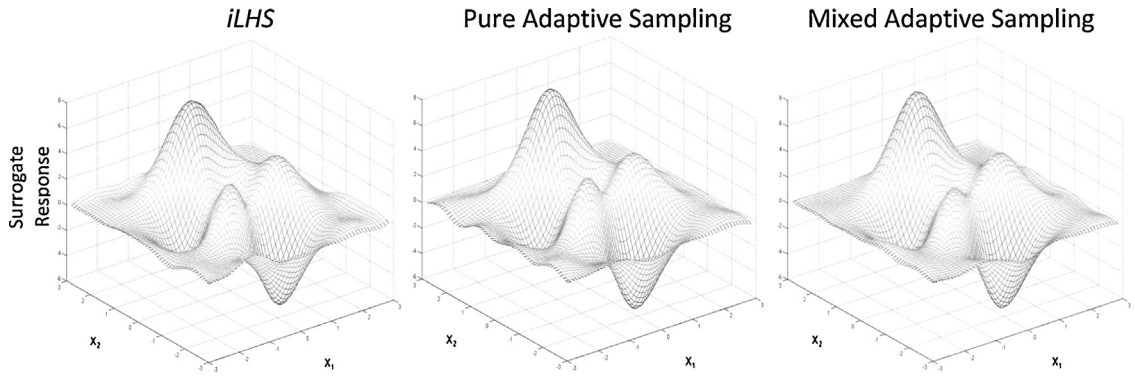

Fig. 11. The response of the neural networks trained at the fourth ($h = 4$) iteration of the *iLHS*, the pure adaptive sampling, and the mixed adaptive sampling algorithms (Peaks function).

costs. The underlying function is the process simulation of the amine-based absorption/desorption process, which processes a gas-turbine flue gas with the following mole percentages: 2.44% CO_2, 7.28% water (H_2O), 17.00% oxygen (O_2), and 73.28% nitrogen (N_2). The simulation was modeled in Aspen HYSYS Version 7.1. The details of the process, the simulation, and the economic analysis can be found in Nuchitprasittichai and Cremaschi (2011).

There are different amine solvents that can be used as the absorber in the absorption/desorption process. Nuchitprasittichai and Cremaschi (2013) observed sharp nonlinearities in the CO_2 capture cost when monoethanolamine (MEA) is used as the solvent opposed to a smooth cost function when diethanolamine (DEA) is the solvent (Nuchitprasittichai and Cremaschi, 2011). These two solvents, MAE and DAE, were selected for this case study. The bounds of the input variables ($\boldsymbol{X_{lo}}$ and $\boldsymbol{X_{up}}$), i.e., the search space for the surrogate and the optimization, are given in Table 1. Artificial neural networks with 8 neurons was used as the surrogate model for both solvents, and the initial samples set was a 60-point pre-optimized Latin hypercube design obtained from (Dam et al., 2009) for adaptive sampling algorithms and a random 60-point Latin hypercube design for the *iLHS* algorithm. The termination criterion for the sequential sampling algorithm was the slope ratio percentage defined using *SSCVE* as follows:

$$\text{Slope ratio percentage for iteration } i = \left| \frac{SSCVE \text{ slope from interation } i-1 \text{ to } i}{\text{maximum } SSCVE \text{ slove for interation } j \le i} \right| \quad (2)$$

The slope ratio percentage compares the relative change in *SSCVE* per added samples at each iteration against the largest such change for all preceding iterations of the sequential sampling algorithm. Therefore, this percentage gives a measure of the improvement in the surrogate model predictions during the course of the algorithm. The algorithm is terminated when the percentage falls below a certain target value, which is set to 3% for these case studies. The final surrogate model is used in the optimization formulation given in Eq. (1). For this section, we will refer to the objective function value at the "solution" of the MINLP using the surrogate model as the predicted minimum. The actual output, i.e., the one that is calculated by the process simulator, is obtained by running the process simulator at the MINLP "solution". The following two sections evaluate the accuracy of the predicted minimum and the sample size when the samples are collected with the *iLHS* and mixed adaptive algorithm.

Table 1
The upper and lower bounds of the operating conditions and design variables. In the first row; 1 represents the solvent circulation rate, 2 the solvent concentration, 3 the number of absorber stages, 4 the number of stripper stages, 5 the reboiler duty, and 6 the regenerator inlet-temperature.

	1 (m^3/h)	2 (w%)	3 (stages)	4 (stages)	5 (kW)	6 (°C)
MEA						
$\boldsymbol{X_{up}}$	12	24	62	24	5338	105
$\boldsymbol{X_{lo}}$	7	8	43	10	3248	81
DEA						
$\boldsymbol{X_{up}}$	12	32	39	39	2854	105
$\boldsymbol{X_{lo}}$	8	8	28	28	1817	94

5.1. Results for the MEA solvent

The results obtained using the mixed adaptive sampling algorithm and the *iLHS* for training accurate ANNs for the MAE solvent are summarized in Fig. 12(a) and (b), respectively. The figures present the change in *SSCVE* and the progress of slope ratio

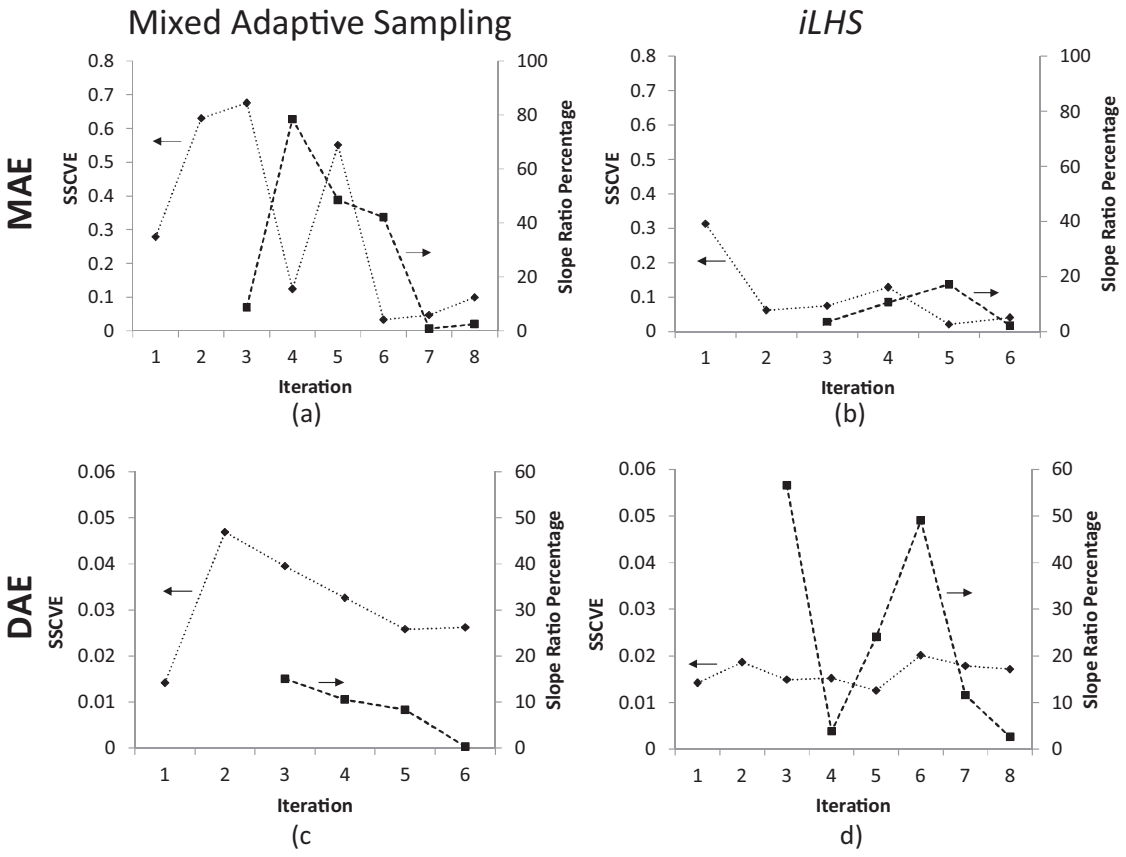

Fig. 12. *SSCVE* and slope ratio percentage vs. sequential sampling algorithm iteration using (a) mixed adaptive sampling algorithm for MAE, (b) *iLHS* algorithm for MAE, (c) mixed adaptive algorithm for DAE, and (d) *iLHS* algorithm for DAE.

percentage with the number of iterations. Table 2 gives the sample sizes and the total number of underlying function evaluations (i.e., total simulation runs) to obtain that sample size, the "optimum" design and operating conditions and the corresponding CO_2 capture cost calculated when the trained ANNs at the termination of the sequential sampling approach are used in the MINLP defined in Eq. (1).

The sequential sampling algorithm converged after seven iterations using mixed adaptive sampling approach, and the slope ratio percentage after this iteration dropped to 0.8% satisfying the termination criteria. At the end of seven iterations, the CO_2 capture process simulation was executed for 360 times, resulting in 360 input–vector output pairs. The comparison of the predicted ($48.49 per ton of CO_2 removed) and the actual ($61.48 per ton of CO_2 removed) CO_2 capture costs reveals a large discrepancy. Speculating that the threshold of 3% as the slope percent ratio may not be small enough for capturing the underlying behavior of the complex cost function for the MAE solvent, we run the sequential sampling algorithm for an additional iteration, the eighth iteration. As can be seen from Fig. 12(a), there is very little difference between the slope ratio percentages of the last two iterations, confirming that there was little change in the *SSCVE* with the addition of new samples. Although the discrepancy between the predicted ($48.44 per ton) and actual ($52.33 per ton) CO_2 capture costs reduced after the eight iteration, the difference is still significant (7.5%). When the MINLP solutions of the two ANNs (the one trained using 360 points and the one trained using 480 points) are compared (Table 2), it is observed that they only differ in two out of six variables, the number of absorber stages (23 and 24) and the reboiler duty (3275 kW and 3405 kW). The resulting cost behavior suggests a sharp nonlinearity in this underlying function, and both surrogate models fail to capture this behavior.

Table 2
Operating conditions and design variables for the case studies. In the first row; 1 represents the solvent circulation rate, 2 the solvent concentration, 3 the number of absorber stages, 4 the number of stripper stages, 5 the reboiler duty, and 6 the regenerator inlet-temperature.

Solvent	Sampling approach	Sample size	Total simulation runs	1 (m^3/h)	2 (wt%)	3 (stages)	4 (stages)	5 (kW)	6 (°C)	CO_2 capture costs ($/ton CO_2 recovered)	
										ANN	HYSYS
MEA	Mixed	360	360	62.0	11.7	23	12	3275	81	48.49	61.48
	Adaptive	480	480	62.0	11.7	23	12	3405	81	48.44	52.33
	iLHS	270	475	62.0	12.3	22	10	3249	81	48.45	57.31
		360	646	62.0	13.0	21	12	3252	81	48.54	56.16
	Mixed adaptive + 40 BC	400	400	62.0	11.5	24	12	3250	89	47.40	48.08
DAE	Mixed adaptive	270	270	30.2	39.0	32	12	2010	105	46.46	46.63
	iLHS	480	891	32.5	39.0	32	8	2350	94	46.21	46.66

The algorithm using *iLHS* terminates after the sixth iteration at a sample size of 270, which required 475 underlying function evaluations, i.e., simulation runs generating a total of 475 input–vector output pairs. In order to keep the Latin Hypercube design for each sequential sample set, the *iLHS* algorithm is not able to utilize all the existing samples, and hence, requires more function evaluations compared to mixed adaptive sampling algorithm to generate same size sample sets. However, it should be noted that by reusing as many samples as possible from previous iterations, the *iLHS* algorithm reduced the total number of function evaluations from 870 (60 + 80 + 110 + 150 + 200 + 270) to 475 while maintaining the Latin Hypercube design of the samples at each iteration. The slope ratio percentage at the end of the sixth iteration is 2.2%. The solution of the MINLP (Eq. (1)) using the ANN trained at the sixth iteration yields an objective function value of \$48.45 per ton of CO_2 recovered (Table 2). When a HYSYS simulation is built and run using the solution of this MINLP, the actual cost of the process is calculated as \$57.31 per ton of CO_2 recovered (Table 2), revealing a 15.5% difference between the predicted and actual costs of the process. We also extended the sequential sampling algorithm for an additional iteration, seventh iteration with a total of 360 sample points with the Latin Hypercube design requiring an additional 171 additional HYSYS simulation runs. The slope ratio percentage dropped to 0.24% at the end of this iteration, the predicted CO_2 removal cost as the solution of the MINLP using the resulting ANN was \$48.54 per ton of CO_2, and the corresponding actual cost for the MINLP solution was \$56.15 per ton of CO_2 removed, a 13.6% difference. The solutions of both MINLPs are similar to the ones obtained when the ANNs trained using the data collected by the mixed adaptive sampling approach was used (Table 2).

Based on the above results, we hypothesize that the underlying function that links the cost of CO_2 removal to our input vector for solvent MEA is difficult to fit with a surrogate, and that the region near the minimum is relatively flat with a sharp nonlinearity near the predicted minimum. Capturing this spike in the predictions of the surrogate model requires that the sampling algorithm selects a point in this region. The mixed adaptive sampling algorithm selected several points in the relatively flat region around the nonlinearity and failed to generate a sample within the area itself. The space-filling objective of the mixed adaptive sampling algorithm would eventually encourage further sampling in the region, but the nonlinearity was not sampled in the first eight iterations. In order to test our hypothesis, a 40 points bounds contraction (BC) approach (Caballero and Grossmann, 2008) was used. In this method, a 40 point Latin hypercube sample is generated on a range of ±5% of each input variable centered at the predicted minimum obtained using the ANN trained at the seventh iteration of the sequential sampling algorithm. These 40 points are evaluated and added to the existing 360 sample points. The solution of the MINLP that uses the ANN trained using these 400 sample points are given in Table 2 row Mixed Adaptive + 40 BC. As can be seen from Table 2, the predicted CO_2 removal cost (\$47.70 per ton CO_2 recovered) is within 1.5% of the actual cost (\$48.08 per ton CO_2 recovered). If we examine the solution, we realize that the main difference between the optimum and the solutions suggested by the ANN trained using the mixed adaptive algorithm only differ significantly at the number of absorber stages (optimum: 24 stages, previous: 23 stages) and the regenerator inlet temperature (optimum: 89 °C, previous: 81 °C). At a previous study from our group, we demonstrated that CO_2 capture cost is very sensitive to the number of absorber stages, and the regenerator inlet temperature had insignificant impact on the cost (Nuchitprasittichai and Cremaschi, 2011). For the MAE solvent, the optimum number of absorber stages is at the upper bound of the range, and both sampling algorithms failed to generate enough number of samples at the bounds of the search space, which is a common problem and an open challenge for most sampling approaches. The sample points added during bounds contraction focused on the upper bound of the number of stages, and hence, improved the prediction performance of ANN in that region considerably. The future work for our sampling algorithms will focus on addressing this weakness.

5.2. Results for the DEA solvent

The optimization surface for CO_2 capture cost with DEA as solvent is smoother than that of MEA solvent (Nuchitprasittichai and Cremaschi, 2011). Therefore, we expect that fitting a surrogate model to replicate the behavior of this function would be easier compared to the MEA solvent case. For the DEA solvent case, both mixed adaptive sampling and *iLHS* algorithms yielded accurate surrogate models. Fig. 12(c) and (d) displays the progression of *SSCVE* and the slope ratio percentage with the number of sequential sampling algorithm iteration counter using mixed adaptive sampling and *iLHS* approaches, respectively. The smoothness of both SSCVE and the slope ratio percentage lines using the mixed adaptive sampling approach for DEA solvent (Fig. 12(c)) as samples are added compared to MEA case (Fig. 12(a)) suggests a smoother underlying function. Sharp increases in *SSCVE* are generally associated with locating a region that has unexpected or nonlinear behavior. At termination, predicted minima using both ANNs trained to the data generated using mixed adaptive sampling and *iLHS* agreed with the actual cost obtained using the process simulator within <1% (Table 2).

Although both sampling approaches (mixed adaptive sampling and *iLHS*) yielded samples that resulted in accurate ANNs, the number of samples, and hence, the underlying function evaluations, generated by the mixed adaptive sampling algorithm was considerably smaller. The mixed adaptive sampling algorithm terminated at iteration six with a sample size of 270, while the *iLHS* terminated at iteration eight with a sample size of 480 (Table 2). This is a 44% reduction in the required sample size, which saves significant time during the surrogate model construction process. This difference becomes more pronounced when the actual number of function evaluations required by each approach is considered. By the time the *iLHS* terminates at the eighth iteration, it had performed 891 underlying function evaluations, which is over three times the number of underlying function evaluations required to construct the surrogate using the mixed adaptive sampling algorithm.

6. Conclusions and future directions

In this article, several important aspects of sample selection for constructing artificial-neural-network surrogate models are discussed. Most previous work with surrogate models has used a one-shot sample generation and surrogate training approach, selecting an experimental design and guesstimating a suitable sample size. However, when the underlying function is computationally expensive, the surrogate construction process could be expedited by using a sequential sampling approach. We introduced two different sample generation algorithms, pure and mixed adaptive sampling algorithms that can be used in sequential sampling approaches for surrogate model construction. Using the Shekel function as a challenge function, it was shown that an adaptive sampling approach is required in order to construct accurate surrogate models when strong nonlinearities are present in the underlying function. However, the Ackley function demonstrated a case in which a space-filling approach was more efficient (i.e. reduced the required sample size). Our mixed adaptive sampling algorithm provides a compromise between purely adaptive and purely space-filling sampling approaches. The CO_2 case study with DEA solvent demonstrated that mixed adaptive sampling algorithm may greatly reduce the required sample size. However, case study with MAE as the solvent revealed that both space-filling and

adaptive algorithms may fail to capture the behavior of the underlying function if there are strong nonlinearities at the bounds of the search space. If the main objective of surrogate construction is optimization, one approach to alleviate this weakness may be to incorporate optimization to the sequential sampling algorithm. Here, the surrogate may be used to identify the "best" performance regions with an optimization approach, and the sampling algorithm may exploit these regions with additional samples. However, this approach may add considerable computational cost to sample selection algorithm. The future work will focus on addressing this weakness of the mixed adaptive sampling algorithm in addition to adjusting weights of the normalized space-filling/adaptive objectives.

Acknowledgement

The financial support from The University of Tulsa is gratefully acknowledged.

Appendix A. Supplementary data

Supplementary data associated with this article can be found, in the online version, at http://dx.doi.org/10.1016/j.compchemeng.2014.05.021.

References

Baş D, Boyacı İH. Modeling and optimization I: usability of response surface methodology. J Food Eng 2007;78:836–45.
Basheer I, Hajmeer M. Artificial neural networks: fundamentals, computing, design, and application. J Microbiol Methods 2000;43:3–31.
Caballero JA, Grossmann IE. An algorithm for the use of surrogate models in modular flowsheet optimization. AIChE J 2008;54:2633–50.
Chi H, Mascagni M, Warnock T. On the optimal Halton sequence. Math Comput Simul 2005;70:9–21.
Crombecq K. Surrogate modeling of computer experiments with sequential experimental design; 2011.
Crombecq K, De Tommasi L, Gorissen D, Dhaene T. A novel sequential design strategy for global surrogate modeling. In: Proceedings of the 2009 winter simulation conference (WSC). IEEE; 2009. p. 731–42.
Crombecq K, Laermans E, Dhaene T. Efficient space-filling and non-collapsing sequential design strategies for simulation-based modeling. Eur J Oper Res 2011;214:683–96.
Dam EV, Hertog DD, Husslage B, Rennen G. Space-filling designs. the Netherlands: Tilburg University; 2009 http://www.spacefillingdesigns.nl (accessed on April 2013).
Davis E, Ierapetritou M. A kriging based method for the solution of mixed-integer nonlinear programs containing black-box functions. J Glob Optim 2009;43:191–205.
Davis E, Ierapetritou M. A centroid-based sampling strategy for kriging global modeling and optimization. AIChE J 2010;56:220–40.
Devabhaktuni VK, Zhang QJ. Neural network training-driven adaptive sampling algorithm for microwave modeling. In: 30th European microwave conference, 2000. IEEE; 2000. p. 1–4.
Fahmi I, Cremaschi S. Process synthesis of biodiesel production plant using artificial neural networks as the surrogate models. Comput Chem Eng 2012;46:105–23.
Fernandes FAN. Optimization of Fischer–Tropsch synthesis using neural networks. Chem Eng Technol 2006;29:449–53.
Forrester AIJ, Keane AJ. Recent advances in surrogate-based optimization. Progr Aerospace Sci 2009;45:50–79.
Ghosh BK, Sen PK. Handbook of sequential analysis. New York, NY: Marcel Dekker; 1991.
Giunta AA, Wojtkiewicz SF, Eldred MS. Overview of modern design of experiments methods for computational simulations. In: Proceedings of the 41st AIAA aerospace sciences meeting and exhibit, AIAA-2003-0649; 2003.
Gorissen D, Crombecq K, Hendrickx W, Dhaene T. Adaptive distributed metamodeling. In: High performance computing for computational science-VECPAR 2006; 2007. p. 579–88.
Henao CA, Maravelias CT. Surrogate-based process synthesis. Comp Aided Chem Eng 2010;28:1129–34.
Henao CA, Maravelias CT. Surrogate-based superstructure optimization framework. AIChE J 2011;57:1216–32.
Jin R, Chen W, Simpson TW. Comparative studies of metamodelling techniques under multiple modelling criteria. Struct Multidiscip Optim 2001;23:1–13.
Jones D, Schonlau M, Welch W. Efficient global optimization of expensive black-box functions. J Glob Optim 1998;13:455–92.
Kleijnen JPC. Kriging metamodeling in simulation: a review. Eur J Oper Res 2009;192:707–16.
Kleijnen JPC, Van Beers WCM. Application-driven sequential designs for simulation experiments: kriging metamodelling. J Oper Res Soc 2004;55:876–83.
Lang Y-D, Malacina A, Biegler LT, Munteanu S, Madsen JI, Zitney SE. Reduced order model based on principal component analysis for process simulation and optimization. Energy Fuels 2009;23:1695–706.
Lang Y, Zitney SE, Biegler LT. Optimization of IGCC processes with reduced order CFD models. Comput Chem Eng 2011;35:1705–17.
Nascimento CAO, Giudici R, Guardani R. Neural network based approach for optimization of industrial chemical processes. Comput Chem Eng 2000;24:2303–14.
Nuchitprasittichai A, Cremaschi S. Optimization of CO_2 capture process with aqueous amines using response surface methodology. Comput Chem Eng 2011;35:1521–31.
Nuchitprasittichai A, Cremaschi S. An algorithm to determine sample sizes for optimization with artificial neural networks. AIChE J 2013;59(3):805–12.
Peng K, Obradovic Z, Vucetic S. Towards efficient learning of neural network ensembles from arbitrarily large datasets. ECAI, vol. 16; 2004. p. 623.
Provost F, Jensen D, Oates T. Efficient progressive sampling. In: Proceedings of the fifth ACM SIGKDD international conference on knowledge discovery and data mining. San Diego, CA, USA: ACM; 1999. p. 23–32.
Queipo NV, Haftka RT, Shyy W, Goel T, Vaidyanathan R, Kevin Tucker P. Surrogate-based analysis and optimization. Progr Aerospace Sci 2005;41:1–28.
Simpson TW, Lin DKJ, Chen W. Sampling strategies for computer experiments: design and analysis. Int J Reliab Appl 2001;2:209–40.
Smith J, Cremaschi S, Crunkleton D. CFD-based optimization of a flooded bed bioreactor for Algae production. In: Iftekhar AK, Rajagopalan S, editors. Computer aided chemical engineering, vol. 31. Oxford, UK: Elsevier; 2012. p. 910–4.
Smith J, Neto A, Cremaschi S, Crunkleton D. CFD-based optimization of a flooded bed algae bioreactor. Ind Eng Chem Res 2013;52(22):7181–8.
Wan X, Pekny JF, Reklaitis GV. Simulation-based optimization with surrogate models—application to supply chain management. Comput Chem Eng 2005;29:1317–28.
Wolter K. Introduction to variance estimation. New York, NY: Springer; 2007.
Yuste AJ, Dorado MP. A neural network approach to simulate biodiesel production from waste olive oil. Energy Fuels 2005;20:399–402.