HIERARCHICAL NONLINEAR APPROXIMATION FOR EXPERIMENTAL DESIGN AND STATISTICAL DATA FITTING

DANIEL BUSBY*, CHRIS L. FARMER†, AND ARMIN ISKE‡

Abstract. This paper proposes a hierarchical nonlinear approximation scheme for scalar-valued multivariate functions, where the main objective is to obtain an accurate approximation with using only very few function evaluations. To this end, our iterative method combines at any refinement step the selection of suitable evaluation points with kriging, a standard method for statistical data analysis. Particular improvements over previous non-hierarchical methods are mainly concerning the construction of new evaluation points at run time. In this construction process, referred to as experimental design, a flexible two-stage method is employed, where adaptive domain refinement is combined with sequential experimental design. The hierarchical method is applied to statistical data analysis, where the data is generated by a very complex and computationally expensive computer model, called a simulator. In this application, a fast and accurate statistical approximation, called an emulator, is required as a cheap surrogate of the expensive simulator. The construction of the emulator relies on computer experiments using a very small set of carefully selected input configurations for the simulator runs. The hierarchical method proposed in this paper is, for various analysed models from reservoir forecasting, more efficient than existing standard methods. This is supported by numerical results, which show that our hierarchical method is, at comparable computational costs, up to ten times more accurate than traditional non-hierarchical methods, as utilized in commercial software relying on the response surface methodology (RSM).

Key words. nonlinear approximation, hierarchical approximation, experimental design, statistical data fitting, computer experiments, data-adaptive modelling, kriging, reservoir forecasting

AMS subject classifications. 62K20, 62P30, 65D15

1. Introduction. Approximation of scalar-valued multivariate functions from scattered data is of primary importance in many applications from computational science and engineering. To describe the mathematical problem, let $f : \Omega \to \mathbb{R}$, for some domain $\Omega \subset \mathbb{R}^d$, denote an unknown multivariate function, whose scalar function values in $f\big|_X = (f(\mathbf{x}_1), \ldots, f(\mathbf{x}_n))^T \in \mathbb{R}^n$, taken at a scattered set $X = \{\mathbf{x}_1, \ldots, \mathbf{x}_n\} \subset \Omega$ of pairwise distinct points, are given. Scattered data approximation requires the construction of a suitable function $s : \mathbb{R}^d \to \mathbb{R}$, which is *close* to f in a sense to be described. Usually, the quality of the approximation s is, for a fixed normed linear space $\mathcal{V}$ containing f and s, i.e., $f, s \in \mathcal{V}$, measured by the approximation error

$$\eta_{\mathcal{V}}(f, s) = \|f - s\|_{\mathcal{V}}.$$

The construction of an approximation to f is usually accomplished by first specifying a suitable finite-dimensional approximation space $\mathcal{S} \subset \mathcal{V}$, before a *best approximation* $s^* \in \mathcal{S}$ to f is computed by minimizing the distance between f and $\mathcal{S}$, so that s^* satisfies

$$\min_{s \in \mathcal{S}} \|f - s\|_{\mathcal{V}} = \|f - s^*\|_{\mathcal{V}}.$$

In the above mentioned approximation problem, the beforehand construction of a *linear* approximation space $\mathcal{S}$ usually depends on X, and so the resulting approxi-

* IFP (Institut Français du Pétrole), Rueil-Malmaison, 92500, France (Daniel.BUSBY@ifp.fr).

† Schlumberger Abingdon Technology Center, Abingdon OX14 1UJ, UK, and Oxford Centre for Industrial and Applied Mathematics, University of Oxford, Oxford OX1 3LB, UK (farmer5@slb.com).

‡Department of Mathematics, University of Hamburg, Bundesstr. 55, D-20146 Hamburg, Germany (iske@math.uni-hamburg.de).

mation scheme is *linear*. In such a setting, the sample points in X and the function values $f\big|_X$ are required to be *given*.

We remark that there are only a few powerful approximation methods available for solving such linear scattered data approximation problems for multivariate data, where radial basis function methods are among the most popular tools. For a comprehensive treatment of scattered data approximation and radial basis function methods, we refer to the textbooks [1, 11, 31], where in particular relevant aspects concerning numerical stability, approximation orders for suitable (native) functions spaces, $\mathcal{V}$, and the construction of best approximations are explained in large detail.

In this paper we wish to propose a *nonlinear* approximation scheme for *statistical* data fitting, where the nonlinearity is due to the variation of the sample points in X. The goal of our resulting computational method is to obtain a sufficiently accurate approximation to f by using as few as possible sample points in X, i.e., n is required to be very small. Moreover, we assume that the dimension of the input variables is very large, i.e., $d \gg 1$, and the input data are assumed to be *uncertain*. The latter essentially requires suitable methods for *statistical* data fitting, i.e., statistical approximation schemes which are relying on suitable statistical models.

To this end, we prefer to work with *kriging*, a well-established statistical method first introduced by Krige in the 1950's and then formalized as a fundamental geostatistical technique by Matheron [17] in 1963, and later utilized by Sacks, Welch, Mitchell, and Wynn [25] in computer experiments. We remark that kriging is under specific conditions equivalent to Bayesian methods (see [6] for details), and also to radial basis function interpolation (as recently shown in [9, 32]).

By combining kriging with adaptive domain decomposition, the proposed method of this paper results in a hierarchical nonlinear approximation scheme, where the nonlinearity is due to the adaptive selection of suitable sample points X at the algorithm's run time.

The motivation for this work is mainly given by specific applications from experimental design and statistical data fitting in computer experiments. In the general setting of this wide application field, many scientific phenomena are investigated by using complex computer models or codes. In oil reservoir forecasting, for instance, *simulators* are used to model the evolution of time-dependent physical quantities in a hydrocarbon reservoir production scenario, where long time scales are common, see [12].

The simulation codes are often computationally expensive to run, which requires cheaper predictors, *emulators*, as surrogates for the too expensive simulators. The construction of an emulator relies on a sequence of computer experiments, involving a number of costly simulator runs with various different input configurations, where the overall aim is to obtain a sufficiently accurate emulator at an as small as possible number of simulator runs. This requires a careful selection of input configurations for the simulator runs.

In such and similar application scenarios for computer experiments, the simulator output is *deterministic*, i.e., rerunning the code with the same inputs gives identical observations, but major computational difficulties are often due to *high dimensional* and *uncertain* input data. This requires customized methods for the construction of a sufficiently accurate emulator, in particular as experimenters are interested in *uncertainty evaluation* of the simulation output, *fine-tuning* of the simulator by using external data from the physical system, such as in *history matching*, and the optimization of adjustable input parameters.

In the mathematical formulation of this problem, the simulator is viewed as a multivariate target function $f : \mathbb{R}^d \to \mathbb{R}$, the emulator is an approximation $s : \mathbb{R}^d \to \mathbb{R}$ to f, and the selected input configurations, often referred to as *design sites* are the sample points in X.

In general, the construction of an efficient emulator involves the following steps.

1. *Experimental design*, where the goal is to determine configurations of input parameters for running the simulator to build a sufficiently accurate emulator with an as small as possible number of simulator runs.
2. *Screening*, where the aim is to reduce the dimensionality of the problem under investigation. This is usually done in a preprocessing step by performing a sensitivity analysis with the available data, in order to discard irrelevant parameters which were initially included in the model. Therefore, screening can also be viewed as an *a priori* model selection problem, where the *active* and *inactive* input parameters are determined.
3. *Statistical data analysis*, where model parameters are tuned for the selection of an (optimal) emulator which is used to make accurate predictions on the response at untried inputs. This is usually done by optimizing a functional of the response. A common approach for analyzing statistical data is *kriging*, as explained in Section 3.

The proposed hierarchical approximation scheme aims at the efficient construction of an accurate emulator. This is accomplished by an iterative refinement strategy where the available data collected so far is used to apply at each iteration step an adaptive two-stage process for the experimental design, followed by a statistical data analysis to optimize the parameters of the emulator. Following standard concepts [19] concerning screening, we refrained from including screening in this hierarchical refinement process, although this may be an option in situations, where the dimension d of the input data is extremely high, so that further eliminations of input variables are inevitably needed to reduce the required computational costs. A formal description of the proposed hierarchical approximation scheme is given by Algorithm 1.

Our nonlinear approximation scheme results in a multiresolution representation of the emulator, from coarse to fine. That is, with increasing accuracy at each iteration, to obtain a sufficiently accurate emulator at the finest level after merely a few iterations. In this way, the utilized refinement strategy allows control of the gradually increasing computational costs for building the hierarchical emulator sequence, and their increasing accuracy.

We remark that the proposed approach in this paper uses ideas from previous work [5, 6, 8, 14, 25, 27, 28, 30] concerning experimental (sequential) design, sensitivity analysis and interpolation schemes in non-hierarchical methods.

Our approach aims at two important objectives in experimental design, that is (1) reducing the uncertainty predicted by the model and (2) improving the model reliability. The first objective is addressed by entropy minimization, in combination with a customized adaptive gridding algorithm. The second objective is achieved by adaptive cross validation, with acquiring further observations from input configurations where the model prediction is bad (*local optimal design*). Although both kinds of adaptivity look similar at first sight, these two approaches are fundamentally different from a mathematical and conceptual point of view.

The outline of the paper is as follows. In Section 2, the basic ideas of the hierarchical nonlinear approximation scheme are explained, before the kriging method is discussed in Section 3. In Section 4, the construction of an initial design is explained

and some selected aspects concerning screening are briefly addressed. Sections 5 and 6 are devoted to details concerning adaptivity in experimental design, which works with a two-stage process. In Section 5, we explain the global approach of adaptive gridding, before turning to local optimal design in Section 6. A schematic description of the final algorithm is given in Section 7. Finally, Section 8 presents numerical results obtained by applying our method to a given function, being regarded as a simulator, and to two different reservoir forecasting problems. It is shown that our hierarchical scheme is for the analyzed reservoir models more efficient than existing methodologies. Indeed, our approximation method is, at comparable computational costs, up to ten times more accurate than the quadratic method in reservoir uncertainty analysis software, such as COUGAR [4], which relies on the response surface methodology (RSM).

2. Hierarchical Nonlinear Approximation. The proposed hierarchical approximation scheme constructs a sequence $s_1, s_2, \ldots, s_L$ of approximations to an unknown function $f : \mathbb{R}^d \to \mathbb{R}$ from samples of f taken at scattered locations $X = \{x_1, \ldots, x_m\}$. The construction of the approximation relies on a data hierarchy

$$X_1 \subset X_2 \subset \cdots \subset X_L \subset X \tag{2.1}$$

of nested subsets of X, where L denotes the number of levels. Then, the functions s_ℓ approximate f at the subsets X_ℓ of the level ℓ, $1 \leq \ell \leq L$, according to some specific approximation scheme. Examples for recent approximation schemes are discussed in [11]. Note that the sequence $s_1, s_2, \ldots, s_L$ of approximations to f is from coarse to fine. Indeed, the coarsest approximation s_1 is computed on the basis of the initial design X_1, which includes only very few data. In contrast, the approximations s_ℓ at finer levels ℓ contain gradually more information from their corresponding designs X_ℓ.

The approach taken in this paper is *adaptive*, that is the construction of the data hierarchy in (2.1) is done at run time in the approximation process, rather than in a preprocessing step. This adaptive approach, dating back to [13], allows control of both the cumulative computational cost and the accuracy of the current approximation s_ℓ at any level ℓ.

Recall that in the setting and language of this paper, the sets X_ℓ in (2.1) are constructed, one after the other, by an adaptive experimental design strategy. Moreover, any approximation s_ℓ is viewed as an emulator of the simulator f. Each emulator is computed by statistical data fitting at level ℓ, $1 \leq \ell \leq L$.

The following algorithm schema outlines our hierarchical approximation method, described in detail in Sections 3-6.

ALGORITHM 1. **(Hierarchical Nonlinear Approximation)**

INPUT: *Simulator* $f : \mathbb{R}^d \to \mathbb{R}$.

- **(1)** *Construct the sites* X_1 *by initial design and run the simulator* f *at* X_1 *to obtain the initial data* $\{f(\mathbf{x}) : \mathbf{x} \in X_1\}$.
- **(2)** *Determine the active and inactive input parameters by a screening preprocess. Discard the inactive input parameters.*
- **(3)** *Build the initial emulator* s_1 *by using kriging.*
- **(4)** **FOR** $\ell = 2, 3, \ldots$ **DO**
  - **(4a)** *Construct the sites* X_ℓ *in (2.1) by a two-stage experimental design: apply global adaptive gridding, followed by local optimal design.*
  - **(4b)** *Run the simulator* f *at the new design sites* $X_\ell \setminus X_{\ell-1}$ *to obtain new responses* $\{f(\mathbf{x}) : \mathbf{x} \in X_\ell \setminus X_{\ell-1}\}$.

(4c) *Perform a statistical analysis of the available data* $\{f(\mathbf{x}) : \mathbf{x} \in X_\ell\}$ *by using kriging, and so compute the next emulator* s_ℓ *of level* ℓ.

OUTPUT: *Sequence* $s_1, s_2, \ldots, s_L$ *of emulators with increasing accuracy.*

Note that the iterative approximation scheme performs a sequential refinement of the (emulator) model. In the following sections, we explain steps **(1)**-**(3)**, **(4a)** and **(4c)** of Algorithm 1 in detail. The construction of the initial design X_1 in step **(1)** is done by using *Latin hypercube design* (LHC), as discussed in Section 4, where also screening, as required in step **(2)**, is briefly addressed. The two-stage experimental design of step **(4a)** is explained in Sections 5 and 6. The construction of the emulators s_ℓ in steps **(3)** and **(4c)** relies on kriging, a standard statistical approach, which is for the reader's convenience reviewed in the following section.

3. Kriging. The approximation process at each level of Algorithm 1 can be divided into two main steps: In the first step, a statistical model $S(\mathbf{x})$ for the given data (obtained from the simulator) is derived. In the second step, a deterministic predictor for the statistical model – the emulator – is computed. Therefore, the emulator can be viewed as a statistical approximation, and so the proposed method, Algorithm 1, provides a suitable tool for statistical data fitting.

The construction of the individual emulators s_ℓ, $\ell = 1, \ldots, L$, in Algorithm 1 uses *kriging*. Kriging, due to Matheron [17], is a statistical method for scattered data interpolation which can be used for prediction in computer experiments [25, 30]. To explain kriging, denote, at any fixed level $1 \le \ell \le L$, the emulator by $s \equiv s_\ell$ and let $X \equiv X_\ell$ be the corresponding design at level ℓ.

In the underlying statistical model of kriging, the deterministic response, say $f(\mathbf{x}) : \mathbb{R}^d \to \mathbb{R}$, is considered as a realization of a random function. Therefore, the simulator f is treated as a realization of a stochastic process, whose form is assumed as

$$S(\mathbf{x}) = \sum_{j=1}^{k} \beta_j h_j(\mathbf{x}) + Z(\mathbf{x}). \tag{3.1}$$

The regression part in this stochastic model (3.1) is a linear combination of preselected real-valued functions $h_1, \ldots, h_k$, with coefficients $\beta = (\beta_1, \ldots, \beta_k)^T \in \mathbb{R}^k$, where h_1 is a constant. Moreover, Z in (3.1) is assumed to be a Gaussian random process with mean zero and covariance

$$\operatorname{cov}[\mathbf{x}, \mathbf{y}] = \mathrm{E}[Z(\mathbf{x})Z(\mathbf{y})] = \sigma^2 R(\mathbf{x}, \mathbf{y}) \tag{3.2}$$

between $Z(\mathbf{x})$ and $Z(\mathbf{y})$, where σ^2 denotes the process variance, $R(\mathbf{x}, \mathbf{y})$ is a specific correlation function, and the symbol E denotes the usual statistical expectation. The choice of σ and R is discussed later in this section.

In a Bayesian approach, the random process $S(\mathbf{x})$ is regarded as a *prior* model and the response values $\{f(\mathbf{x}) : \mathbf{x} \in X\}$ are viewed as observations. It remains to determine a suitable *posterior* predictor – the emulator – from the given set of observations.

In kriging, for given design $X = \{\mathbf{x}_1, \ldots, \mathbf{x}_m\} \subset \mathbb{R}^d$ and for given responses $\mathbf{f}_X = (f(\mathbf{x}_1), \ldots, f(\mathbf{x}_m))^T \in \mathbb{R}^m$, a linear predictor

$$\hat{S}(\mathbf{x}) = \mathbf{a}^T(\mathbf{x})\mathbf{f}_X$$

of $f(\mathbf{x}) \in \mathbb{R}$ at an untried $\mathbf{x} \in \mathbb{R}^d$ is assumed (being the required statistical model for the emulator $s(\mathbf{x}) \equiv \hat{S}(\mathbf{x})$), where the entries in the coefficient vector $\mathbf{a}(\mathbf{x}) =$

$(a_1(\mathbf{x}), \ldots, a_m(\mathbf{x}))^T \in \mathbb{R}^d$ are unknown. The *best linear unbiased predictor* for the deterministic response $f(\mathbf{x})$ is then obtained by minimizing the *mean square error*

$$\mathrm{MSE}[\hat{S}(\mathbf{x})] = \mathrm{E}[\mathbf{a}^T(\mathbf{x})\mathbf{f}_X - S(\mathbf{x})]^2 \tag{3.3}$$

under variation of $\mathbf{a}(\mathbf{x})$ and subject to the *unbiasedness constraint*

$$\mathrm{E}[\mathbf{a}^T(\mathbf{x})\mathbf{f}_X] = \mathrm{E}[S(\mathbf{x})]. \tag{3.4}$$

Recall that $\mathbf{f}_X$ is regarded as a realization of $S(\mathbf{x})$. Thus, using (3.4), the mean square error in (3.3) can be rewritten as

$$\sigma^2 \left[\mathbf{a}^T(\mathbf{x})\mathbf{R}\mathbf{a}(\mathbf{x}) - 2\mathbf{a}(\mathbf{x})^T\mathbf{r}(\mathbf{x}) + 1\right],$$

and the linear constraints in (3.4) can be rewritten as

$$\mathbf{H}^T\mathbf{a}(\mathbf{x}) = \mathbf{h}(\mathbf{x}),$$

where

$$\begin{aligned}
\mathbf{h}(\mathbf{x}) &= (h_1, h_2(\mathbf{x}), \ldots, h_k(\mathbf{x}))^T \in \mathbb{R}^k, \\
\mathbf{H} &= (h_j(\mathbf{x}_k))_{1\le k\le m; 1\le j\le k} \in \mathbb{R}^{m\times k}, \\
\mathbf{R} &= (R(\mathbf{x}_i, \mathbf{x}_j))_{1\le i,j\le m} \in \mathbb{R}^{m\times m}, \\
\mathbf{r}(\mathbf{x}) &= (R(\mathbf{x}_1, \mathbf{x}), \ldots, R(\mathbf{x}_m, \mathbf{x}))^T \in \mathbb{R}^m.
\end{aligned}$$

Here, $\mathbf{H}$ is the *design matrix*, $\mathbf{R}$ is the *correlation matrix* between the values of Z at the design sites X, and $\mathbf{r}(\mathbf{x})$ is the correlation vector between Z at the sites X and an untried input $\mathbf{x}$.

Therefore, minimizing (3.3) subject to (3.4) amounts to solving the linear system

$$\begin{bmatrix} \mathbf{R} & \mathbf{H} \\ \mathbf{H}^T & 0 \end{bmatrix} \begin{bmatrix} \mathbf{a}(\mathbf{x}) \\ \mathbf{b}(\mathbf{x}) \end{bmatrix} = \begin{bmatrix} \mathbf{r}(\mathbf{x}) \\ \mathbf{h}(\mathbf{x}) \end{bmatrix}, \tag{3.5}$$

where $\mathbf{b}(\mathbf{x}) = (b_1(\mathbf{x}), \ldots, b_k(\mathbf{x}))^T \in \mathbb{R}^k$ are the Lagrange multipliers for the linear constraints of the quadratic optimization problem.

By solving the linear system (3.5), the best linear unbiased prediction $\hat{S}(\mathbf{x})$ can be written as

$$\hat{S}(\mathbf{x}) = \mathbf{h}^T(\mathbf{x})\hat{\beta} + \mathbf{r}^T(\mathbf{x})\mathbf{R}^{-1}(\mathbf{f}_X - \mathbf{H}\hat{\beta}),$$

where

$$\hat{\beta} = (\mathbf{H}^T\mathbf{R}^{-1}\mathbf{H})^{-1}\mathbf{H}^T\mathbf{R}^{-1}\mathbf{f}_X \tag{3.6}$$

is the usual generalized least squares estimate of coefficients $\beta \in \mathbb{R}^k$ in (3.1).

Note that the kriging method does not only give an estimate of the posterior mean $\hat{S}(\mathbf{x})$ of the stochastic process $S(\mathbf{x})$, but it also provides by

$$\mathrm{MSE}[S(\mathbf{x})] = \sigma^2 \left[1 - [\mathbf{r}^T(\mathbf{x}), \mathbf{h}^T(\mathbf{x})] \begin{bmatrix} \mathbf{R} & \mathbf{H} \\ \mathbf{H}^T & 0 \end{bmatrix}^{-1} \begin{bmatrix} \mathbf{r}(\mathbf{x}) \\ \mathbf{h}(\mathbf{x}) \end{bmatrix}\right]$$

an estimate of the posterior variance of the random process [25]. For further details, see Matheron [17], or the more recent papers [25, 32]. A different formulation based on a maximum probability interpolant can be found in [9].

Now let us finally turn to the model selection problem, where we need to determine the unknown functions and parameters in (3.1). The selection of the functions $h_2, \ldots, h_k$ of the regression part in the stochastic model (3.1) relies on prior knowledge concerning the model response. For instance, linear or quadratic terms are preferred in situations where there is strong evidence to suggest weakly nonlinear trends in the response, or whenever the number of affordable simulations is particularly small.

When using kriging, in combination with hierarchical adaptive design, it is often sufficient to merely include the constant term h_1 in the polynomial regression (3.1) in order to obtain emulators s_ℓ, each of whose performance (in terms of their prediction quality) is at least as good or even superior to that of comparable one-level emulators. This is supported by our numerical results in Section 8.

As regards the selection of the correlation function, R, this also requires prior knowledge concerning the stochastic process. To this end, we follow the common approach in [5, 6, 14, 25, 30], where it is suggested to use a correlation function of the form

$$R(\mathbf{x}, \mathbf{y}) = r(\mathbf{x} - \mathbf{y}) = \exp\left(- \sum_{j=1}^{d} \theta_j |x_j - y_j|^p \right) \tag{3.7}$$

with $\theta_j > 0$, $1 \le j \le d$, and $0 < p \le 2$.

Moreover, by following the recommendations in [30], the unknown coefficients $\theta = (\theta_1, \ldots, \theta_d) \in \mathbb{R}^d$ and p in (3.7) are determined by *maximum likelihood estimation* (MLE). Likewise, the parameter σ for the process variance in (3.2) and $\beta \in \mathbb{R}^k$ are determined by MLE, see [30] for details. In this case, the log-likelihood for the Gaussian process $S(\mathbf{x})$ is given by

$$\ell(\beta, \sigma, \theta, p) \sim -1/2 \left(m \ln \sigma^2 + \ln \det(\mathbf{R}) + (\mathbf{f}_X - \mathbf{H}\beta)^T \mathbf{R}^{-1} (\mathbf{f}_X - \mathbf{H}\beta)/\sigma^2 \right). \tag{3.8}$$

In order to reduce the required computational complexity, it is reasonable to work with the following simplification. Note that with assuming the parameters θ_j, $1 \le j \le d$, and p as fixed, the MLE for $\beta \in \mathbb{R}^k$ is given by the generalized least squares estimate $\hat{\beta} \in \mathbb{R}^k$ in (3.6). Moreover, the MLE estimate for the process variance σ^2 in (3.2) is given by

$$\hat{\sigma}^2 = \frac{1}{m} (\mathbf{f}_X - \mathbf{H}\hat{\beta})^T \mathbf{R}^{-1} (\mathbf{f}_X - \mathbf{H}\hat{\beta}).$$

Using these estimates $\hat{\beta}$ and $\hat{\sigma}^2$ in the log-likelihood (3.8), the coefficients θ_j, $1 \le j \le d$, and p are then determined by maximizing the simpler expression

$$\ell(\theta, p) \sim -1/2 \left(m \ln \sigma^2 + \ln \det(\mathbf{R}) \right). \tag{3.9}$$

Maximizing the expression (3.9) requires global optimization, which can be computationally too expensive. However, as pointed out in [26], maximum likelihood estimation is in practice regarded as sufficiently reliable. But in situations where several regression terms $h_1, \ldots, h_n$ are included in the model (3.1), a *restricted maximum likelihood estimation* (REML) is recommended [26]. In this case, parameter

estimation by REML is accomplished by maximizing the likelihood for transformed data

$$W = \mathbf{C}\mathbf{f}_X \sim \mathrm{N}[\mathbf{C}\mathbf{H}\beta = 0, \sigma^2\mathbf{C}\mathbf{R}(\theta)\mathbf{C}^T],$$

with a transformation matrix $\mathbf{C} \in \mathbb{R}^{(m-k)\times m}$ satisfying $\mathbf{C}\mathbf{H} = 0$, and where $\mathrm{N}(\mu, \sigma^2\mathbf{R})$ denotes the usual statistical normal distribution with mean μ and covariance $\sigma^2\mathbf{R}$. Further details on MLE and REML are immaterial for the purposes of this paper, and so we prefer to refer the interested reader to the textbook [26]. We finally remark that within our hierarchical approximation, Algorithm 1, the computational costs required for the performance of MLE and REML are negligible.

4. Initial Design and Screening. This section first explains in Subsection 4.1 the construction of the utilized initial design, before some relevant aspects concerning screening are briefly addressed in Subsection 4.2.

4.1. Initial Design. To determine the initial design X_1 in Algorithm 1, we prefer to work with Latin hypercube designs (LHC) [18]. This standard technique is easy to use and has mainly two advantages. Firstly, with using LHC most regions of the input domain are well-covered by a well-balanced distribution of the design sites. Secondly, if some of the input variables are found inactive, then the initial design can *uniquely* be projected onto the restricted space of the remaining active variables. Commonly used alternative designs, such as fractional factorial designs (FFD) [8], as they are used in response surface methodology (RSM) [8, 19], usually lack these two important properties, even though they are usually better suited for sensitivity analysis.

LHC designs were first introduced by McKay, Beckman, and Conover [18] in computer experiments. Later on, their utility in numerical integration of real-valued functions of many variables has been discussed in several papers by Owen [20]. As shown in [20], LHC is closely related to quasi-Monte Carlo methods [3].

For the numerical experiments of Section 8, a simplified version of LHC is utilized. To explain this simplified LHC, assume that the range of any input variable is the interval $I = [-1, 1]$. On a preselected number m of input configurations, each of the m input sites in the initial design $X_1 = \{\mathbf{x}_1, \ldots, \mathbf{x}_m\}$ is assumed to have coordinates from

$$I_m = \{-1 + 2i/(m-1) : 0 \le i \le m-1\},$$

where for every input parameter x_i, $1 \le i \le d$, the m corresponding components $\mathbf{x}_1^{(i)}, \ldots, \mathbf{x}_m^{(i)}$ of the sites in X_1 are required to be pairwise distinct. To this end, according to the construction of LHC, the m components $\mathbf{x}_1^{(i)}, \ldots, \mathbf{x}_m^{(i)}$ are chosen from I_m in random order, i.e., for each $1 \le i \le d$, the components $\mathbf{x}_1^{(i)}, \ldots, \mathbf{x}_m^{(i)}$ are a permutation of the m numbers in I_m.

Note that in our formulation of LHD the input coordinates are randomly selected from a *discrete* set, I_m, rather than from a *continuous* parameter interval. The reason for doing so is mainly for the sake of stability. To be more precise, we are aiming at the construction of well-separated design sites. This serves to stabilize the required maximization of matrix determinants by MLE in the subsequent local optimal design. Details on this are explained in Section 6.

Moreover, note that our construction guarantees that the data points in the design X_1 are uniquely projectable to low dimensional point sets, see the second remark at the

outset of this section. This is particularly useful for dimension reduction, especially in situations where we expect only a small set of *active* variables from the tried responses. In this case, we want to discard the remaining *inactive* variables. Using LHC, this can, unlike in FFD, be done unambiguously and without any modelling conflicts by just removing the components of the inactive variables from the design X_1.

As suggested in [30], LHC does also support efficient estimation of the correlation lengths $\lambda_j = 1/\theta_j > 0$ in (3.7). This will be very important in the subsequent steps of our method. As further explained in [30], the correlation lengths λ_j should be used in the screening process. In fact, a very large correlation length λ_j indicates that the response output is not very sensitive to the variation of the input parameter x_j, $1 \leq j \leq d$.

4.2. Screening. Screening usually works with a linear polynomial model, where a specific statistical test, called the *Student test*, is applied. The Student test relies on least squares estimates for the coefficients in the linear model, in order to evaluate their contribution to the response variability. These estimates are used in order to recursively eliminate inactive terms from the model. More details on this standard statistical technique can be found in [19] or in the COUGAR [4] user manual.

As supported by our numerical experiments, a *strict* screening criterion should be preferred to avoid premature discarding of input variables, as they may become significant at a later stage of the hierarchical approximation algorithm. In our implementation, this is accomplished by combining several independent screening criteria, in which case an input variable will only be discarded if it satisfies *all* screening criteria involved, rather than just *one*. For a more detailed discussion on the related sensitivity analysis from a Bayesian perspective, we refer to the recent work of Oakley and O'Hagan [15].

5. Adaptive Gridding. In comparison to the rather easy task of initial design, as described in Subsection 4.1, *optimal design* is a much more challenging task, requiring rather sophisticated optimization techniques. Global sequential optimal designs, as suggested in previous papers [5, 6, 25], are commonly used iterative methods for optimal design in computer experiments. In these methods, a sequence of design sites is constructed for the computational domain Ω, which usually involves very expensive computations to determine an *optimal* design point in each step of the iteration. Due to their large computational costs, the applicability of such methods is rather limited. Moreover, in these global strategies for optimal design, design points often cluster in some region of Ω, whereas they are sparse in other regions of Ω [25]. In fact, sequential optimal designs are often not too efficient, due to a too heterogeneous distribution of the design sites. This section proposes a more efficient approach for hierarchical optimal design, called *adaptive gridding*, as utilized in the design step **(4a)** of Algorithm 1.

Adaptive gridding is a domain decomposition strategy which splits the computational domain Ω into smaller subdomains, *cells*, according to a suitable splitting criterion [13]. In our method, Ω is adaptively partitioned into anisotropic cells of equal size, where the splitting criterion is according to the correlation of the response $f(\mathbf{x})$. By using (3.7), this correlation is chosen as a function $R(\mathbf{x}, \mathbf{y})$ which is rapidly decaying at growing distance between the points $\mathbf{x}$ and $\mathbf{y}$.

Our approach combines global adaptive gridding with local optimal design. To this end, the construction of the design sites is done locally, by applying a local sequential optimal design to the (highly correlated) cells which are output by global

adaptive gridding. The correlation is measured by the correlation lengths $\lambda_i = 1/\theta_i$ of the corresponding input parameters x_i, $1 \le i \le d$.

We shall be more precise on this two-stage procedure in this and in the following section. But let us first make a few more general remarks concerning computational aspects. Note that adaptive gridding, in combination with local optimal design, allows us to distribute the required computations to different processors. Therefore, at any design level ℓ, several computations for new design sites and corresponding simulations may be run, one for each cell, at the same time. In relevant applications, such as in oil reservoir forecasting or climate modelling, where each simulation run may take several hours or even several days, parallelization is in fact essential in order to reduce the required computational time.

Now the basic idea of adaptive gridding is to construct a partitioning of Ω into cells of equal size, where the cell edges have different lengths depending on their spatial direction. Moreover, each cell edge is aligned with one coordinate axis x_i, corresponding to one specific input parameter. The aim of adaptive gridding is to perform the splitting, such that the edge lengths are proportional to the *correlation length* λ_i of the corresponding input parameter x_i, $1 \le i \le d$.

We further explain adaptive gridding in the following Subsection 5.1, before we turn to the construction of a customized error indicator in Subsection 5.2, which allows us to evaluate the prediction quality at each of the individual cells. The latter will establish an important link between global adaptive gridding and local optimal design.

5.1. Adaptive Gridding Algorithm. To define adaptive gridding, suppose that the domain Ω is a hypercuboid in $\mathbb{R}^d$,

$$\Omega = [x_\ell^{(1)}, x_r^{(1)}] \times \cdots \times [x_\ell^{(d)}, x_r^{(d)}] \subset \mathbb{R}^d,$$

and so Ω is defined by its lower left corner point $\mathbf{x}_\ell = (x_\ell^{(1)}, \ldots, x_\ell^{(d)}) \in \mathbb{R}^d$ and its upper right corner point $\mathbf{x}_r = (x_r^{(1)}, \ldots, x_r^{(d)}) \in \mathbb{R}^d$.

We intend to decompose Ω into a collection $\mathcal{L} = \{\omega\}_{\omega \in \mathcal{L}}$ of smaller hypercuboidal subdomains, *cells*, of equal sizes and with pairwise disjoint interior, so that

$$\Omega = \bigcup_{\omega \in \mathcal{L}} \omega$$

yields a partitioning of Ω.

The decomposition of Ω is computed by splitting the $d \times 2^{d-1}$ edges of Ω. Note that each edge e is aligned with one coordinate axis x_i, $1 \le i \le d$. That is, any edge e in Ω, corresponds to one variable x_i, $1 \le i \le d$, and therefore e may be viewed as an interval I_i on coordinate axis x_i. In adaptive gridding, each interval I_i, corresponding to one edge e (and thus to one variable x_i) is uniformly split into n_i pairwise disjoint intervals $I_i^{(1)}, \ldots, I_i^{(n_i)}$, $1 \le i \le d$, of equal length, so that

$$I_i = \bigcup_{j=1}^{n_i} I_i^{(j)}, \qquad \text{with } |I_i^{(j)}| \equiv |I_i|/n_i \quad \text{for } 1 \le j \le n_i,\ 1 \le i \le d.$$

Now the basic idea of adaptive gridding is to perform the splitting of each interval I_i, such that the resulting constant length of the n_i subintervals $|I_i^{(j)}| \equiv |I_i|/n_i$, $1 \le j \le n_i$, is of the order of the correlation length λ_i of variable x_i, i.e., $\lambda_i \sim |I_i|/n_i$, or $n_i \sim |I_i|/\lambda_i$ for $1 \le i \le d$, so that $\prod_{i=1}^d n_i \sim \prod_{i=1}^d |I_i|/\lambda_i$ is the number of cells.

In this way, each of the resulting cells $\omega \in \mathcal{L}$ has long edges along coordinate directions where the variation of the response f is small, whereas short edges of ω correspond to variables with stronger fluctuation of the response. Therefore, adaptive gridding captures preferred directions of the response f quite effectively. This in turn leads to an adaptive distribution of the design points, according to the global behaviour of f.

The adaptive gridding algorithm outputs a uniform partitioning $\mathcal{L} = \{\omega\}_{\omega\in\mathcal{L}}$ of Ω into $\prod_{i=1}^{d} n_i$ smaller cells of equal size, where each cell $\omega \in \Omega$ is a hypercuboid in $\mathbb{R}^d$. For the illustration, we refer to Figure 8.2 of Subsection 8.1, where examples for partitionings of a planar domain $\Omega \subset \mathbb{R}^2$ are shown.

Recall that the adaptive gridding algorithm is performed at each level $\ell \geq 2$ in step **(4a)** of Algorithm 1. In particular, the correlation lengths $\lambda_i \equiv \lambda_i^{(\ell)}$ are reevaluated at each level $\ell \geq 2$, and so the partitioning $\mathcal{L} = \{\omega\}_{\omega\in\mathcal{L}}$ is updated accordingly.

5.2. Cell Prediction Errors. Having applied adaptive gridding to obtain a partitioning $\mathcal{L} = \{\omega\}_{\omega\in\mathcal{L}}$ of the domain Ω into smaller cells ω, further design points are to be added to the cells $\omega \in \mathcal{L}$. This is done by following ideas from our previous work [10], where again adaptivity plays an important role.

To this end, we first assign for the current design X_ℓ a prediction error η_ω to each cell ω, by cross validation. More precisely, let $X_\omega = X_\ell \cap \omega$ denote the set of current design sites lying in cell ω, so that by

$$X_\ell = \bigcup_{\omega\in\mathcal{L}} X_\omega$$

this yields a partitioning of X_ℓ.

Note that the set X_ω of design sites contained in a cell ω may be empty, in which case we let $\eta_\omega = \eta_{\max}$ for the prediction error of cell ω, where $\eta_{\max}$ is a (very high) preselected value. Otherwise, if ω is not empty, we work with a cell prediction error of the form

$$\eta_\omega = \max_{\mathbf{x}_\omega \in X_\omega} |f(\mathbf{x}_\omega) - s_{X_\ell\setminus\mathbf{x}_\omega}(\mathbf{x}_\omega)|, \tag{5.1}$$

where $s_{X_\ell\setminus\mathbf{x}_\omega}$ is the kriging emulator (see Section 3), whose construction uses *all* design sites from X_ℓ except $\mathbf{x}_\omega$. This way, the cell prediction error (5.1) reflects the *local* approximation behaviour, i.e., the quality of the emulator's local prediction at the cell ω.

Indeed, if η_ω is small, then the response f is usually smooth on ω and the emulator makes good predictions, whereas for large η_ω the response is usually subject to strong variation in ω (e.g. for nonlinear behaviour of f), in which case further design points should be added to ω in order to improve the prediction of the emulator.

The cell prediction error is used in combination with a preselected *target accuracy*, $\eta^* \leq \eta_{\max}$, where further points are to be added to every cell ω satisfying $\eta_\omega \geq \eta^*$. We call any cell ω, where (by $\eta_\omega \geq \eta^*$) the emulator's prediction is rather poor, a *bad* cell, otherwise (if $\eta_\omega < \eta^*$) the cell ω is called a *good* cell. Moreover, an empty cell is said to be a bad cell, unless the prediction error η_ω of *every* non-empty cell ω lies below the global threshold η^*. In this case, there are no bad cells in the partitioning $\mathcal{L} = \{\omega\}_{\omega\in\mathcal{L}}$.

Now the insertion of new design points is done by applying a local optimal design (as explained in Section 6) to each bad cell ω, where we add – primarily for the sake

of computational efficiency – only *one* point per (bad) cell ω. Note that the chosen refinement strategy allows one to combine the construction of new design sites with parallel computing, and so to further reduce the computational time required for the experimental design.

The utilized refinement strategy makes up an important link between global adaptive splitting and local optimal design through this two-stage construction of new design sites. Indeed, our construction reduces the likelihood that some unexplored areas of the computational domain Ω are classified as *good* at a too early refinement level by a too optimistic prediction indicator.

Finally let us remark that the target η^* should ideally reflect the desired accuracy threshold of the computer experiment. However, one should note that η^* can only be a rough estimate of the emulator's global prediction error. Nevertheless, as supported by our numerical experiments, the accuracy η^* can in fact be regarded as a suitable estimate for the emulator accuracy. Moreover, the estimate turns out to be useful for subsequent optimization and postprocessing calibration.

In practice, however, the stopping criterion is usually dominated by the maximum number of affordable simulations rather than by the desired problem accuracy. Therefore, any beforehand selection of the target accuracy η^* should not only depend on the required problem accuracy, but also on the available computational power and hardware capacity. Note that by the iterative construction in our hierarchical approximation scheme, the pay-off in terms of number of simulations versus accuracy can be monitored quite effectively during the performance of Algorithm 1.

6. Local Optimal Design. This section explains how the local optimal design is performed. Recall that this step concerns construction of new design sites, to be added to the current set $X \equiv X_\ell$ in step **(4a)** of Algorithm 1.

The local design is performed on a cell ω, which is constructed by adaptive gridding, as explained in the previous Section 5. The local optimal design is, for any cell ω, done by using an *entropy minimization principle*. In this approach, dating back to Lindley [16], a design is determined by minimizing the posterior Shannon's entropy [23].

As later shown by Shewry and Wynn [24], this approach is equivalent to maximizing the prior entropy. For the special case of a Gaussian prior, this in turn is equivalent to maximizing the determinant of the covariance matrix $\sigma^2\mathbf{R}$. For a more detailed discussion on this, we refer to [6, 24]. As a result, regarding β as fixed and considering the variance σ^2 as stationary, this amounts to maximizing the determinant $\det(\mathbf{R}_X)$ of the correlation matrix

$$\mathbf{R}_X = (R(\mathbf{x}_i, \mathbf{x}_j))_{1\le i,j\le m} \in \mathbb{R}^{m\times m}$$

under variation of design sites $X = \{\mathbf{x}_1, \ldots, \mathbf{x}_m\}$, see [6].

Therefore, we wish to insert one point $\mathbf{x} = \mathbf{x}^*$ into ω, which maximizes the determinant $\det(\mathbf{R}_{X\cup\mathbf{x}})$ of the resulting correlation matrix $\mathbf{R}_{X\cup\mathbf{x}}$ among all points in ω. To determine $\mathbf{x}^* \in \omega$, the search for a maximum of $\det(\mathbf{R}_{X\cup\mathbf{x}})$ on ω is done by varying the position of the point $\mathbf{x} = \mathbf{x}_\omega$ in the cell ω, while keeping the design sites in X fixed. The required numerical optimization relies on the algorithm *L-BFGS-B*, a constrained version of the quasi-Newton method, due to Byrd, Lu, Nocedal & Zhu [2]. In this local optimization algorithm, we apply the L-BFGS-B algorithm with different choices for the initial point $\mathbf{x}_0 \in \omega$, namely with the centre of ω and with a small number $n \ll 2^d$ of randomly chosen points from the 2^d corners of ω. We then take a

point $\mathbf{x}^* \in \omega$, where the maximum of the determinant $\det(\mathbf{R}_{X\cup\mathbf{x}})$ among these $n+1$ points is attained.

In our local optimal design, the points in X are, for each cell $\omega \in \mathcal{L}$, given by the union of the points in X_ω and a *small* set of neighbours around ω. Therefore, the size of the matrix $\mathbf{R}_X$ is *small*. In our numerical experiments, we considered using a number of n_ω *nearest* neighbours of the centre $\mathbf{c}_\omega$, where n_ω is between 15 and 20. In order to determine these nearest neighbours for $\mathbf{c}_\omega \in \omega$, we considered working with the *Mahalanobis distance*

$$\mathbf{d}(\mathbf{x},\mathbf{y}) = \sqrt{\sum_{j=1}^{n}\left(\frac{x_j - y_j}{\lambda_j}\right)^2}, \qquad \text{for } \mathbf{x} = (x_1,\ldots,x_d),\ \mathbf{y} = (y_1,\ldots,y_d),$$

for measuring the distance between two points. This is in order to take different statistical variations for the different input parameters better into account.

Note that this procedure can recursively be applied to any cell ω, just in case one wants to add more than one new design point into ω. This then complies with the ideas of (local) sequential optimal design.

7. Formulation of the Hierarchical Approximation Algorithm. Now that each step of Algorithm 1 has been explained in detail we provide a more precise schematic description in the following algorithm.

ALGORITHM 2. **(Hierarchical Nonlinear Approximation)**

INPUT: *Simulator* $f : \mathbb{R}^d \to \mathbb{R}$.

- **(1)** *Perform initial Latin hypercube design (LHC) to obtain* X_1.
- **(2)** *Determine coefficients* θ_j, p *in (3.7) by maximum likelihood estimation.*
- **(3)** *Run the simulator* f *at the initial design* X_1*, and so obtain the initial data* $\mathbf{f}_{X_1} = \{f(\mathbf{x}) : \mathbf{x} \in X_1\}$.
- **(4)** *Apply screening by using our criterion* $\theta_j \sim 0$*, for all* $1 \le j \le d$*. Determine the inactive input variables and reduce the input space by removing the inactive variables.*
- **(5)** *Compute the initial emulator* s_1 *from* X_1 *and* $\mathbf{f}_{X_1}$ *by using kriging.*
- **(6) FOR** $\ell = 2, 3, \ldots$ **DO**
  - **(6a)** *Recalculate the coefficients* θ_j, p *in (3.7) by MLE.*
  - **(6b)** *Refine the current cells* ω *by using global adaptive gridding.*
  - **(6c)** *Compute the cell prediction error* η_ω *for each current cell* ω*, and then determine all bad cells satisfying* $\eta_\omega \ge \eta^*$.
  - **(6d)** *For each bad cell* ω*, add one point to* ω *by applying local optimal design to* ω*, and so obtain new design sites* $D_\ell = X_\ell \setminus X_{\ell-1}$.
  - **(6e)** *Run the simulator* f *at the new sites* D_ℓ*, and so obtain the new responses* $\mathbf{f}_{D_\ell} = \{f(\mathbf{x}) : \mathbf{x} \in D_\ell\}$.
  - **(6f)** *Compute the next emulator* s_ℓ *from* X_ℓ *and* $\mathbf{f}_{X_\ell}$ *by using kriging.*

OUTPUT: *Sequence* $s_1, s_2, \ldots, s_L$ *of emulators with increasing accuracy.*

8. Numerical Results. We have implemented the proposed hierarchical approximation scheme, Algorithm 2, for arbitrary space dimension d by using the programming language `R` [21]. This section investigates the performance of our algorithm. To this end, we work with three different model problems, one involving an analytical function (Subsection 8.1), and two model problems from reservoir forecasting (Subsection 8.2), called *PUNQS* and the *IC Fault* model. In the two examples concerning reservoir forecasting, the method is compared with the standard *response surface*

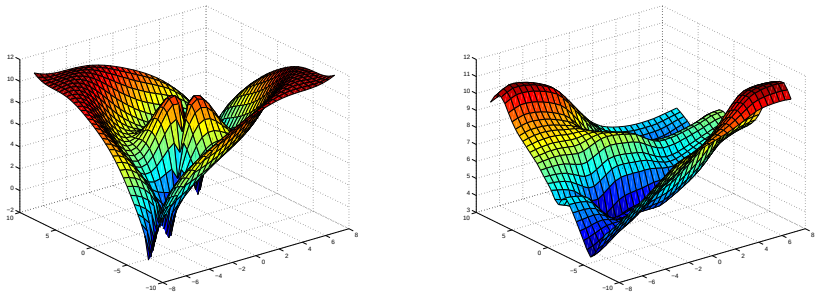
methodology (RSM), see [8, 19], which is utilized in reservoir uncertainty analysis software, such as COUGAR [4].

In each numerical comparison, the required number of simulations is fixed, and the quality of the resulting emulators is compared by their prediction accuracy. Note that the required number of simulations reflects the required computational costs, since each simulator run takes much more time than the computational time required for building the emulator, no matter which method is employed.

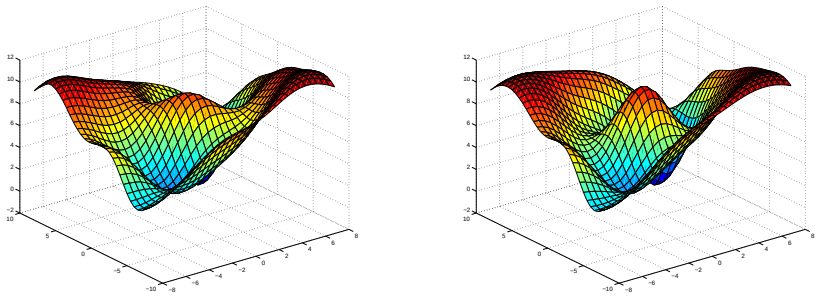
8.1. Analytical Test Example. In this first test case, consider the trivariate function

$$f(x,y,z) = 7\,\frac{\sin(\sqrt{x^2+y^2})+\epsilon}{\sqrt{x^2+y^2}} + 3|x-y|^{1/2} + 0.01z,$$

where $\epsilon = 10^{-7}$, which is regarded as the "simulator". Note that f is linear in variable z, but nonlinear in x and y. In fact, the two variables x and y are identified as *active* variables, whereas z is classified as *inactive*. This conclusion was reached in the preprocessing screening, where it was found that $\lambda_3 \gg \max(\lambda_1, \lambda_2)$. This allows us to view $f \equiv f_z(x,y)$, for fixed z, as a bivariate surface, where $z \equiv 0$ after screening. The graph of the "true response surface" f is shown in Figure 8.1 (top left).



simulator $f(x,y)$ emulator $s_1(x,y)$



emulator $s_2(x,y)$ emulator $s_3(x,y)$

FIG. 8.1. *Hierarchical approximation. Graphs of the simulator f and emulators s_1, s_2, s_3.*

The correlation lengths λ_1 and λ_2 belonging to the remaining two active variables, x and y, are shown, along with the parameter p in (3.7), in Table 8.1 for three levels

$\ell = 1, 2, 3$. For the construction of the initial design X_1, we apply a Latin hypercube design (LHC) to generate 18 points, and so the size of the initial point set X_1 is $|X_1| = 18$.

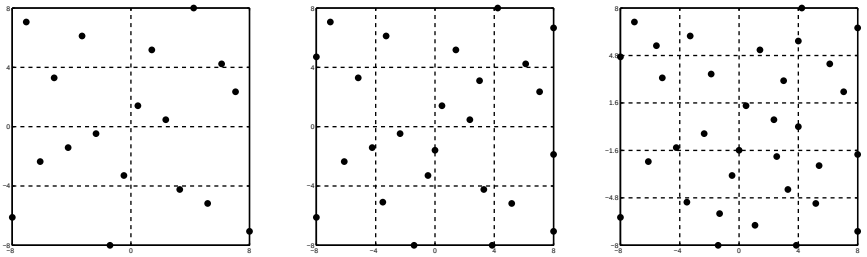
TABLE 8.1

Analytic test example. Correlation parameters at three levels, $\ell = 1, 2, 3$, and prediction accuracy of emulator sequence s_1, s_2, s_3. Note the definitions of η_1, η_2, and η_∞ in equations (8.1), (8.2), and (8.3); n_1 and n_2 are the grid dimensions.

ℓ	$\lvert X_\ell\rvert$	λ_1	λ_2	p	n_1	n_2	η_1	η_2	η_∞
1	18	13.5	4.9	1.4	2	4	1.3	1.8	7.6
2	25	11.4	10.3	2.0	4	4	0.8	1.1	5.0
3	33	14.4	11.0	2.0	4	5	0.5	1.1	4.8

Then, the surface $f : \Omega \to \mathbb{R}$ is approximated on the computational domain $\Omega = [-8, 8] \times [-8, 8]$ by an emulator, whose construction is done by using our hierarchical approximation scheme. In this method, we apply kriging without including a regression part in the stochastic model (3.1).

Figure 8.1 shows the graphs of the three resulting emulators, s_1, s_2, s_3, from coarse to fine, and Figure 8.2 shows the corresponding sequence X_1, X_2, X_3 of three design sets.



X_1 X_2 X_3

FIG. 8.2. *Hierarchical approximation. Adaptive grids and designs X_1, X_2, X_3.*

Note from the visual appearance of the surface graphs in Figure 8.1 that the prediction quality of emulators s_1, s_2, s_3 is gradually improved. This is confirmed by our numerical results, which are shown in Table 8.1. Table 8.1 displays for each level $\ell = 1, 2, 3$ the number of points $|X_\ell|$ in the design set X_ℓ, the resulting *mean absolute error*

$$\eta_1 = \|f - s_\ell\|_1/|G| = \frac{1}{|G|} \sum_{\mathbf{x}\in G} |f(\mathbf{x}) - s_\ell(\mathbf{x})|, \tag{8.1}$$

mean square error,

$$\eta_2 = \left(\|f - s_\ell\|_2^2/|G|\right)^{1/2} = \sqrt{\frac{1}{|G|} \sum_{\mathbf{x}\in G} |f(\mathbf{x}) - s_\ell(\mathbf{x})|^2}, \tag{8.2}$$

and *maximum error*,

$$\eta_\infty = \|f - s_\ell\|_\infty = \max_{\mathbf{x}\in G} |f(\mathbf{x}) - s_\ell(\mathbf{x})|, \tag{8.3}$$

where G denotes a fine uniform grid contained in Ω.

Note that by our adaptive design, that is by the efficient distribution of the sites in X_1, X_2, X_3, the features of f are captured remarkably well, especially in regions where f has large derivatives.

8.2. Test Cases from Reservoir Forecasting.

8.2.1. The PUNQS Reservoir Test Case. In this second example, we analyze the case of a reservoir simulator containing seven uncertain scalar input variables and one scalar output. The example is generated by using the PUNQS [8] reservoir model in combination with Schlumberger's multi-phase flow simulator ECLIPSE [22]. The input variables and their range of variation were suggested by reservoir engineers. In this model problem, the utilized input variables are corresponding to the aquifer strength, the residual gas oil saturation, the residual water oil saturation, the vertical and the horizontal permeability multipliers of the low and of the high quality sands. We have analyzed several outputs of the ECLIPSE simulator. The numerical results, presented in this subsection, were obtained by regarding the seven input variables, and the output oil production rate at 13 years of a given production well (called PRO-15 in the model [8]).

A Latin hypercube design (LHC) is used to generate the initial design X_1 comprising 50 different configurations, i.e., $|X_1| = 50$. In the screening preprocess, the input parameter x_4 is classified as inactive. In fact, the corresponding correlation length $\lambda_4 \gg 1000$, as predicted by maximum likelihood estimation, is much larger than the initial correlation lengths of the other six input variables. Therefore, x_4 is discarded, the correlation lengths of the other six variables are for each level $\ell = 1, \ldots, 5$ shown in Table 8.2.

TABLE 8.2
PUNQS Model. Correlation parameters at different levels.

ℓ	$\lvert X_\ell \rvert$	λ_1	λ_2	λ_3	λ_4	λ_5	λ_6	λ_7	p
1	50	3.6	5.2	3.5	-	2.0	7.3	3.9	1.6
2	93	4.8	3.2	0.3	-	2.0	7.3	3.9	1.6
3	134	182	0.7	1.3	-	1000	2.3	0.2	1.3
4	200	147	0.8	1.0	-	1000	2.0	0.5	1.3
5	258	141	0.8	0.8	-	1000	2.1	0.4	1.2

Recall that by the Latin hypercube design (LHC), the projection of the initial 7-dimensional design to the active 6-dimensional design is injective, and so it does not produce coincident data points.

The construction of the emulators s_ℓ is done by using our hierarchical approximation scheme, Algorithm 2. In this method, we apply kriging with using a quadratic regression part in the stochastic model (3.1) at each level ℓ, where the regression terms $x_1, x_2, x_3, x_5, x_6, x_1x_5, x_1^2$ were found to be dominant by the utilized statistical data analysis.

In order to analyze the accuracy of the emulators s_ℓ, we compare the emulators' function values $s_\ell(\mathbf{x})$ with the response output $f(\mathbf{x})$ of the simulator at 200 randomly selected configurations of input parameters $\mathbf{x}$, which are not contained in the design sites X_ℓ. We let $\eta^* = 3$ for the target accuracy. Our numerical results are shown in Table 8.3.

TABLE 8.3

*PUNQS Model. Results obtained from our hierarchical approximation and comparison with kriging using LHC (***LHC***) and with RSM (***RSM***). Note the definitions of η_1, η_2, and η_∞ in equations (8.1), (8.2), and (8.3).*

ℓ	$\lvert X_\ell \rvert$	η_1	η_2	η_∞
1	50	4.62	5.60	14.80
2	93	3.86	4.90	15.70
3	134	3.58	4.70	15.10
4	200	3.18	4.30	13.30
5	258	2.88	4.20	13.80
LHC	258	2.90	4.70	16.30
RSM	79	3.93	4.84	13.10
RSM	258	3.93	4.80	12.90

The numerical results in Table 8.3 document significant gain in performance between consecutive levels, in terms of the emulators' accuracy, see the different errors in Table 8.3. At final level $\ell = 5$, the target accuracy, $\eta^* = 3$, is approximated sufficiently well, as the mean absolute error drops from $\eta_1 = 4.62$ (at initial level $\ell = 1$) to $\eta_1 = 2.88$ (at $\ell = 5$), and so we obtain $\eta_1 \leq \eta^*$ at final level $\ell = 5$.

In order to further evaluate the performance of our hierarchical design, we have performed another test run based on a *one-level* Latin hypercube design with 258 points, matching the number of points $|X_5| = 258$ in the hierarchical design at the final level $\ell = 5$. The numerical results are shown in the sixth row (**LHC**) of Table 8.3. Note that our hierarchical approximation method is superior to this plain LHC in terms of the emulator accuracy. Indeed, while the mean absolute error η_1 of the emulator s_5 and that of the one-level LHC emulator, say s_{LHC}, are about equal, the emulator s_5 reduces the root mean square error η_2 of s_{LHC} by about 10 %, and the maximum error η_∞ of s_{LHC} is reduced by even about 20 %.

We can explain the significant gain concerning the reduction of the maximum error η_∞ as follows. Due to the hierarchical design, the design sites are adaptively inserted, such that at each refinement step new points are inserted in regions (by local optimal design) where the current maximum error is likely to be highest. This interactive prediction cannot be achieved by a plain one-level LHC, where *all* design sites are selected beforehand in a preprocessing step.

We have also compared our hierarchical approximation method with a polynomial regression model, as usually used in the standard *response surface methodology* (RSM) [8, 19]. Two experiments were made; the first uses 79 simulations selected from a central composite face centered design, the second uses 258 design sites generated from a one-level LHC.

In the first experiment the regression terms were selected by using the step by step fit used in COUGAR [4] based on sensitivity analysis arguments similar to the ones described in Section 4 for the screening process. The results, displayed in the penultimate row of Table 8.3, are about comparable to the ones obtained by our emulator s_2, at a slightly smaller maximum error though. The difference in maximum error is mainly due to small overshoots of the kriging emulator in local areas of the domain, where the simulator f is subject to strong variation. The emulator s_{RSM} manages to somewhat reduce these undesired local overshoots.

However, in the second experiment, displayed in the last row (**RSM**) of Table 8.3,

our emulator s_5 is clearly superior to the comparable one-level method RSM. Indeed, the emulator s_5 of our hierarchical method reduces the mean absolute error η_1 obtained by RSM (i.e., by emulator s_{RSM}), by more than 25 %, and the root mean square error η_2 of s_{RSM} is reduced by about 15 %. The resulting maximum error η_∞ of s_5, however, is again slightly larger than that of the emulator s_{RSM}.

Our numerical results also show that the emulator s_{RSM} does not significantly improve its accuracy at the insertion of more design points, see the last two rows in Table 8.3. In contrast, our hierarchical emulator gradually reduces the prediction errors η_1 and η_2 at increasing refinement level.

We finally remark that the variation of the response surface of the PUNQS model is rather small. More complicated test case scenarios from relevant applications are dealing with discontinuous response surfaces of strong variation. One example is the challenging IC Fault model treated in the following subsection.

8.2.2. The IC Fault Model. In this final example, we apply our hierarchical approximation method to a more difficult test case from Imperial College, called the *IC Fault* model, which has previously been used for parameter identification by history matching. The IC Fault model incorporates a rather complicated discontinuous response surface. Therefore, the IC Fault model is regarded as a very challenging test example, where standard methods for history matching often fail [29]. For a detailed description on the IC Fault Model, see [29].

Let us first discuss only some of the relevant features of the IC Fault model. The simplified reservoir of the IC Fault model has three uncertain input parameters, corresponding to the fault throw, the good and the poor sand permeability multipliers. The analyzed output from the response surface is in this test case the oil production rate at 10 years. The three input parameters are referred to as x_1, x_2 and x_3. According to [29], the input configurations are uniformly distributed in the computational domain $\Omega = [0, 50] \times [100, 200] \times [0, 50]$.

Having rescaled each of the variables' range to the interval $[-1, 1]$, the reservoir simulator ECLIPSE is run on a Latin hypercube design consisting of $|X_1| = 22$ different configurations for the initial design X_1. We then apply our hierarchical approximation method to the IC Fault model, using Algorithm 2, where we let $\eta^* = 0.5$ for the target accuracy. This results in a hierarchical approximation with six design levels, that is $L = 6$.

The construction of the emulators s_ℓ is done by using our hierarchical approximation scheme, Algorithm 2. In this method, we apply kriging with using a quadratic regression part in the stochastic model (3.1) at each level ℓ, where the regression terms x_2, x_3, x_3^2 were found to be dominant by the utilized statistical data analysis.

In order to evaluate the prediction accuracy of emulator s_ℓ at each design level ℓ, $1 \le \ell \le 6$, the simulator was run at an additional set X comprising 200 points, where the points in X are generated by another Latin hypercube design (LHC), being different from the one used to generate the initial design X_1 in our hierarchical approximation method.

We have recorded the mean absolute error η_1, the root mean square error η_2, as well as the maximum error η_∞ for our emulator s_ℓ at each level. The numerical results are shown in Table 8.4. Note that the mean absolute error η_1 is at the final level $\ell = 6$ very close to the target accuracy $\eta^* = 0.5$, after only 146 simulations.

For the purpose of comparison, we consider two alternative (one-level) methods to compare their numerical results with those of our emulator. The first emulator, s_{RSM}, is obtained by response surface methodology (RSM) in combination with a fractional

TABLE 8.4

*IC Fault Model. Results obtained from our hierarchical approximation method; comparison with RSM using FFD (***RSM***), and kriging using LHC (***KRG***). Note the definitions of η_1, η_2, and η_∞ in equations (8.1), (8.2), and (8.3).*

ℓ	$\lvert X_\ell \rvert$	η_1	η_2	η_∞
1	22	3.6	5.4	28.7
2	49	1.4	3.2	25.2
3	72	1.1	1.8	12.5
4	102	0.8	1.5	11.3
5	124	0.7	1.3	8.6
6	146	0.6	1.0	5.4
RSM	22	8.7	10.0	25.8
RSM	146	7.2	8.4	22.9
KRG	146	1.0	1.5	11.7

factorial design (FFD). For the emulator s_{RSM} a quadratic regression polynomial was used. The second emulator, s_{KRG}, is obtained by using kriging in combination with a Latin hypercube design (LHC). Table 8.4 shows the numerical results obtained by using these two emulators, s_{RSM} and s_{KRG}.

Note that the final emulator s_6 of our hierarchical approximation method (see Table 8.4) is superior to both the kriging emulator s_{KRG} (see Table 8.4, last row) and the emulator s_{RSM} (see Table 8.4, second last row). Indeed, the mean absolute error η_1 and the maximum error η_∞ obtained by s_6 is about half of the corresponding error obtained by s_{KRG}, whereas the root mean square error η_2 of s_{KRG} is reduced by a third. Note that a comparable prediction accuracy of s_{KRG} is obtained by s_3 after only three steps of our hierarchical approximation method, with using only $|X_3| = 72$ simulator runs rather than $|X_6| = 146$ runs.

As for the comparison between our emulator and that of RSM, s_{RSM}, the performance of our hierarchical emulator is much better than that of s_{RSM}. Indeed, note that the accuracy of our final emulator s_6 is for the mean absolute error η_1 more than ten times smaller than that of the emulator s_{RSM}, when using $|X_6| = 146$ design points, whereas its root mean square error η_2 is about eight times smaller than that of s_{RSM}, and the maximum error η_∞ of s_6 is only about one quarter of the maximum error of s_{RSM}.

Note that the accuracy of our initial emulator, s_1, is superior to the accuracy of the emulator s_{RSM}, when using only $|X_1| = 22$ design points. Indeed, while the maximum error η_∞ of s_1 is comparable to that of s_{RSM}, the mean absolute error η_1 of s_1 is less than half of the corresponding mean absolute error of s_{RSM}. Moreover, the root mean square error η_2 of s_1 is about half the corresponding mean square error of s_{RSM}, see Table 8.4, first row, and Table 8.4, third last row. That s_1 outperforms s_{RSM} is not too surprising, since s_1 is the best linear unbiased predictor for the deterministic response $\mathbf{f}_{X_1}$, see the discussion in Section 3.

Finally, Table 8.5 shows the correlation lengths λ_i, $1 \le i \le 3$, of the correlation parameters for the three different input parameters, at the six different design levels. Note that input parameter x_3 has, in comparison with x_1 and x_2, a rather short correlation length λ_3. This behaviour is well-treated by the adaption strategy (i.e., global adaptive gridding in combination with local optimal design) of our hierarchical approximation method, which prefers to insert further design points (at run time of

Algorithm 2) in the direction of coordinate axis x_3. This enhanced flexibility gives our hierarchical approximation method another advantage over commonly used non-hierarchical methods which are not designed to determine potentially dominant input parameters.

TABLE 8.5
IC Fault Model. Correlation parameters at different levels.

ℓ	$\lvert X_\ell \rvert$	λ_1	λ_2	λ_3	p
1	22	16.40	1.70	0.40	2.00
2	49	24.00	42.20	0.13	1.80
3	72	22.90	89.90	0.40	1.20
4	102	34.00	98.80	0.12	1.40
5	124	16.10	37.40	0.10	1.50
6	146	10.40	37.70	0.30	1.00

9. Conclusion and Future Work. We have proposed a hierarchical nonlinear approximation scheme for scalar-valued multivariate functions. The main objective of the utilized iterative refinement is to obtain a sufficiently accurate approximation at very few function evaluations. The hierarchical approximation method has been applied to relevant applications from experimental design and statistical data analysis in computer experiments. Our hierarchical method combines different important requirements for efficient designs in computer experiments, such as screening, entropy minimization, statistical data fitting, and parallelization. This results in a flexible and efficient iterative method, whose main components are adaptive gridding, local optimal design, and kriging, in combination with maximum likelihood estimation. As supported by numerical examples, including two real-world model problems from reservoir forecasting, the performance of our hierarchical approximation method is in terms of the resulting emulators' prediction accuracy superior to that of traditional non-hierarchical methods, such as the widely used response surface methodology (RSM). Further important aspects of this research are uncertainty reduction and calibration, which we will treat in a forthcoming paper.

Acknowledgements. We wish to thank Jonathan Rougier for many useful suggestions and fruitful discussions. Moreover, we received much technical help and advice from our colleagues at Schlumberger. The work of Daniel Busby, while at the Schlumberger Abingdon Technology Center, and his collaboration with Armin Iske was supported by Schlumberger Abingdon Technology Center and by the European Union through the project FAMOUS (Fast Model Utilization and Screening), contract no. ENK6-CT-2002-50528. Chris L. Farmer thanks the Royal Society for support through an Industry Fellowship at the University of Oxford. Last but not least, we thank the anonymous referees for their useful comments and constructive suggestions, which helped improve a previous version of the paper.

REFERENCES

[1] M. D. BUHMANN, *Radial Basis Functions*. Cambridge University Press, Cambridge, UK, 2003.
[2] R. H. BYRD, P. LU, J. NOCEDAL, AND C. ZHU, *A limited memory algorithm for bound constrained optimization*, SIAM J. Scientific Computing, 16 (1995), pp. 1190–1208.
[3] R. E. CAFLISCH, *Monte Carlo and quasi-Monte Carlo methods*, Acta Numerica, 1998, pp. 1–49.
[4] The COUGAR project, `http://consortium.ifp.fr/cougar/`.

[5] P. S. Craig, M. Goldstein, J. C. Rougier, and A. H. Seheult, *Bayesian forecasting for complex systems using computer simulators*, Journal of the American Statistical Association, 96 (2001), pp. 717–729.
[6] C. Currin, T. Mitchell, M. Morris, and D. Ylvisaker, *A Bayesian approach to the design and analysis of computer experiments*, ORNL Technical Report 6498, National Technical Information Service, Springfield, Va. 22161, 1988.
[7] D. R. Cox and N. Reid, *The Theory of the Design of Experiments*, Chapman and HALL/CRC, 2000.
[8] J. P. Dejean and G. Blanc, *Managing uncertainties on production predictions using integrated statistical methods*, SPE Journal, SPE 56696, 1999.
[9] C. L. Farmer, *Geological modelling and reservoir simulation*, in Mathematical Methods and Modelling in Hydrocarbon Exploration and Production, A. Iske and T. Randen, eds., Springer, Berlin, 2005, pp. 119–212.
[10] T. Gutzmer and A. Iske, *Detection of discontinuities in scattered data approximation*, Numerical Algorithms, 16 (1997), pp. 155–170.
[11] A. Iske, *Multiresolution Methods in Scattered Data Modelling*, Springer, Berlin, 2004.
[12] A. Iske and T. Randen, eds., *Mathematical Methods and Modelling in Hydrocarbon Exploration and Production*, Springer, Berlin, 2005.
[13] A. Iske and J. Levesley *Multilevel scattered data approximation by adaptive domain decomposition*, Numerical Algorithms, 39 (2005), pp. 187–198.
[14] M. C. Kennedy and A. O'Hagan, *Bayesian calibration of computer models*, Journal of the Royal Statistical Society B., 63 (2000), pp. 425–464.
[15] J. Oakley and A. O'Hagan, *Probabilistic sensitivity analysis of complex models: a Bayesian approach*, Journal of the Royal Statistical Society B., 16 (2004), pp. 751–769.
[16] D. V. Lindley, *On a measure of the information provided by an experiment*, The Annals of Mathematical Statistics, 27 (1956), pp. 986–1005.
[17] G. Matheron, *Principles of geostatistics*, Economic Geology, 58 (1963), pp. 1246–1266.
[18] M. D. McKay, R. J. Beckman, and W. J. Conover, *A comparison of three methods for selecting values of input variables in the analysis of output from a computer code*, TECHNOMETRICS, 21 (1979), pp. 239–245.
[19] R. H. Myers and D. C. Montgomery, *Response Surface Methodology: Process and Product Optimization Using Designed Experiments*, Second ed., Wiley-Interscience, 2002.
[20] A. B. Owen, *Monte Carlo variance of scrambled net quadrature*, SIAM J. Numer. Anal., 34 (1997), pp. 1884–1910.
[21] R Development Core Team, *R: a language and environment for statistical computing*, `http://www.R-project.org`, 2004.
[22] Schlumberger GeoQuest, *ECLIPSE Technical Description Manual*, 2001.
[23] C. E. Shannon, *A mathematical theory of communication*, The Bell System Technical Journal, 27 (1948), pp. 379–423, pp. 623–656.
[24] M. C. Shewry and H. P. Wynn, *Maximum entropy sampling*, J. Appl. Statist., 14 (1987), pp. 165–170.
[25] J. Sacks, W. J. Welch, T. J. Mitchell, and H. P. Wynn, *Design and analysis of computer experiments*, Statistical Science, 4 (1989), pp. 409–435.
[26] T. J. Santner, B. J. Williams, and W. I. Notz, *The Design and Analysis of Computer Experiments*, Springer, New York, 2003.
[27] M. Sambridge, *Geophysical inversion with a neighbourhood algorithm I – searching a parameter space*, Geophys. J. Int., 138 (1999), pp. 479–494.
[28] M. Sambridge, *Geophysical inversion with a neighbourhood algorithm II – appraising the ensemble*, Geophys. J. Int., 138 (1999), pp. 727–746.
[29] Z. Tavassoli, J. N. Carter, and P. R. King, *Errors in history matching*, SPE Journal, SPE 86883, 2004.
[30] W. J. Welch, R. J. Buck, J. Sacks, H. P. Wynn, T. J. Mitchell, and M. D. Morris, *Screening, predicting, and computer experiments*, Technometrics, 34 (1992), pp. 15–25.
[31] H. Wendland, *Scattered Data Approximation*. Cambridge University Press, Cambridge, UK, 2005.
[32] T. Werther, *Optimal multivariate interpolation*, in Mathematical Methods and Modelling in Hydrocarbon Exploration and Production, A. Iske, T. Randen, eds., Springer, Berlin, 2005, pp. 389–407.