Review of surrogate modeling in water resources

Saman Razavi,1 Bryan A. Tolson,1 and Donald H. Burn1

Received 19 October 2011; revised 4 June 2012; accepted 12 June 2012; published 31 July 2012.

[1] Surrogate modeling, also called metamodeling, has evolved and been extensively used over the past decades. A wide variety of methods and tools have been introduced for surrogate modeling aiming to develop and utilize computationally more efficient surrogates of high-fidelity models mostly in optimization frameworks. This paper reviews, analyzes, and categorizes research efforts on surrogate modeling and applications with an emphasis on the research accomplished in the water resources field. The review analyzes 48 references on surrogate modeling arising from water resources and also screens out more than 100 references from the broader research community. Two broad families of surrogates namely response surface surrogates, which are statistical or empirical data-driven models emulating the high-fidelity model responses, and lower-fidelity physically based surrogates, which are simplified models of the original system, are detailed in this paper. Taxonomies on surrogate modeling frameworks, practical details, advances, challenges, and limitations are outlined. Important observations and some guidance for surrogate modeling decisions are provided along with a list of important future research directions that would benefit the common sampling and search (optimization) analyses found in water resources.


1. Introduction

[2] Computer simulation models, which simulate abstract representations of physically based systems using mathematical concepts and language, are playing a key role in engineering tasks and decision making processes. There are various types of problems utilizing computer simulation models (for simplicity referred to as simulation models hereafter) including prediction, optimization, operational management, design space exploration, sensitivity analysis, and uncertainty analysis. There are also problems such as model calibration and model parameter sensitivity analysis dealing with simulation models to enhance their fidelity to the real-world system. Fidelity in the modeling context refers to the degree of the realism of a simulation model. Modern simulation models tend to be computationally intensive as they rigorously represent detailed scientific knowledge about the real-world systems [Keating et al., 2010; Mugunthan et al., 2005; Zhang et al., 2009]. Many model-based engineering analyses require running these simulation models thousands of times and as such demand prohibitively large computational budgets.

[3] Surrogate modeling, which is a second level of abstraction, is concerned with developing and utilizing cheaper-to-run “surrogates” of the “original” simulation models. Throughout this paper, the terms “original functions,” “original simulation models,” and “simulation models” are used interchangeably. A wide variety of surrogate models have been developed to be intelligently applied in lieu of simulation models. There are two broad families under the large umbrella of surrogate modeling, response surface modeling and lower-fidelity modeling. Response surface surrogates employ data-driven function approximation techniques to empirically approximate the model response. Response surface surrogates may also be referred to as “metamodels” [Blanning, 1975; Kleijnen, 2009] as a response surface surrogate is a “model of a model.” “Model emulation” is another term referring to response surface surrogate modeling [O’Hagan, 2006]. The term “Proxy models” has also been used in the literature to refer to response surface surrogates [Bieker et al., 2007]. Unlike response surface surrogates, lower-fidelity surrogates are physically based simulation models but less-detailed compared to original simulation models, which are typically deemed to be high-fidelity models; they are simplified simulation models preserving the main body of processes modeled in the original simulation model [Forrester et al., 2007; Kennedy and O’Hagan, 2000].

[4] In a surrogate modeling practice (response surface surrogates or lower-fidelity physically based surrogates), the goal is to approximate the response(s) of an original simulation model, which is typically computationally intensive, for various values of explanatory variables of interest. The surface representing the model response with respect to the variables of interest (which is typically a nonlinear hyperplane) is called “response surface” or “response landscape” throughout this paper. For the majority of response surface surrogate modeling techniques, different response surfaces must be fit to each model response of interest (or each
function aggregating multiple model responses). The neural network technique is one exception capable of fitting multiple model responses. In contrast, since lower-fidelity surrogates retain some physically based characteristics of the original model, one lower-fidelity surrogate model is typically capable of approximating multiple model responses of interest.

The main motivation of developing surrogate modeling strategies is to make better use of the available, typically limited, computational budget. Simpson et al. [2008] report that the common theme in six highly cited metamodeling (or design and analysis of computer experiments) review papers is indeed the high cost of computer simulations. Global optimization algorithms based on response surface surrogates such as EGO [Jones et al., 1998], GMSRBF and MLMSRBF [Regis and Shoemaker, 2007a], and Gutmann’s method [Gutmann, 2001] and also uncertainty analysis algorithms such as ACUARS [Magunthan and Shoemaker, 2006] and RBF-enabled MCMC [Bliznyuk et al., 2008] all have been developed to circumvent the computational budget limitations associated with computationally intensive simulation models. In this regard, surrogate modeling may only be beneficial when the simulation model is computationally intensive, justifying the expense of moving to a second level of abstraction (reduced model fidelity) which typically leads to reducing the accuracy of analyses. Therefore, even though Jones [2001] and Simpson et al. [2008] both point out that surrogate modeling is more than simply reducing computation time, reviewing other possible motivations for surrogate modeling is beyond the scope of this paper.

Many iterative water resources modeling analyses potentially stand to benefit from surrogate modeling. Benefits are only potential because any surrogate-enabled modeling analysis provides an approximation to the analysis with the original model and the error of the analysis result seems difficult or impossible to assess without repeating the exact analysis with the original simulation model. For example, there is no guarantee that a model parameter deemed insensitive on the basis of surrogate modeling analysis is truly insensitive in the original simulation model. An incomplete list of classic or popular iterative modeling analyses in water resources, which are candidates for efficiency enhancement with surrogate modeling, include deterministic model parameter optimization (calibration) studies with evolutionary algorithms [e.g., Duan et al., 1992; Wang, 1991], uncertainty-based or Bayesian model calibration studies [e.g., Beven and Freer, 2001; Kavetski et al., 2006; Vrugt et al., 2009], management or design optimization with evolutionary algorithms [e.g., McKinney and Lin, 1994; Savic and Walters, 1997], multiobjective optimization algorithms [e.g., Cieniawski et al., 1995; Reed et al., 2003], global sensitivity analysis methods [e.g., Hornberger and Spear, 1981; Saltelli et al., 2000], and any traditional Monte Carlo-based reliability or uncertainty analysis [e.g., Melching et al., 1990; Skaggs and Barry, 1997].

This paper aims to review, analyze, and classify the research on surrogate modeling with an emphasis on surrogate modeling efforts arising from the water resources modeling field. Simpson et al. [2001] and Wang and Shan [2007] also review the literature on response surface surrogates for engineering design optimization problems. Simpson et al. [2004] summarize a discussion panel on response surface surrogate modeling held at the 9th AIAA/ISSMO Symposium on Multidisciplinary Analysis and Optimization. Simpson et al. [2008] review the literature on response surface modeling and motivations from a historical perspective and also emphasize the appeal of lower-fidelity physically based surrogate modeling. Forrester and Keane [2009] review recent advances in surrogate modeling including advances in lower-fidelity physically based surrogates in the field of optimization. Special journal issues on surrogate modeling summarize the first and second International Workshops on Surrogate Modeling and Space Mapping for Engineering Optimization (see Bandler and Madsen [2001] and Bandler et al. [2008]). Another special issue publication on surrogate modeling is a recent thematic journal issue on surrogate modeling for the reduction and sensitivity analysis of complex environmental models (see Ratto et al. [2012]). In addition, there are more specific review papers focusing on specific tools/strategies involved in surrogate modeling. Kleijnen [2009] reviews kriging and its applications for response surface surrogate modeling. Jin et al. [2001] and Chen et al. [2006] review and compare multiple function approximation models acting as response surface surrogates. Jin [2005] focuses on response surface surrogate modeling when used with evolutionary optimization algorithms.

Surrogate modeling has been increasingly more popular over the last decade within the water resources community and this is consistent with the increasing utilization of metamodels in the scientific literature since 1990 as documented by Viana and Haftka [2008]. A research database search of formal surrogate modeling terminology in mid-2011 (search of Thomson Reuters (ISI) Web of Knowledge) in 50 journals related to surface water and groundwater hydrology, hydrological, environmental science and engineering, and water resources planning and management returned 110 articles on surrogate modeling in water resources. We believe that the actual number of articles on surrogate modeling in water resources is higher as there are articles not using the formal terminology of surrogate modeling and/or not published in water resources related journals. Forty-eight of the available surrogate modeling references published until mid-2011 dealing with water or environmental resources problems were selected for detailed review based on their relevance, our judgment of their quality and clarity of reporting and the significance of surrogate modeling to the contribution of the publication. The phrase “water resources literature” used throughout this paper refers to the 50 water resources journals and these 48 surrogate modeling references.

1.1. Goals and Outline of Review

Our primary objective (1) is to provide water resources modelers considering surrogate modeling with a more complete description of the various surrogate modeling techniques found in the water resources literature along with some guidance for the required subjective decisions when utilizing surrogate models. The depth of the review of the topics covered here generally varies with the popularity of the topic in the water resources literature and as such, discussion largely revolves around optimization applications. Additional more specific objectives are as follows: (2) describe each of the components involved in surrogate modeling practice as
depicted in Figure 1; (3) provide a categorization of the different surrogate-enabled analysis frameworks (i.e., the different ways the components in Figure 1 can interact); (4) relate existing surrogate modeling efforts in the water resources literature with similar efforts in the broader research community; and (5) identify relevant underutilized ideas for consideration in future water resources studies.

[10] Figure 1 presents a diagram that shows all the components involved in the surrogate modeling analysis framework and the sections in the paper that are directly related to each component. Conventional frameworks not involving surrogate models, such as different simulation-optimization frameworks, consist of only the original model and the search or sampling algorithm components being directly linked together. In surrogate-enabled frameworks, however, three new components, design of experiments, response surface surrogate, and/or lower-fidelity surrogate, may also be involved. These three components, and the framework through which all the components interact, are of particular interest in this paper. Such frameworks generally begin with a design of experiments to generate a sample with which to train or fit a response surface or lower-fidelity surrogate model; then the sampler/search algorithm repeatedly runs the original computationally expensive model and/or the surrogate and collects their response. During this metamodel-enabled analysis, the surrogate model can be static or dynamically updated. Any original model evaluation which is utilized to fit the surrogate model is referred to as a design site. Section 2 details the elements associated with the response surface surrogates and presents their advances, considerations, and limitations. Section 3 presents the motivation and different types and frameworks for lower-fidelity surrogate modeling. Section 4 discusses how the performance of a surrogate-enabled framework should be evaluated and benchmarked against other alternatives. The paper ends with summary and concluding remarks in section 5.

1.2. Case Study or Problem Characteristics Influencing Surrogate Model Design

[11] The most critical problem characteristics that should influence surrogate model/technique selection are as follows:

1. Model analysis type to be augmented by the surrogate model—search or sampling. For the remainder of this paper, search analysis is meant to refer to optimization (management, calibration, single-objective or multiobjective) or uncertainty-based/Bayesian model calibration procedure while all other modeling analyses are referred to as sampling analyses.

2. Computational budget constraints. This refers to how many original model evaluations can be utilized to build the surrogate model and ultimately perform the model analysis of interest. In applications where a surrogate model is to be repeatedly utilized after it is initially constructed (i.e., optimize real-time operational decisions), the time available for each utilization can be critical.

3. Dimensionality of the problem. In general, as the number of explanatory variables increases, surrogate modeling becomes less advantageous and even infeasible.

4. Single-output versus multivariate surrogates. This is a key distinction in the context of environmental simulation modeling, where model outputs are typically variable in both time and space. Single-output surrogates are common where the original model output of interest is a function calculated from a large number of model outputs (i.e., calibration error metric).

5. Exact emulation versus inexact emulation. In other words, should the surrogate model exactly predict the original model result at all design sites?

6. Availability of original model developers/experts. Some surrogate modeling techniques require these experts (lower fidelity modeling), and Gorissen [2007] notes that they can provide valuable insight into the significance of surrogate modeling errors relative to original model errors.

Although not an original problem characteristic, the availability of surrogate modeling software and experts also has an impact on surrogate model design. The aforementioned surrogate modeling reviews and the literature in general do not precisely map all problem characteristics to specific or appropriate types of surrogate models. We also do not attempt this and instead only make periodic observations and judgments as to when certain types of surrogate modeling techniques might be more or less appropriate.

![Figure 1. Diagram of the elements involved in surrogate (metamodel) enabled analysis framework.](image-url)
than others. Properly considering the case study specific factors above is the first key to avoid implementing a poor surrogate modeling technique.

2. Response Surface Surrogates

[19] Response surface surrogate modeling as a research field arising from various disciplines has been in existence for more than six decades and has become very active since the beginning of 1990s [Simpson et al., 2008]. The first generation of response surface surrogates, initiated by Box and Wilson [1951], relied heavily on polynomials (typically second-order) and have been the basis of the so-called response surface methodology (RSM). Response surface surrogates do not emulate any internal component of original simulation models; instead they approximate the relationships between several explanatory variables, typically the simulation model parameters and/or variables affecting model inputs, and one or more model response variables. In other words, a response surface surrogate is an approximation or a model of the “original” response surface defined in a problem domain—response surface surrogates are metamodels of original models. The terms “metamodel” and “response surface surrogate” are used interchangeably throughout this paper. Notably, the existence of the response surface concept dates back to well before formalizing modern response surface approaches as, for example, the traditional nonlinear local optimization techniques based on the Taylor series expansion (i.e., different variations of Newton’s method) use simple approximations. In such methods, a response surface surrogate, typically a variation of polynomials, is locally fitted on the (single) current best solution through the use of first- and/or second-order derivative information of the original function, unlike the formalized response surface approaches in which surrogates are fitted on multiple design sites usually regardless of derivatives. Trust-region methods are another example family of traditional local optimization strategies based on the response surface concept, as they iteratively approximate a certain region (the so-called trust region) of the original function typically using polynomials.

[20] Research and advances in response surface surrogate modeling can be classified into three main categories: (1) identification and development of experimental designs for effective approximation, (2) developing and applying function approximation techniques as surrogates, and (3) framework development utilizing surrogates. The research efforts in the water resources community tend to focus on the second and third categories. In the following, the literature on response surface surrogate modeling for water resources applications is reviewed in section 2.1. Then sections 2.2–2.6 further detail this review in relation with the above three categories and outline the advances, considerations, and limitations.

2.1. Response Surface Surrogates in Water Resources Literature

[21] Response surface surrogate modeling has been widely applied in various water and environmental modeling problems for decades. Table 1 summarizes 32 studies utilizing response surface surrogate models in water resources problems published since 2000. Although this table does not cover all surrogate modeling studies over that period, we believe that it provides readers with an adequate coverage of the subject area. Note that not all these studies have been published in water resources related journals. According to Table 1, about 45% of the surrogate modeling studies focus on automatic model calibration. Most surrogate-enabled auto-calibration studies involve surrogates in conjunction with optimization algorithms; in four studies, surrogates have been used with uncertainty analysis algorithms for the purpose of model calibration (i.e., GLUE in the work of Khu and Werner [2003] and Zhang et al. [2009], ACUARS in the work of Megunthan and Shoemaker [2006], and Markov Chain Monte Carlo in the work of Bliznyuk et al. [2008]). Schultz et al. [2004, 2006] and Borgonovo et al. [2012] also use surrogates to speed up Monte Carlo sampling studies. Ten studies of the surrogate model applications listed in Table 1 are groundwater optimization problems, mostly groundwater remediation. Four surrogate modeling studies are in the context of water distribution system design and optimization. Borgonovo et al. [2012] is the only study in Table 1 using surrogate modeling for sensitivity analysis and interested readers are thus referred to Blatman and Sudret [2010], Ratto et al. [2007], and Storlie et al. [2009] for metamodel-enabled sensitivity analysis examples from the broader research community. Five studies, Liong et al. [2001], Bui and Mayer [2006], Behzadian et al. [2009], di Pierro et al. [2009], and Castelletti et al. [2010], use response surface surrogates in multiobjective optimization settings. In addition, most studies fit the response surface surrogates on continuous explanatory variables. Five studies [i.e., Bui and Mayer, 2006; Behzadian et al., 2009; Broad et al., 2005; Broad et al., 2010; Castelletti et al., 2010] apply surrogates to integer optimization problems (response surface surrogates are fitted on discrete variables), and Yan and Minsker [2006, 2011] and Hemker et al. [2008] apply surrogate modeling in mixed-integer optimization settings. Note that problems with discrete or mixed variables require special considerations and have been solved largely on an ad hoc basis [Simpson et al., 2004]. For mixed-integer optimization problems, Hemker et al. [2008] utilize response surface surrogates within a branch-and-bound optimization framework such that the surrogate model is employed when solving the relaxed optimization subproblems (integer variables allowed to assume noninteger values). Shrestha et al. [2009] use neural networks as a surrogate to completely replace computationally intensive Monte Carlo sampling experiments. In their approach, neural networks are used to emulate the predictive uncertainty (i.e., 90% prediction intervals) of a hydrologic model. The surrogate in this study does not follow the general strategy of response surface modeling where surrogates map model parameters or other problem variables to the system response, and instead their neural network model maps observed rainfall and runoff in the preceding time intervals to the model predictive uncertainty in the next time interval. In sections 2.2–2.6, and also section 4, we refer back to the studies listed in Table 1 to elaborate the details related to the subject of each section.

2.2. Design of Experiments

[22] Deterministic simulation systems like computer simulations can be very complex involving many variables with complicated interrelationships. Design of Experiments
<table>
<thead>
<tr>
<th>Work by</th>
<th>Type of Problem</th>
<th>Type of Metamodel</th>
<th>Type of Search or Sampling Algorithm</th>
<th>Type of Framework</th>
<th>Number and Type of Explanatory Variables</th>
<th>Computational Saving</th>
</tr>
</thead>
<tbody>
<tr>
<td>Johnson and Rogers [2000]</td>
<td>Groundwater remediation</td>
<td>Linear Regression and Neural network (single hidden layer)</td>
<td>Simulated Annealing</td>
<td>Basic sequential framework</td>
<td>30 continuous</td>
<td>Not reported</td>
</tr>
<tr>
<td>Liong et al. [2001]</td>
<td>Automatic calibration of the HydroWorks watershed model (Bi-objective)</td>
<td>Neural network (three hidden layers)</td>
<td>Accelerated convergence GA (ACGA)</td>
<td>Basic sequential framework</td>
<td>8 continuous</td>
<td>Not reported</td>
</tr>
<tr>
<td>Khu and Werner [2003]</td>
<td>Uncertainty based Automatic calibration of the SWMM rainfall-runoff model</td>
<td>Neural network (single hidden layer)</td>
<td>GLUE</td>
<td>Basic sequential framework (Instead of formal DoE, GA with niching was used to generate design sites)</td>
<td>8 continuous</td>
<td>80% in number of full evaluations</td>
</tr>
<tr>
<td>Schultz et al. [2004, 2006]</td>
<td>Uncertainty analysis of a water quality model for regulatory decision support</td>
<td>Linear Regression, polynomial, and process-inspired simple functions</td>
<td>Monte Carlo Simulation</td>
<td>Basic sequential framework</td>
<td>9 continuous</td>
<td>Not reported</td>
</tr>
<tr>
<td>Khu et al. [2004]</td>
<td>Automatic calibration of MIKE11/NAM</td>
<td>Radial basis functions</td>
<td>Genetic algorithms</td>
<td>Metamodel-embedded evolution framework, No formal DoE</td>
<td>9 continuous</td>
<td>60% in number of full evaluations</td>
</tr>
<tr>
<td>Regis and Shoemaker [2004]</td>
<td>Groundwater bioremediation optimization and 17 test functions</td>
<td>Radial basis functions and second-order polynomial (both local)</td>
<td>(μ,λ)-evolution strategy</td>
<td>Metamodel-embedded evolution framework</td>
<td>12 continuous</td>
<td>Not reported</td>
</tr>
<tr>
<td>Ostfeld and Samonos [2005]</td>
<td>Automatic calibration of CE-QUAL-W2 water quality model</td>
<td>k-nearest neighbors (kNN)</td>
<td>Genetic algorithms</td>
<td>Metamodel-embedded evolution framework, No formal DoE</td>
<td>3 continuous</td>
<td>Not reported</td>
</tr>
<tr>
<td>Mugunthan et al. [2005]</td>
<td>Automatic calibration of a groundwater bioremediation model</td>
<td>Radial basis functions</td>
<td>Gaussian random sampler</td>
<td>Approximation uncertainty based framework</td>
<td>8 continuous</td>
<td>Not reported</td>
</tr>
<tr>
<td>Broad et al. [2005]</td>
<td>Water distribution system design optimization</td>
<td>Neural network (single hidden layer)</td>
<td>Genetic algorithms</td>
<td>Basic sequential framework</td>
<td>22 discrete</td>
<td>21% of CPU time</td>
</tr>
<tr>
<td>Mugunthan and Shoemaker [2006]</td>
<td>Automatic calibration and parameter uncertainty analysis of groundwater models</td>
<td>Radial basis functions</td>
<td>Gaussian random sampler</td>
<td>Approximation uncertainty based framework</td>
<td>3 and 7 continuous</td>
<td>More than 87% in number of full evaluations</td>
</tr>
<tr>
<td>Bai and Mayer [2006]</td>
<td>Pump-and-treat optimization (Bi-objective)</td>
<td>Kriging</td>
<td>Complete enumeration</td>
<td>Approximation uncertainty based framework</td>
<td>4 discrete</td>
<td>Not reported</td>
</tr>
<tr>
<td>Yan and Minsker [2006]</td>
<td>Groundwater remediation design – MODFLOW and RT3D models used for flow field and contaminant concentration</td>
<td>Neural network (single hidden layer)</td>
<td>Genetic algorithms</td>
<td>Metamodel-embedded evolution framework, No formal DoE</td>
<td>9 and 28 mixed integer</td>
<td>85-90% in number of full evaluations</td>
</tr>
<tr>
<td>Zou et al. [2007]</td>
<td>Automatic calibration of the WASP water quality model</td>
<td>Neural network (single hidden layer)</td>
<td>Genetic algorithms</td>
<td>Basic sequential adaptive-recursive frameworks</td>
<td>19 continuous</td>
<td>97% of CPU time</td>
</tr>
<tr>
<td>Regis and Shoemaker [2007a]</td>
<td>Groundwater bioremediation optimization and 17 test functions</td>
<td>Radial basis functions</td>
<td>Gaussian and uniform random samplers</td>
<td>Approximation uncertainty based framework</td>
<td>Test functions: 2-14 Groundwater problem: 12 continuous</td>
<td>Not reported</td>
</tr>
<tr>
<td>Shoemaker et al. [2007]</td>
<td>Automatic calibration of two SWAT watershed models</td>
<td>Radial basis functions</td>
<td>(μ,λ)-evolution strategy</td>
<td>Metamodel-embedded evolution framework, No formal DoE</td>
<td>8 and 14 continuous</td>
<td>Not reported</td>
</tr>
<tr>
<td>Work by</td>
<td>Type of Problem</td>
<td>Type of Metamodel</td>
<td>Type of Search or Sampling Algorithm</td>
<td>Type of Framework</td>
<td>Number and Type of Explanatory Variables</td>
<td>Computational Saving</td>
</tr>
<tr>
<td>---------</td>
<td>-----------------</td>
<td>-------------------</td>
<td>--------------------------------------</td>
<td>-------------------</td>
<td>----------------------------------------</td>
<td>----------------------</td>
</tr>
<tr>
<td>Hemker et al. [2008]</td>
<td>Well field design optimization and hydraulic capture problem</td>
<td>Kriging</td>
<td>Branch-and-bound on main problem – Sequential quadratic programming on subproblems</td>
<td>Adaptive-recursive framework</td>
<td>5+ integer with 15+ continuous</td>
<td>Not reported</td>
</tr>
<tr>
<td>Biely et al. [2008]</td>
<td>Automatic calibration and Bayesian uncertainty analysis of an environmental model</td>
<td>Radial basis functions</td>
<td>Markov-Chain Monte Carlo Sampler</td>
<td>Basic sequential framework</td>
<td>4 continuous</td>
<td>More than 90% in number of full evaluations</td>
</tr>
<tr>
<td>Koumakos and Mantoglou [2009]</td>
<td>Coastal aquifer pumping management</td>
<td>Neural network (modular) single hidden layer</td>
<td>Evolutionary Annealing Simplex Scheme optimization</td>
<td>Metamodel-embedded evolution framework, No formal DoE</td>
<td>34 continuous (For metamodeling, disaggregated into smaller subsets having members with negligible correlations with members of other subsets)</td>
<td>95% of CPU time</td>
</tr>
<tr>
<td>Fen et al. [2009]</td>
<td>Cost and contaminant removal optimization for soil vapor extraction system design</td>
<td>Second-order polynomials and exponential functions</td>
<td>Genetic algorithms</td>
<td>Adaptive-recursive framework</td>
<td>6 and 9 continuous</td>
<td>40-94% of total CPU time across 4 case studies&lt;sup&gt;a&lt;/sup&gt;</td>
</tr>
<tr>
<td>Behzadian et al. [2009]</td>
<td>Water distribution system monitoring locations optimization (Bi-objective)</td>
<td>Neural network (single hidden layer)</td>
<td>NSGA-II</td>
<td>Metamodel-embedded evolution framework, No DoE</td>
<td>15 and 50 discrete</td>
<td>87% and 96% of CPU time (~34% of total saving is due to caching) 20-42% of total CPU across 4 case studies</td>
</tr>
<tr>
<td>Zhang et al. [2009]</td>
<td>Uncertainty-based automatic calibration of SWAT models</td>
<td>Support vector machines (SVMs) and Neural network (single hidden layer)</td>
<td>GLUE (was used only with SVMs, ANN was used for benchmarking SVMs)</td>
<td>Basic sequential framework</td>
<td>6, 9, 12, and 16 continuous</td>
<td>95% of CPU time</td>
</tr>
<tr>
<td>Zou et al. [2009]</td>
<td>Automatic calibration of the WASP water quality model</td>
<td>Neural network (single hidden layer)</td>
<td>Genetic algorithms</td>
<td>Adaptive-recursive framework</td>
<td>19 continuous</td>
<td>Not reported</td>
</tr>
<tr>
<td>Shrestha et al. [2009]</td>
<td>Predictive uncertainty estimation of a hydrologic model (HBV)</td>
<td>Neural network (single hidden layer)</td>
<td>N/A (see section 2.1)</td>
<td>N/A (see section 2.1)</td>
<td>N/A (see section 2.1)</td>
<td>Not reported</td>
</tr>
<tr>
<td>di Pierro et al. [2009]</td>
<td>Water distribution system design optimization (bi-objective)</td>
<td>Kriging</td>
<td>Genetic algorithms</td>
<td>Approximation uncertainty based</td>
<td>34 and 632 (continuous)</td>
<td>N/A Inferior performance</td>
</tr>
<tr>
<td>Broad et al. [2010]</td>
<td>Water distribution system design optimization</td>
<td>Neural network (single hidden layer)</td>
<td>Genetic algorithms</td>
<td>Basic sequential framework</td>
<td>49 discrete</td>
<td>98% of CPU time</td>
</tr>
<tr>
<td>Castelletti et al. [2010]</td>
<td>Water quality rehabilitation in reservoirs (multiobjective optimization)</td>
<td>Radial basis functions, n-dimensional linear interpolator, and inverse distance weighted</td>
<td>Genetic algorithms</td>
<td>Complete enumeration</td>
<td>Adaptive-recursive framework</td>
<td>3 discrete</td>
</tr>
</tbody>
</table>
Table 1 (continued).

<table>
<thead>
<tr>
<th>Work by</th>
<th>Type of Problem</th>
<th>Type of Metamodel</th>
<th>Type of Framework</th>
<th>Type of Search or Sampling Algorithm</th>
<th>Number and Type of Explanatory Variables</th>
<th>Computation Savings</th>
</tr>
</thead>
<tbody>
<tr>
<td>Lo and Mansur [2011]</td>
<td>Coastal aquifer pumping and contaminant transport</td>
<td>Ensemble of genetic programming (single hidden layer)</td>
<td>Basic sequential framework</td>
<td>Genetic algorithms and Gaussian random samplers</td>
<td>26 mixed integer</td>
<td>Not reported</td>
</tr>
<tr>
<td>Sreekant and Datta [2012]</td>
<td>Reliability-based groundwater remediation design</td>
<td>Randomization and Latin hypercube sampling</td>
<td>Test functions: 2, 3, and 3</td>
<td>Test functions: 2, 3, and 3</td>
<td>15 continuous</td>
<td>Not reported</td>
</tr>
<tr>
<td>Mugunthan et al. [2002]</td>
<td>Human dose risk assessment</td>
<td>Gaussian process (single hidden layer)</td>
<td>Basic sequential framework</td>
<td>Gaussian process (single hidden layer)</td>
<td>12 continuous</td>
<td>Not reported</td>
</tr>
<tr>
<td>Regis and Shoemaker [2007a]</td>
<td>Environmental and hydrogeological processes</td>
<td>Smoothing spline ANOVA models</td>
<td>Monte Carlo Simulation</td>
<td>Monte Carlo Simulation</td>
<td>Test functions: 2, 3, and 3</td>
<td>90% of CPU time</td>
</tr>
</tbody>
</table>

(Also referred to as DoEs) employ different space filling strategies to empirically capture the behavior of the underlying system over limited ranges of the variables. As a priori knowledge about the underlying (original) response surface is usually unavailable, DoEs tend to assume uniformity in distributing the commonly called “design sites,” which are the points in the explanatory (input) variable space evaluated through the original simulation model. Most metamodel-enabled optimizers start with DoEs as in 23 out of 32 studies listed in Table 1. There are a wide variety of DoE methods available in the literature; however, full factorial design (e.g., used for metamodeling in the work of Gutmann [2001]), fractional factorial design, central composite design [Montgomery, 2008], Latin hypercube sampling (LHS) [McKay et al., 1979], and symmetric Latin hypercube sampling (SLHS) [Ye et al., 2000] appear to be the most commonly used DoE methods. Full and fractional factorial designs and central composite design are deterministic and typically more applicable when the number of design variables is not large. For example, the size of the initial DoE sampled by the full factorial design in a 10 dimensional space with only two levels would be 1024 (≈10^2), which may be deemed extremely large and beyond the computational budget. Latin hypercube sampling and symmetric Latin hypercube sampling both involve random procedures and can easily scale to different numbers of design variables. Research efforts on the metamodel-inspired DoEs mostly focus on determining the optimal type [e.g., Alam et al., 2004] and size of existing DoEs (i.e., number of initial design sites, e.g., Sobester et al. [2005]) for a specific problem or to develop new and effective DoEs [e.g., Ye et al., 2000].

2.3. Function Approximation Models

[22] Response surface surrogates approximate the response surface of computationally intensive models (i.e., original models) by fitting over a set of previously evaluated design sites. A variety of approximation techniques have been developed and applied as surrogates. Examples of such techniques include: polynomials [Fen et al., 2009; Hussain et al., 2002; Myers and Montgomery, 2002; Wang, 2003], kriging, which is also sometimes referred to as design and analysis of computer experiment (DACE) in the surrogate modeling context [Sacks et al., 1989; Sakata et al., 2003; Simpson and Mistree, 2001], k nearest neighbors (kNN) [Ostfeld and Salomons, 2005], artificial neural networks (ANNs) [Bezhad et al., 2009; Khu and Werner, 2003; Papadrakakis et al., 1998], radial basis functions (RBFs) [Hussain et al., 2002; Mugunthan et al., 2005; Mullur and Messac, 2006; Nakayama et al., 2002; Regis and Shoemaker, 2007a], support vector machines (SVMs) [Zhang et al., 2009], multivariate adaptive regression splines (MARS) [Friedman, 1991; Jin et al., 2001], high-dimensional model representation [Rabitz et al., 1999; Ratto et al., 2007; Sobol, 2003], treed Gaussian processes [Gramacy and Lee, 2008], Gaussian emulator machines [Kennedy and O’Hagan, 2001; O’Hagan, 2006], smoothing splines ANOVA models [Gu, 2002; Ratto and Pagano, 2010; Storlie et al., 2011; Wahba, 1990], and proper orthogonal decomposition [Audouze et al., 2009]. In the RBF references above, the RBFs are not associated with ANNs, whereas in some publications [e.g., Jin, 2005;
Khu et al., 2004], as well as in the MATLAB neural network toolbox [Beale et al., 2010], RBFs are considered as a type of feed-forward artificial neural networks.

[24] In the recent water resources related response surface surrogate modeling literature, ANNs and RBFs are the most commonly used function approximation techniques as, among all surrogate modeling studies listed in Table 1, 14 and 10 of the 32 studies have applied ANNs and RBFs, respectively. In addition, according to Table 1, polynomials have been used in five studies, and in two of these [i.e., Johnson and Rogers, 2000; Regis and Shoemaker, 2004], polynomials were used to highlight their weakness in comparison with other more promising alternatives. Five studies in Table 1 have employed kriging. Each of SVMs, kNN, and smoothing spline ANOVA has been used in only one study. Moreover, the approximation models mostly act as global surrogates of underlying functions, which represent the original models over the entire input range. However, in some studies [e.g., Regis and Shoemaker, 2004; Wang et al., 2004], the approximation model is fitted locally only over a specified (limited) number of design sites (a subset of all available design sites), which are in close vicinity of the point of interest in the explanatory variable space.

[25] The information used to fit the response surface surrogates are typically the response values of the original function (i.e., original simulation model) at the design sites; however, there are studies aiming to include the sensitivity (gradient information) of the original function with respect to the explanatory variables (i.e., derivative values) to enhance the approximation accuracy and form the so-called “gradient-enhanced response surface surrogates.” Examples include Kim et al. [2005] and van Keulen and Vervenne [2004] for gradient-enhanced polynomials and Liu [2003] for gradient-enhanced kriging and gradient-enhanced artificial neural networks. In practice, such methods have serious limitations as in most of the problems that response surface surrogates are applied, the derivatives are not readily available and have to be calculated using numerical methods requiring the evaluation of the original function at extra points. The extra computational burden imposed can become prohibitively large when the number of dimensions in the explanatory variable space is more than a few, whereas this extra computation could be saved to evaluate the original function at new more intelligently sampled points.

[26] Selection of an appropriate function approximation technique for a given surrogate-enabled analysis requires careful consideration. There are significant differences in the logic inspiring the development of different techniques and also in their level of practicality for a particular problem. In the following, some practical and technical details are presented on five common function approximation techniques used as response surface surrogates of computer simulation models (polynomials, RBFs, kriging, SVMs, and ANNs), all of which have been used for surrogate modeling in the water resources literature. SVMs were selected for detailed review in addition to the four most common techniques in Table 1 because Viana and Haftka [2008] include them as one of the four main classes of techniques in their surrogate modeling review paper. The basic information and formulations of these techniques were not included as they are available elsewhere.

2.3.1. Polynomials and Simple Functions

[27] Polynomials have the simplest type of parameters (i.e., coefficients in a polynomial regression), which are objectively determined usually through the least square regression method. Second-order polynomial functions, which are the most popular polynomials used as response surface surrogates, have \((D + 1)(D + 2)/2\) parameters where \(D\) is the number of explanatory variables (dimension of the input space). First- and second-order polynomials have had very successful applications in local nonlinear programming optimization algorithms (e.g., different variations of gradient-descent and Newton/ quasi-Newton methods) where, for example, a second-order polynomial is used to emulate a local mode in the original response landscape. However, the use of polynomials as global surrogates may be only plausible when the original response landscape is, or is reasonably close to, unimodal, which is not often the case in many water resources related problems.

[28] Application of higher-order polynomials (third-order or more, which are common in curve-fitting) is typically infeasible when \(D\) is greater than only a few variables. This is largely because specifying a proper polynomial form for a particular problem may become very challenging. Second, the number of polynomial parameters to be tuned (and therefore the minimum number of design sites required, see also section 2.6.3) becomes excessively large. The inferior performance of polynomials compared to other function approximation techniques mostly due to having fairly inflexible prespecified forms and being inexact emulators (see section 2.6.2) has been acknowledged in several studies [e.g., Hussain et al., 2002; Regis and Shoemaker, 2004; Simpson and Mistree, 2001]. Note that when the form of the underlying function is similar to a polynomial and when this form is known a priori (e.g., in the work of Fen et al. [2009] polynomials had been reportedly successful for the problem of interest) polynomials can be one of the best options to choose for surrogate modeling.

[29] Other simple functional forms such as exponential functions [Aly and Peralta, 1999; Fen et al., 2009] fitted by least square methods may also be applied for response surface modeling. Schultz et al. [2004] and Schultz et al. [2006] develop “reduced-form models” based on components of the actual process equations in an original model and fit them to design sites sampled from the original model. They point out that when the specified functional forms “are informed by the mechanics” of the original model, the reduced-form models demonstrate better predictive (generalization) ability. Notably, what is called a reduced-form model in these publications is different from the lower-fidelity physically based surrogates outlined in section 3.

2.3.2. Radial Basis Functions

[30] Radial basis function (RBF) models consist of a weighted summation of typically \(n\) (sometimes fewer) radial basis functions (also called correlation functions) and a polynomial (usually zero- or first-order) where \(n\) is the number of design sites. There are different forms of basis functions including Gaussian, thinplate spline, and multiquadric. Some forms (e.g., Gaussian) have parameters specifying the sensitivity/spread of the basis function over the input domain, while some (e.g., thinplate spline) have fixed sensitivity (no parameter in the basis function)
regardless of the scale (unit) and importance or sensitivity of each input variable. To address the scale problem, all the data are typically normalized to the unit interval. Moreover, in RBF models, the sensitivity of the basis functions in all $D$ directions is typically assumed identical (only one single parameter, if utilized, in all dimensions for all basis functions) treating all variables as equally important, although such an assumption is often not true. An RBF model is defined by the weights of the basis functions, the coefficients of the polynomial used, and the basis function parameter if it exists. Weights of the basis functions as well as the polynomial coefficients can be objectively determined by efficient least squares techniques; however, the basis function parameter is usually specified arbitrarily or by trial and error. The minimum number of design sites required to fit an RBF model is the number of coefficients of the polynomial used to augment the RBF approximation.

### 2.3.3. Kriging

[31] Application of kriging in the context of design and analysis of computer experiments (DACE) was first formalized by Sacks et al. [1989] and since then has been frequently called DACE in some publications [Hemker et al., 2008; Ryu et al., 2002; Simpson et al., 2001]. More recent literature uses DACE to refer to the suite of all metamodel/emulation techniques [e.g., Ratto et al., 2012; Simpson et al., 2008]. Similar to RBF models, the kriging model is also a combination of a polynomial model, which is a global function over the entire input space, and a localized deviation model (correlation model consisting of basis functions) based on spatial correlation of samples. The special feature of kriging (main difference from RBF models) is that kriging treats the deterministic response of a computer model as a realization of a stochastic process, thereby providing a statistical basis for fitting. This capability enables kriging to provide the user with an approximation of uncertainty associated with the expected value predicted by kriging at any given point. Approximation of uncertainty is the basis of the so-called “approximation uncertainty based framework” for surrogate-enabled analyses, described in section 2.4.4. Note that such approximation of uncertainty may be available (heuristically or directly) for other function approximation models (see section 2.4.4).

[32] As opposed to RBF models, the correlation parameters (sensitivities) in kriging are typically different along different directions in the input space ($D$ different values for correlation functions in a $D$-dimensional space), resulting in higher model flexibility. All the kriging parameters, including the correlation parameters, can be determined objectively using the maximum likelihood estimation methodology. Like RBF models, the minimum number of design sites needed to fit a kriging model is the number of coefficients in the polynomial augmenting the approximation. Kriging users only need to specify the lower and upper bounds on the correlation parameters, although the appropriate bounds are sometimes hard to specify [Kleijnen, 2009]. The kriging correlation parameters can be interpreted to some extent in that large values for a dimension indicate a highly nonlinear function in that dimension, while small values indicate a smooth function with limited variation. Moreover, Jones et al. [1998] pointed out that the correlation matrix in kriging may become ill-conditioned (nearly singular) toward the end of an optimization run as then the optimization algorithm tends to sample points near the previously evaluated points (design sites); this ill-conditioning is usually manageable.

### 2.3.4. Support Vector Machines

[33] Support vector machines (SVMs) are a relatively new set of learning methods designed for both regression and classification. Although SVMs to some extent rely on utilizing the concept of basis (correlation) functions as used in RBFs and kriging (especially when using Gaussian kernel function), unlike RBFs and kriging, they only involve a subset of design sites lying outside an $\varepsilon$-insensitive tube around the regression model response, referred to as support vectors, to form an approximation. When fitting the SVMs on data, the $\varepsilon$-insensitive tube is formed to ignore errors that are within a certain distance of the true values. This capability enables SVMs to directly control and reduce the sensitivities to noise (very suitable for inexact emulation, see section 2.6.2). The other special feature of SVMs is that in the SVM formulation, there is a term directly emphasizing the regularization (smoothness) of the fitted model. There are two specific SVM parameters which are associated with the radius of the $\varepsilon$-insensitive tube and the weight of the regularization term. The Kernel function used within SVM (e.g., Gaussian function) may also have a parameter to be determined; this parameter acts like the correlation parameters in RBF and kriging models and adjusts the sensitivity/spread of the Kernel function with respect to explanatory variables. Users only require dealing with these two or three SVM parameters, and once their values are available, the dual form of the SVM formulation can be efficiently solved using quadratic programming to determine all the other SVM formulation parameters. SVMs can better handle larger numbers of design sites as the operator associated with design site vectors in the SVM formulation is dot product [Yu et al., 2006]. The two aforementioned specific parameters are mutually dependent (changing one may influence the effect of the other) and usually determined through a trial-and-error process or optimization. Cherkassky and Ma [2004] present some practical guidelines to determine these parameters. Optimal SVM parameter values are difficult to interpret and relate to characteristics of the response surface.

### 2.3.5. Neural Networks

[34] Feedforward artificial neural networks (ANNs) are highly flexible tools commonly used for function approximation. ANNs in this paper refer to multilayer perceptrons (MLPs), which are by far the most popular type of neural networks [Maier et al., 2010]. Development and application of ANNs involve multiple subjective decisions to be made by the user. Determination of the optimal structure of ANNs for a particular problem is probably the most important step in the design of ANN-based surrogates. ANN structural parameters/decisions include number of hidden layers, number of neurons in each hidden layer, and the type of transfer functions. Various methodologies have been developed to determine appropriate ANN structures for a given problem, including the methods based on the growing or pruning strategies [Reed, 1993] such as the methods presented in the work of Teoh et al. [2006] and Xu et al. [2006], methods based on the network geometrical
interpretation such as Xiang et al.’s [2005] method, and methods based on the Bayesian approaches such as the methods in the work of Kingston et al. [2008] and Vila et al. [2000]. However, these methods, each of which may result in an appropriate but different structure for a given problem, typically require extensive numerical analyses on the training data as they generally attempt to test different network structures in systematic ways. Despite these methodologies, trial-and-error is the approach to determine the number of hidden neurons in all ANN-based surrogate modeling studies listed in Table 1. Considering alternative types of neural network architectures beyond MLP (such as generalized regression neural network (GRNN) [Maier et al., 2010]) may provide another way to reduce or eliminate subjective ANN building decisions.

[35] ANNs with one sigmoidal hidden layer and the linear output layer have been proven capable of approximating any function with any desired accuracy provided that associated conditions are satisfied [Hornik et al., 1989; Leshno et al., 1993]. Although one hidden layer is adequate to enable neural networks to approximate any given function, some researchers argue that neural networks with more than one hidden layer may require fewer hidden neurons to approximate the same function. It is theoretically shown in the work of Tamura and Tateishi [1997] that to be an exact emulator (interpolating approximator, see also section 2.6.2), neural networks with two hidden layers require considerably fewer hidden neurons compared to neural networks with one hidden layer. However, developing an exact emulator is not usually the objective of neural network practitioners (except when used as response surface surrogates of deterministic computer simulation models) as the data used are usually noise-prone and the number of input-target sets is typically large. The need for exact emulation is still a case study specific determination. In addition, to be an exact emulator, excessively large ANN structures are required, whereas such networks would more likely fail in terms of generalization (perform poorly at unsampled regions of input space). Section 2.6.2 deals with exact emulation versus inexact emulation. From a more practical perspective, it is shown in the work of de Villiers and Barnard [1993] through extensive numerical experiments that single-hidden-layer neural networks are superior to networks with more than one hidden layer with the same level of complexity mainly due to the fact that the latter are more prone to fall into poor local minima in training.

[36] Neural network practitioners tend to use single-hidden-layer neural networks as, for example, 13 of 14 neural network applications in response surface surrogate modeling listed in Table 1 have used feed-forward neural networks with only one hidden layer; Liong et al. [2001] is the only study in Table 1 using more than one hidden layer. Single-hidden-layer neural networks form the approximation by combining m sigmoidal units (i.e., sigmoidal lines, planes, or hyperplanes in the 1-, 2-, or 3-and-more-dimensional problem space) where m is the number of hidden neurons. The number of parameters (weights and biases) of the single-hidden-layer neural networks is $m \times (2 + D) + 1$ where $D$ is the dimension of input space (i.e., number of input variables of the response surface surrogate). The optimal number of hidden neurons, $m$, is a function of shape and complexity of the underlying function [Xiang et al., 2005] as well as the training data availability [Razavi et al., 2012]. In the response surface surrogate modeling context, the form of the original function is often unclear, therefore, the number of data points available (i.e., design sites) for training, $p$, is the main factor involved in determining $m$. It is usually preferred that the number of ANN parameters be less (or much less) than $p$ as discussed in the work of Maier and Dandy [2000], although mathematically, there is no limitation when the number of parameters is higher than $p$. A possible strategy is to enlarge $m$ dynamically as more design sites become available. Generally, for a specific problem, there are multiple appropriate ANN structures, and for each structure, there are many appropriate sets of network weights and biases. The term “appropriate” here refers to the network structures and weight and bias values that can satisfactorily represent the training data. Neural network training (adjusting the network parameters) can be a time-consuming optimization process depending on the ANN structure and training method selected (see for example discussion on the work of Kourakos and Mantoglou [2009] in section 2.6.5). Second-order variations of backpropagation algorithms (i.e., quasi-Newton algorithm such as Levenberg-Marquardt) are the most computationally efficient ANN training methods [Hamm et al., 2007] the role of each weight and bias in forming the network response is typically unclear.

2.4. Metamodel-Enabled Analysis Frameworks

[37] Research efforts on metamodeling arising from water resources modeling are mainly focused on framework development for utilizing metamodels. Metamodel-enabled analysis frameworks in the literature can be categorized under four main general frameworks. The basic sequential framework and adaptive-recursive framework outlined in sections 2.4.1 and 2.4.2 are multipurpose frameworks (conceivably applicable in all sampling or search analyses), while the metamodel-embedded evolution framework in sections 2.4.3 is clearly limited to search analyses dependent on evolutionary optimization algorithms. The approximation uncertainty based framework outlined in section 2.4.4 is primarily used for search analyses but may also be applicable in some sampling studies.

2.4.1. Basic Sequential Framework

[38] Basic sequential framework (also called off-line) is the simplest metamodel-enabled analysis framework consisting of three main steps. It starts (Step 1) with design of experiment (DoE) through which a prespecified number of design sites over the feasible space are sampled and their corresponding objective function values are evaluated through the original function. In Step 2, a metamodel is globally fitted on the design set. Then in Step 3, the metamodel is fully substituted for the original model in performing the analysis of interest. In this step, a search or sampling algorithm is typically conducted on the metamodel. The result obtained from the metamodel is assumed to be the result of the same analysis with the original model; for example, in metamodel-enabled optimizers with the basic sequential framework, the optimal point found on the metamodel is typically evaluated by the original function and is considered as the optimal (or near optimal) solution to the original function. In some studies on optimization with metamodeling such as Broad et al. [2005] and Broad et al. [2010], at Step 3,
extra promising points on the metamodel found on the convergence trajectory of the optimization algorithm may also be evaluated by the original function. The size of DoE in Step 1 of this “off-line” framework is large compared to the size of initial DoEs in more advanced “online” metamodeling frameworks since almost the entire computational budget allocated to solve the problem is spent on Step 1 in the basic sequential framework. In this regard, the other frameworks (see sections 2.4.2–2.4.4) may be called “online” as they frequently update the metamodel when new data become available.

[38] According to Table 1, eleven out of 32 recent metamodeling studies use the basic sequential framework. As the sampler in Step 3, Khu and Werner [2003] and Zhang et al. [2009] use the GLUE uncertainty-based calibration algorithm, Schultz et al. [2004, 2006] use traditional Monte Carlo simulation for uncertainty analysis, Borgonovo et al. [2012] use Monte Carlo simulation for sensitivity analysis, and Bliznyuk et al. [2008] use a Markov Chain Monte Carlo (MCMC) sampler for uncertainty-based calibration. In all the other studies with the basic sequential framework, different optimization algorithms are used in Step 3. In the work of Liong et al. [2001] and Khu and Werner [2003], instead of fitting the metamodel over the design sites obtained through a formal DoE, which tends to assume uniformity, an optimization trial is conducted on the original function and the set of points evaluated over the convergence trajectory is used for metamodel fitting (see section 2.5 for discussion on whether an initial DoE is required). Instead of having a conventional DoE at Step 1, Bliznyuk et al. [2008] first locate a high posterior density region of the explanatory variable space by direct optimization on the original function and then fit a metamodel on the approximate high posterior region (local) rather than the entire space (global). Zou et al. [2007] contrast the basic sequential framework with the adaptive-recursive framework described in section 2.4.2.

[39] Although widely used, the basic sequential framework has potential failure modes arising from the fact that the metamodel, especially when developed off-line, is not necessarily a reasonably accurate representation of the original model in the regions of interest in the explanatory variable space. For example, in the optimization context, the global optimum of the metamodel found in Step 3 (i.e., the final solution returned by the framework) is very unlikely to be a local optimum of the original function. In other words, there is no guarantee that the returned solution is even located close to a stationary point of the original function. Figure 2 demonstrates a simple case in which the basic sequential framework fails and returns a solution close to a local maximum of the original function in a minimization problem. The metamodel in Figure 2 is a tuned kriging model rather than a conceptual example. The original function used in Figure 2 (and also used in Figures 3–7) was generated in this review to demonstrate the behavior of different surrogate modeling strategies. When the original function is simple (e.g., unimodal functions), the probability of such failures is minimal.

2.4.2. Adaptive-Recursive Framework

[40] Like the basic sequential framework, in Step 1, the adaptive-recursive framework starts with a DoE to design the initial set of design sites. In Step 2, a global/local metamodel is fitted on the set of design sites. In Step 3, a search or sampling algorithm is employed on the metamodel to identify the regions of interest in the explanatory variable space and screen out one or multiple points. When used for optimization, an optimization algorithm is typically used to find the near-optimal point (or multiple high-quality points) on the metamodel. The point(s) in the explanatory variable space obtained in Step 3, are evaluated by the original function and added to the set of design sites to update the metamodel. Step 2 and Step 3 are subsequently repeated many times to adaptively evolve the metamodel until convergence or stopping criteria are met. When the adaptive-recursive framework is used for optimization, the best point the framework finds (available in the final set of design sites) is considered as the optimal (or near-optimal) solution to the original function.

[41] Six out of the 32 studies listed in Table 1 apply procedures lying under the adaptive-recursive framework. Five studies utilize formal optimization algorithms in Step 3, but Castelletti et al. [2010] conduct a complete enumeration at this step since their combinatorial optimization problem has only three decision variables. Johnson and Rogers [2000] point out that the quality of solutions obtained through a metamodel-enabled optimizer is mostly controlled by metamodeling performance, not the search technique applied on the metamodel.

[42] The adaptive-recursive framework attempts to address the drawbacks of the basic sequential framework [Zou et al., 2007]. When used for optimization purposes, however, the adaptive-recursive framework is helpful at best for local optimization as reported by Jones [2001] who identified some possible cases where the adaptive-recursive framework may even fail to return a stationary point on the original function. Figure 3 which depicts possible behaviors of the adaptive-recursive framework, was partially inspired by the discussion in the work of Jones [2001]. As shown in Figure 3a, if the optimal point on the metamodel (found in Step 3) is located at a previously evaluated point on the original function (i.e., a point already existing in the set of design sites used in Step 2), the algorithm would stall as reevaluating and adding this point to the set of design
sites would not change the metamodel (may also cause mathematical problems in some function approximation methods such as kriging). Another similar failure mode that is less likely to occur is when the optimal point on the metamodel (found in Step 3) has the same objective function value in both the metamodel and the original function (see Figure 3a but assume there was no design site at the surrogate minimizer, which is on a nonstationary point of the original function); evaluating and adding this new point to the set of design sites would not have any effect on the updated metamodel. Although it seems very unlikely that Step 3 returns a point exactly the same as the points described in Figure 3a, returning new points lying in their close vicinity may also result in very little change in the updated metamodel and lead to the same problems. Some procedure is required in this framework to address these algorithm stalls and revive the search (one example procedure is in the work of Hemker et al. [2008]). Jones [2001] notes that one way to mitigate these algorithm stalls is to match the gradient of the response surface surrogate with the gradient of the original function at the sampled points (at least on the sample point at which the search stalls). However, as explained in section 2.3, the application of the “gradient-enhanced response surface surrogates” is nontrivial with practical limitations.

The adaptive-recursive framework when used for optimization can easily miss the global optimum of the original function [Gutmann, 2001; Jones, 2001; Schonlau, 1997; Sobester et al., 2005]. Although significance of missing the global optimum depends on a number of factors, Figure 3b depicts a case where the framework has found a local optimum, but as the metamodel is misleadingly inaccurate in the neighborhood of the global optimum, Step 3 would never return a point in the global region of attraction to be evaluated by the original model. The adaptive-recursive framework with gradient-enhanced response surface surrogates (see section 2.3) may only guarantee convergence to a stationary point on the original function, which may be a global/local optimum, plateau, or saddle point [Jones, 2001]. Figure 3c shows a case where the framework has converged to a point on a plateau on the original function at which the gradient of the metamodel matches the gradient of the original function. Notably, the surrogate functions in Figures 3a–3c represent the actual response of a kriging model developed for the purpose of this review and tuned for the given sets of design sites.

### 2.4.3. Metamodel-Embedded Evolution Framework

The metamodel-embedded evolution framework is a popular framework for optimization. Among the 32 studies on metamodeling listed in Table 1, eight studies utilize algorithms lying under this framework. The metamodel-embedded evolution framework shares some characteristics with the adaptive-recursive framework but with significant differences including: the metamodel-embedded evolution framework is inherently designed to be used with evolutionary (i.e., population-based) optimization algorithms, no formal DoE is typically involved at the beginning (except very few studies [e.g., see Regis and Shoemaker, 2004]), and decision criteria through which candidate solutions are selected for evaluation by the original function are different. In the metamodel-embedded evolution framework, a population-based optimization algorithm is employed and run initially on the original function for a few generations. All the individuals evaluated by the original function in the course of the first generations are then used as design sites for metamodel fitting. In the following generations, individuals are selectively evaluated by either the metamodel or
the original model. The metamodel is usually updated (refitted) a couple of times in the course of optimization as more design sites become available. Jin et al. [2002] adopt the metamodel-embedded evolution framework and formalize the concept of evolution control with two approaches: controlled individuals through which a subset of the individuals in the population at each generation (i.e., the best \( \eta \) individuals or \( \eta \) randomly selected individuals) are evaluated by the original function and controlled generations through which all the individuals in the population, but at selected generations, are evaluated by the original function. The parameters of metamodel-embedded evolution framework, such as \( \eta \) and the frequency of metamodel updating, can be adaptively changed in the course of optimization depending on the accuracy of the metamodel.

When an evolutionary algorithm is enabled with metamodels through the metamodel-embedded evolution framework, it typically exhibits less consistent behavior compared to when the same evolutionary algorithm is used without metamodeling [Jin et al., 2002]. This degradation in stability is expected as such an algorithm switches between two different response landscapes (the original function and the metamodel) while searching. A metamodel-enabled optimizer with this framework should satisfy at least two necessary conditions to become a global optimizer: (1) the evolutionary algorithm used should be a global optimizer, and (2) any solution (i.e., any individual in any generation) regardless of the approximate function value obtained by the metamodel should have the chance to be selected for evaluation through the original function. Thus, like the optimizers with the adaptive-recursive framework, metamodel-enabled optimizers with evolution control strategies that only evaluate best individuals (best in terms of approximate function values) through the original function are at best local optimizers. In such cases, the failure modes explained in Figure 3 for the adaptive-recursive framework (see section 2.4.2) may also apply to the metamodel-embedded evolution framework. Convergence properties of metamodel-embedded evolution framework with other evolution control strategies could likely overcome these failure modes.

Some variations of the Learnable Evolution Model (LEM) [Michalski, 2000] might also fall under the metamodel-embedded evolution framework. LEM is an attempt to improve upon the basic and inefficient Darwinian evolution operators by using machine learning tools. Unlike the conventional metamodeling practice which typically produces a continuous approximate response surface, LEM employs classifying techniques (e.g., decision tree learners) to discriminate promising/nonpromising solutions based on the search history. The classifier in LEM can involve domain-specific knowledge for rule induction. Jourdan et al. [2006] develop a multiobjective version of LEM, called LEMMO (LEM for multiobjective), for water distribution network design problems. LEMMO embeds a decision tree classifier within the NSGAII multiobjective optimization algorithm; the solutions generated by the evolution operators in NSGAII are first evaluated by the classifier and then modified if needed. di Pierro et al. [2009] compare LEMMO with ParEGO which is a multiobjective metamodel-enabled optimization algorithm (see section 2.5.6) on water distribution network design problems.

### 2.4.4. Approximation Uncertainty Based Framework

The approximation uncertainty based framework may be deemed as an extension to the adaptive-recursive framework designed for optimization. As explained in section 2.4.2, the adaptive-recursive framework relies solely on the approximate values from the response surface surrogate in the course of optimization. As the adaptive-recursive framework assumes these approximate values as true, it may easily miss the main region of attraction where the global optimum lies. This behavior is illustrated in section 2.4.2 and also reported in the work of Gutmann [2001], Jones [2001], Schonlau [1997], and Sobester et al. [2002]. Addressing this shortcoming, the approximation uncertainty based framework considers the uncertainties associated with the approximation. In this framework, the approximation value resulting from the response surface surrogate is deemed as approximation expected value and then a measure is utilized to quantify the associated approximation uncertainty. Such a measure is explicitly available in some function approximation techniques, e.g., in polynomials, kriging, Gaussian radial basis function models [Sobester et al., 2005], and smoothing spline ANOVA [Gu, 2002]. To provide the approximation uncertainty, these techniques assume that the deterministic response of a simulation model is a realization of a stochastic process. Unlike these techniques, which provide some statistical basis for approximation uncertainty, Regis and Shoemaker [2007a] propose a distance-based metric (i.e., the minimum distance from previously evaluated design sites) as a heuristic measure of the approximation uncertainty applicable to any given response surface surrogate. Other deterministic function approximation techniques may also provide measures of uncertainty when trained with Bayesian approaches; for example, Bayesian neural networks [Kingston et al., 2005] provide the variance of prediction. Bayesian learning typically involves a Markov Chain Monte Carlo procedure, which is more computationally demanding than conventional learning methods.

The approximation uncertainty based framework consists of the three steps outlined in section 2.4.2 with a major difference in Step 3 being, instead of optimizing the approximate response surface (i.e., the surface formed by approximation expected values) as in the adaptive-recursive framework, it optimizes a new surface function that is defined to emphasize the existence of approximation uncertainty. Different ways have been proposed in the literature to define such a surface function, each of which emphasizes the approximation uncertainty to a different extent. In this regard, the adaptive-recursive framework, in which the surface function totally ignores the approximation uncertainty, can be deemed an extreme case that typically yields a local optimum. The other extreme is to build and maximize a surface function solely representing a measure of approximation uncertainty (e.g., approximation variance available in kriging) over the explanatory variable space. In the approximation uncertainty based framework with such a surface function, Step 3 returns the point where the approximation uncertainty is the highest, and as such, subsequently repeating Step 2 and Step 3 in the framework would evolve a globally more accurate metamodel on the basis of a well-distributed set of design sites. Although globally convergent...
under some mild assumptions [Sobester et al., 2005], the framework solely using the approximation uncertainty would require impractically large number of original function evaluations, especially when the number of decision variables is more than a few.

[50] An effective uncertainty based surface function to be optimized in Step 3 combines the approximation expected value and the associated approximation uncertainty in a way that balances exploration (i.e., searching unexplored areas of high approximation uncertainty) and exploitation (i.e., fine-tuning a good quality solution by reducing the attention to approximation uncertainty) during the search. Most common methods to define the uncertainty based surface functions assume that the hypothetical stochastic process is normal with the expected value \( \hat{y} \) and standard deviation \( \sigma \) generated by the surrogate model for any given point in the explanatory variable space, \( x \). Figure 4a depicts the information typically available from a response surface surrogate that is capable of producing approximation uncertainty—this plot is the outcome of an actual experiment with kriging. There are two popular approaches to make use of such information in Step 3 of the approximation uncertainty based framework: (1) maximizing a new surface function representing the probability of improvement and (2) maximizing the so-called “expected improvement” surface function [Schonlau, 1997].

[51] Figures 4a–4c illustrate the concept of probability of improvement during optimization based on a real experiment with a tuned kriging model. As can be seen, \( f_{\text{min}} \) is the current best solution found so far (the design site with the minimum original function value), and as such, any function value which lies below \( f_{\text{min}} \) is an improvement. Thus, at any given point \( x \) in the explanatory variable space, a possible improvement, \( I \), and the probability of improvement, \( PI \), over the current best are given by:

\[
I = \begin{cases} 
 f_{\text{min}} - Y(x) & \text{if } Y < f_{\text{min}}, \\
 0 & \text{otherwise} 
\end{cases}, \tag{1}
\]

\[
PI = \Phi \left( \frac{f_{\text{min}} - \hat{y}(x)}{\sigma(x)} \right), \tag{2}
\]

where \( Y(x) \) is a possible function value at the point \( x \) being a random number following \( N(\hat{y}(x), \sigma(x)^2) \), and \( \Phi \) is the normal standard cumulative distribution function. Notably, if the probability of improvement over the current best is used as the surface function in the framework, the search will be highly local around the current best solution unless there is a point on the response surface surrogate having an estimated expected value, \( \hat{y} \), less than \( f_{\text{min}} \). To address this drawback, a desired target improvement, \( T \), which is smaller than \( f_{\text{min}} \) is assumed and then the probability that the original function value is equal or smaller than \( T \) would be:

\[
PI = \Phi \left( \frac{T - \hat{y}(x)}{\sigma(x)} \right). \tag{3}
\]

[52] The higher the desired improvement is (the smaller the value of \( T \) is), the more global the search is. As such, when desired improvement is assumed very small, the search would be very local typically around the current best until the standard error of the surrogate in that local area becomes very small. Very large desired improvements (\( T \) values much smaller than \( f_{\text{min}} \)) may force the algorithm to search excessively global resulting in very slow convergence. As Jones [2001] also points out, the performance of the algorithm based on the probability of improvement is highly sensitive to the choice of desired target, \( T \), and determining the appropriate value for a given problem is not trivial. To diminish the effect of this sensitivity, Jones [2001] presents a heuristic way to implement the probability of improvement approach based on multiple desired target values. Further details on the probability of improvement approach are available in the work of Jones [2001], Sasena et al. [2002], and Watson and Barnes [1995].

[53] The so-called expected improvement approach might be considered as a more advanced extension to the probability of improvement approach. Expected improvement is a measure that statistically quantifies how much improvement

---

**Figure 4.** (a) Probabilistic output of a response surface surrogate capable of producing approximation uncertainty assuming the deterministic response of simulation model follows a normal stochastic process, (b) approximation uncertainty function (\( AU \) or \( \sigma \)), (c) function representing probability of improvement (\( PI \)), and (d) function representing expected improvement (\( EI \)).
is expected if a given point is sampled to evaluate through the original function. Expected improvement at a given point \( x \), \( EI(x) \), is the expectation of improvement, \( I(x) \), (defined in equation (1)) for all possible \( Y(x) \) function values, which follow \( N(\hat{y}(x), \sigma(x)^2) \), and calculated by:

\[
EI = \left(f_{\min} - \hat{y}(x)\right)\Phi\left(\frac{f_{\min} - \hat{y}(x)}{\sigma(x)}\right) + \sigma(x)\phi\left(\frac{f_{\min} - \hat{y}(x)}{\sigma(x)}\right),
\]

where \( \Phi \) is standard normal cumulative distribution function, and \( \phi \) is standard normal probability density function. Interested readers are referred to Schonlau [1997] for derivation of equation 4. Figure 4d is a real example of the new surface function formed by the expected improvement approach. The EGO (Efficient Global Optimization) algorithm developed by Jones et al. [1998] is the most commonly used metamodel-enabled optimizer with the approximation uncertainty based framework that utilizes the expected improvement surface function.

[54] The approximation uncertainty based framework may utilize a different statistical approach than the approaches explained above to build a new surface function to use in Step 3. This approach, introduced by Gutmann [2001] and implemented first using Gaussian radial basis functions, hypothesizes about the location and the objective function value of the global optimum, and then evaluates the “credibility” of the hypothesis by calculating the likelihood of the response surface surrogate passing through the design sites conditioned to also passing through the hypothetical global optimum. As such, for any given hypothesis objective function value, a surface function can be formed over the entire variable space representing the credibility of having the hypothetical objective function value at different points. If kriging is used as the response surface surrogate, its parameters can also be optimized for any given point in the variable space to maximize the conditional likelihood [Jones, 2001]—different kriging parameter values are generated for different points in the variable space. Jones [2001] points out that the key benefit of this approach over the expected improvement approach is that the next candidate solution in this approach is not selected solely based on the parameters of the surrogate, which may be in substantial error when the initial design sites are sparse and/or deceptive. However, as even an estimate of the optimal objective function value is not known a priori in many real-world problems, hypothesizing about the optimal objective function value to be used in the framework may be nontrivial. Therefore in practice, the hypothetical optimal function value(s) should be heuristically defined and changed as the framework proceeds [Gutmann, 2001; Jones, 2001]. In addition, this approach is typically more computationally demanding than the expected improvement approach, especially when the surrogate parameters (e.g., kriging parameters) are also to be optimized to calculate the conditional likelihood for any given point. Regis and Shoemaker [2007b] propose a strategy to improve the method by Gutmann [2001] by controlling and increasing its local search ability.

[55] Among the studies listed in Table 1, three studies utilize the statistics-based approaches for the approximation uncertainty based framework: di Pierro et al. [2009] use the expected improvement approach through the ParEGO algorithm, Mugunthan et al. [2005] use the conditional likelihood approach developed by Gutmann [2001], and Bau and Mayer [2006] make use of the kriging error estimates to make termination decisions in their surrogate-enabled optimization framework. Regis and Shoemaker [2007a] also emphasize the need of considering the approximation uncertainty from a nonstatistical point of view and develop a metamodel-enabled optimizer involving a heuristic distance-based measure indirectly representing the uncertainty in approximation. Among the studies listed in Table 1, Mugunthan et al. [2005], Razavi et al. [2012], and Regis and Shoemaker [2007a, 2009] use the Regis and Shoemaker [2007a] metamodel-enabled optimizer.

[56] The metamodel-enabled optimizers with the approximation uncertainty based framework such as EGO are essentially for global optimization. They work very well when the general shape of the original function and its degree of smoothness are properly approximated typically as a result of a good initial DoE [Schonlau, 1997]. However, there are two possible drawbacks associated with this framework. Any new surface function based on the approximation uncertainty is highly multimodal, and as such finding its global optimum in Step 3 is not easy especially when the number of decision variables is large. To search for the global optimum in Step 3 of the framework, the original EGO algorithm [Jones et al., 1998] uses the branch-and-bound algorithm, the EGO-based metamodel-enabled optimizer developed in the work of Sobester et al. [2005] uses a multistart BFGS algorithm, and the algorithm by Regis and Shoemaker [2007a] uses a brute-force random sampler.

[57] The other possible drawback of this framework arises from the fact that the approximation uncertainty based framework relies extensively on a measure of approximation uncertainty to select the next candidate solution as it is correct; whereas such a measure is only an estimation of the approximation uncertainty suggesting that if poorly estimated, the approximation uncertainty would be very deceptive in guiding the search. The approximation uncertainty at a given point is mainly a function of the degree of nonsmoothness of the underlying (original) function being approximated through the design sites and then the distance from surrounding previously evaluated points (i.e., design sites) (note that the measure of uncertainty proposed in Regis and Shoemaker [2007a] is solely based on distance). As such, if the degree of nonsmoothness of the original function is poorly captured by poorly distributed initial design sites, the framework would result in a very slow convergence or even, in extreme cases, premature stalls. Figure 5 demonstrates one of our experiments with kriging where the degree of nonsmoothness of the original function is underestimated by the response surface surrogate due to a poor distribution of the initial design sites. In such a case, the framework tends to search locally around the minimum of the surrogate (similar to the behavior of the adaptive-recursive framework—see section 2.4.2) until a point better representing the degree of nonsmoothness of the original function is evaluated or until an exhaustive search is completed in this local region of attraction, which results in very small approximation uncertainty values at this region guiding the search to other regions. Jones [2001] identifies an extreme case where the framework stalls. There are different modifications in the literature to improve upon the approximation uncertainty based framework when
2.5.1. Local Optimization Versus Global Optimization

Search (Optimization) Frameworks

2.5. Design Considerations of Metamodel-Enabled Uncertainty based framework. On the global search capability of the approximation uncer-

tainly based framework. These modifications, include the generalized expected improvement concept. These

modifications, include the generalized expected improvement [Schonlau, 1997] and the weighted expected improvement

[Sobester et al., 2005], dynamically changing the emphasis on the global search capability of the approximation uncertainty based framework.

2.5. Design Considerations of Metamodel-Enabled Search (Optimization) Frameworks

2.5.1. Local Optimization Versus Global Optimization

[58] As illustrated in section 2.4.4, the metamodel-enabled optimizers under the approximation uncertainty based framework aim to improve upon other frameworks by recognizing the importance of approximation uncertainty for global optimization. In this framework, the chance of missing unexplored promising regions in the feasible space is reduced, and the evolved metamodels have more uniform global accuracies. Nonetheless, these gains might be at the expense of increasing the number of required original function evaluations and thus lowering the speed of convergence to a promising solution [Sobester et al., 2005]. In some cases when the original function is highly computationally expensive to evaluate, the maximum possible number of original function evaluations will be very limited (sometimes as small as 100–200). As such, practitioners have to be satisfied with adequate solutions that might not be very close to the global optimum. There is typically a trade-off between the global search (exploration) capability and the efficiency of a metamodel-enabled optimizer when computational budget is limited, especially in higher dimensional problems where the size of the feasible space is very large. In such cases, the adaptive-recursive framework might be more favorable as it may find an adequate local optimum within fewer original function evaluations than the approximation uncertainty based framework. This statement is consistent with Regis and Shoemaker’s [2007a] conclusion, after evaluating different metamodel-enabled optimizers involving approximation uncertainty, that “…more emphasis on local search is important when dealing with a very limited number of function evaluations on higher dimensional problems.” In this regard, Regis and Shoemaker [2007b] propose strategies to control the local search ability of two metamodel-enabled optimizers with the approximation uncertainty based framework.

2.5.2. Is Initial DoE Required?

[59] As demonstrated in section 2.2, most metamodel-enabled optimizers developed in the literature start with formal DoEs to generate an initial set of design sites that is uniformly distributed in the explanatory variable space. A well-distributed initial set helps the metamodel better represent the underlying (original) function. Nonetheless, there are metamodeling studies not involving formal DoEs. There are studies that only use previously evaluated points typically from previous optimization attempts to initially develop the metamodel. Furthermore, as pointed out in section 2.4.3, the studies that follow the metamodel-embedded evolution framework typically use the points evaluated in the early generations of the evolutionary algorithm to develop the first metamodels.

[60] We believe that an initial DoE is in fact required and a sufficiently large, well-distributed initial set of design sites to develop the metamodel is a key factor to success of a metamodeling practice. As demonstrated in section 2.4.4, metamodel-enabled optimizers can be easily deceived by metamodels fitted to poorly distributed design sites. Typically, the metamodel fitted on a set of points collected in a previous optimization attempt would be biased toward the already explored regions of the feasible space (probably containing local optima) and could be quite misleading; therefore, the unexplored regions would likely remain unexplored when optimizing on such metamodels [Jin et al., 2002; Regis and Shoemaker, 2007a; Yan and Minsker, 2006].

[61] In an evolutionary algorithm the initial population is usually uniformly distributed, but the individuals in the following generations are conditioned to the individuals in the previous generations. As such, in the metamodel-embedded evolution framework, the initial set of design sites to develop the metamodel, which is a collection of points evaluated in the initial and first few generations, may not be adequately distributed, and therefore the resulting metamodel may not...
have adequate global accuracy. In such a case, the metamodel that is only accurate in small regions might be completely misleading in the remaining parts that may contain the global optimum [Broad et al., 2005]. However, if a sufficiently large subset of well-distributed points exists in the initial set of design sites (formed by the first few generations in an evolutionary algorithm), the subset may act as if it is from a formal DoE.

2.5.3. Size of Initial DoE

The optimal size of an initial set of design sites is highly dependent on the shape and complexity of the original response surface as well as the computational budget available. The term “optimal” here reflects the fact that, for a given original response function, increasing the number of initial design sites would enhance the accuracy of fit (a positive effect), however, after some point (which is the optimum) this enhancement would be at the expense of unnecessarily increasing the computational budget having to be initially spent on DoEs while it could have been spent more effectively in the next steps (a negative effect). The framework through which the metamodel is used (see sections 2.4.1–2.4.4 for different frameworks) is also a factor affecting the optimal size of initial DoEs; for example, smaller initial DoEs may suffice when the metamodel-enabled optimizer puts more emphasis on approximation error (global accuracy). The optimal size also varies for different function approximation techniques based on their level of flexibility and conformability.

In practice, determination of the optimal size for a particular problem may only be possible through extensive numerical experiments. The only prior knowledge is usually the minimum limit on the number of design sites (see section 2.6.3), which is mathematically required to use a particular function approximation technique. There are some suggestions in the metamodeling literature on the size of initial DoEs when the approximation techniques are kriging and RBFs as summarized in the following. Jones et al. [1998] find that the approximate number of initial design sites required, \( p \), is

\[
p = 10D.
\]

where \( D \) is the number of dimensions of the explanatory variable space. Gutmann [2001] uses a two-level full factorial design, which samples the corners of the variable space, and as such the number of initial design sites used is

\[
p = 2^D,
\]

which becomes very large when the number of dimensions is more than a few. Regis and Shoemaker [2007a], based on the fact that kriging and RBFs with linear polynomials need at least \( D + 1 \) design sites to fit, suggest that

\[
p = 2(D + 1).
\]

[62] Razavi et al. [2012] relate the proper size of the initial DoE to the available computational budget for optimization and suggest the following equation for kriging and RBFs with linear polynomials:

\[
p = \max \{ 2(D + 1), 0.1n \}.
\]

where \( n \) is total number of original function evaluations (which typically accounts for almost all available computational budget) to be evaluated during the optimization. For relatively small \( n \) values, equation (8) is equivalent to equation (7), but when \( n \) becomes larger, in order to design a more detailed metamodel with a better global accuracy, \( 0.1n \) is used as the size of the initial DoE.

Sobester et al. [2005] conduct extensive numerical experiments with their proposed metamodel-enabled optimizer based on EGO (see section 2.4.4) on multiple test functions to study the effect of the size of initial DoEs on the algorithm performance. They suggest that

\[
p = 0.35n.
\]

Sobester et al. [2005] also note that the size of initial DoEs from equation (9) is an upper bound (safe choice) suitable for very deceptive, highly multimodal functions, and for simpler functions, smaller initial DoEs may be more appropriate. They also demonstrate that if the size of the initial DoE exceeds 0.60\( n \), the metamodel-enabled algorithm becomes inefficient. Overall, as the size of initial DoEs (and the total number of function evaluations) cannot typically be large when the original function is computationally expensive, there is no guarantee that the initial design sites are adequately well distributed to effectively represent the shape of the underlying function (e.g., estimate locations of the regions of attraction), particularly when it is very deceptive.

2.5.4. Metamodel Refitting Frequency

All algorithms utilizing metamodels except those using the basic sequential framework, aim to evolve the metamodels over time by refitting them over the newly evaluated points (the growing set of design sites). The ideal strategy is to refit the metamodel after each original function evaluation; this strategy is the basis of the adaptive-recursive (see section 2.4.2) and approximation uncertainty based (see section 2.4.4) frameworks. Metamodel-enabled optimizers with the metamodel-embedded evolution framework do not fundamentally need to refit the metamodel frequently (e.g., after each original function evaluation), although the higher the frequency, the more accurate the metamodel, and therefore, the better algorithm performance. Generally, the computational time required for refitting a metamodel (mostly nonlinearly) increases with an increase in the size of the set of design sites. The type of the function approximation technique used to build the metamodel is also a main factor in determining the appropriate refitting frequency. As such, the computational time required for the metamodel refitting substantially varies for different types of function approximation techniques and different data sizes and may become computationally demanding and even sometimes prohibitively long. Neural networks may suffer the most in this regard, as the neural network training process is typically computationally demanding relative to other alternatives even for small sets of design sites. Kriging refitting may also become computationally demanding for large numbers (more than a few hundreds) of design sites [Gano et al., 2006; Razavi et al., 2012]. The maximum likelihood estimation methodology for correlation parameter tuning is the main computational effort in the kriging (re)fitting procedure. Razavi et al. [2012] propose a two-level strategy for
refitting ANNs and kriging in which the first level (fast but not very accurate) is performed after each original function evaluation, but the frequency of performing the second level (complete refitting, computationally demanding but more accurate) is reduced through a function representing the complexity of the fitting problem as the number of design sites becomes larger. Refitting polynomials and RBFs with no correlation parameters is very fast even for moderately large sets (i.e., about 1000) of design sites. SVMs, as explained in section 2.3.4, also have two or more parameters that are determined by trial-and-error or direct optimization and as such their refitting might be time consuming. The appropriate metamodel refitting frequency for a given problem is also a function of the computational demand of the original computationally expensive model as the computational budget required for metamodel refitting may sometimes be negligible and easily justified when compared to the computational demands of the original computationally expensive models.

### 2.5.5. Optimization Constraint Function Surrogates

[67] In the water resources modeling literature, as well as the scientific literature in general, surrogate modeling has been used the most in an optimization context where the surrogate of a computationally intensive simulation model is used to approximate either the objective function or the constraints or both. The discussions in this paper mainly apply to surrogate modeling when emulating the objective functions and also when constraints are included in the objective function through penalty function approaches. When a binding constraint is approximated using a surrogate model, the approximation accuracy is highly important as it determines the feasibility/infeasibility of a solution.

[68] A number of the optimization studies in Table 1 [e.g., Broad et al., 2005; Broad et al., 2010; Koutrakos and Mantoglou, 2009; Yan and Minsker, 2006, 2011] apply surrogate models to approximate constraint functions and these constraint functions are built into the overall objective function via penalty functions. Overlooking the importance of constraint satisfaction and thus failing to take special precautions to ensure the metamodel-enabled optimizer yields a feasible solution could compromise the entire metamodeling procedure [Broad et al., 2005]. As a result, Broad et al. [2005] demonstrate an insightful three stage approach to deal with constraint function inaccuracies and part of this approach simply involves archiving the good quality solutions found in the course of optimization with the surrogate and then evaluating a set of these with the original model after optimization if it turns out that the final solution is infeasible. They also note the importance of training the surrogate model on both feasible and infeasible design sites. Yan and Minsker [2011] report for their ANN surrogate model of constraints that their penalty function parameters were determined by trial and error experiment. Although such a trial and error approach to penalty function parameters can be difficult to avoid, such experimentation with a metamodel-enabled optimizer will present incredible computational challenges. There are also different approaches in the broader research community to more accurately handle constraints with surrogates [e.g., Kim and Lee, 2010; Lee et al., 2007; Picheny et al., 2008; Viana et al., 2010]. The paper by Viana et al. [2010] nicely overviews these general approaches (designing conservative surrogates and adaptively improving surrogate accuracy near the boundary between feasible and infeasible solutions).

### 2.5.6. Multiobjective Optimization

[69] Surrogate models have been used in combination with a variety of multiobjective optimization algorithms to approximate the true Pareto-front within limited original model evaluations. Example metamodel-enabled multiobjective optimization algorithms are formed by (1) fitting response surface surrogate models to only one computationally expensive objective (or constraint) when other objectives are fast to run [e.g., Behzadian et al., 2009], (2) aggregating multiple objectives into one response function (e.g., by a weighting scheme) to be approximated by a single response surface surrogate [e.g., di Pierro et al., 2009; Knowles, 2006; Zhang et al., 2010], and (3) using multiple surrogate models for multiple objectives [e.g., Bau and Mayer, 2006; Keane, 2006; Li et al., 2008; Ponweiser et al., 2008]. The design considerations/limitations of surrogate modeling in single-objective optimization also typically apply to all metamodel-enabled multiobjective optimization algorithms; however, the algorithms following form 3 have additional considerations or limitations discussed below.

[70] The use of multiple surrogate models for multiple objectives would have the potential to increase the problems with inaccuracy and would definitely increase metamodeling time. The multiobjective optimization algorithms that utilize multiple surrogates commonly assume that the approximation errors (uncertainties) of these multiple surrogates are independent (no correlation) despite the fact that the objectives are typically conflicting [Wagner et al., 2010]. These considerations practically limit the metamodel-enabled optimization algorithms in applicability to problems with only a small number of objectives. The issue of multiple correlated outputs being approximated by surrogate models that is discussed in section 2.6.5 becomes relevant for metamodel-enabled multiobjective optimizers that approximate two or more objectives. Recent research by Bautista [2009] and Svenson [2011] address the dependencies between multiple objective functions when these functions are emulated with multivariate Gaussian processes.

[71] The metamodel-enabled frameworks outlined in sections 2.4.1-2.4.4 are all applicable to multiobjective optimization (see Table 1 for example applications). When using the basic sequential framework, at the end of Step 3, all approximate tradeoff solutions should be evaluated with the original computationally expensive objective functions to determine which of these solutions are actually nondominated (i.e., still tradeoff solutions). The Efficient Global Optimization (EGO) single-objective optimization algorithm [Jones et al., 1998] under the approximation uncertainty based framework, explained in section 2.4.4, has stimulated a great deal of research to extend the expected improvement concept to multiobjective optimization [Bautista, 2009; Ginsbourger et al., 2010; Jeong et al., 2005; Keane, 2006; Knowles, 2006; Ponweiser et al., 2008; Svenson, 2011; Wagner et al., 2010; Zhang et al., 2010]. ParEGO [Knowles, 2006], which utilizes EGO to optimize a single aggregated function (weighted sum with varying weights) of all the objective functions, and SMS-EGO [Ponweiser et al., 2008],
which develops multiple surrogates for multiple objectives, are among the most popular algorithms extending EGO to multiobjective optimization. Li et al. [2008] propose an approximation uncertainty based approach independent of EGO to account for uncertainties of multiple surrogate models for multiobjective optimization.

### 2.6. Limitations and Considerations of Response Surface Surrogates

[72] In addition to the considerations listed in section 2.5 that are specific to the design of response surface surrogate-enabled search frameworks, there are more general limitations and considerations that are relevant in any modeling analysis utilizing response surface surrogates. These limitations and considerations are discussed below in sections 2.6.1 to 2.6.4.

#### 2.6.1. High-Dimensional Problems

[73] Dimensionality is a major factor affecting the suitability of response surface surrogate modeling. Response surface surrogate modeling becomes less attractive or even infeasible when the number of explanatory variables is large. In such problems, the primary issue is that the minimum number of design sites required to develop some function approximation models can be excessively large. For example, to determine the coefficients of a second-order polynomial in a D-dimensional input space, at least \( p = (D + 1)(D + 2)/2 \) design sites are required; in a 25-variable input space, at least 351 design sites are required. Koch et al. [1999] demonstrate that in order to obtain a reasonably accurate second-order polynomial, the minimum number of design sites may not be sufficient and suggest that 4.5\( D \) design sites (1580 when \( D = 25 \)) are necessary, which might be well beyond the computational budget available when dealing with computationally expensive models. Note that this curse of dimensionality problem exists in all other function approximation models that are augmented by second-order polynomials (e.g., RBFs and kriging used in conjunction with second-order polynomials).

[74] Most importantly, high-dimensional problems have an extremely large search space. As such, the number of design sites required to reasonably cover the space becomes extremely large for higher number of variables. As an example, O’Hagan [2006] notes that 200 design sites in a 25-D space yield a very sparse coverage, but the same figure can result in a quite dense, adequate coverage for metamodeling in a 5-D space. As a result, the number of explanatory variables (decision variables, DVs, in optimization problems) in metamodel-enabled frameworks is typically not large. Among the metamodeling studies listed in Table 1, more than 65% of the metamodel applications are on functions having less than ten decision variables, and more than 85% have less than 20. Behzadian et al. [2009] and Broad et al. [2010] are the only studies reporting successful applications of metamodeling in relatively large-sized problems (with 50 and 49 decision variables, respectively). Notably in both studies, the number of original model evaluations is very large (10,000), and ANNs are used as metamodels to fit the very large sets of design sites. However, di Pierro et al. [2009] report an unsuccessful application of a metamodel-enabled optimizer (ParEGO by Knowles [2006], see section 2.5.6) on problems having 34 and 632 DVs—the other optimizer they used (LEMMO by Jourdan et al. [2006], see section 2.4.3) considerably outperformed ParEGO on both problems. They point out that they could not improve the final solution quality of ParEGO even with increasing the total number of original function evaluations.

Shan and Wang [2010b] survey existing strategies to tackling the problems associated with high-dimensional problems in optimization. These strategies, which are also typically applicable to metamodel-enabled optimization, include (1) screening aiming at identifying and removing less important decision variables [e.g., Ratto et al., 2007; Young and Ratto, 2009], (2) decomposition aiming to decompose the original problem into a set of smaller-scale subproblems [Shan and Wang, 2010b], and (3) space reduction being concerned with shrinking the feasible decision variable space by reducing the variable ranges to only focus on more attractive regions in optimization [Shan and Wang, 2004; Shin and Grandhi, 2001]. Notably, none of the above strategies is a complete remedy to the issues arising in metamodeling in the context of high-dimensional problems, and as outlined by Shan and Wang [2010b], each has its own limitations in applicability and usefulness. Shan and Wang [2010a] develop a new function approximation model that is computationally efficient for larger number of decision variables.

#### 2.6.2. Exact Emulators Versus Inexact Emulators

[76] A question that metamodel users need to address in any metamodeling practice is whether an exact fit (i.e., exact emulator) to the set of design sites or an approximate fit (i.e., inexact emulator), possibly with smoothing capabilities, is required. Exact emulation, also referred to as interpolation in numerical analysis, aims to construct a response surface surrogate representing the underlying function that goes through all design sites (i.e., exactly predicts all design sites). Kriging for computer experiments, RBFs, and Gaussian emulator machines [O’Hagan, 2006] are examples of exact emulators. Unlike exact emulators, there are emulation techniques that are inexact in that they produce a varying bias (deviations from the true values that are sometimes unpredictable) at different design sites. Polynomials, SVMs, MARS (multivariate adaptive regression splines), and ANNs are example inexact emulators generating non-interpolating emulator of the underlying function.

[77] Under certain (usually impractical) circumstances, inexact emulators may turn to exact emulators. For example, a polynomial can exactly reproduce all design sites when the degree of freedom of the polynomial regression is zero—in case where there are as many coefficients in the polynomial as there are design sites. SVMs can also behave as if they are (almost) exact emulators when the radius of the \( \varepsilon \)-insensitive tube is set very close to zero and the weight of the regularization term is set very small so that it becomes nondominant (see section 2.3.4 for the details of SVM parameters). It has been proven that single-hidden-layer ANNs are also able to exactly fit \( n \) design sites provided that there are \( n+1 \) neurons in the hidden layer [Tanaka and Tateishi, 1997]. ANNs with two hidden layers having \( (n/2) + 3 \) hidden neurons are also capable of acting as almost exact emulators [Tanaka and Tateishi, 1997]. Nevertheless, neither SVMs nor ANNs have been developed...
to apply as exact emulators and such applications would be impractical.

[78] SVMs have been fundamentally developed for inexact emulation with strong and direct smoothing capabilities. Although ANNs are inexact emulators, their smoothing properties are usually unclear to the user and very hard to manipulate [Razavi and Tolson, 2011]. Kriging with the so-called “Nugget effect” [Cressie, 1993] is also an inexact emulation technique with smoothing capabilities producing a statistics-based bias at design sites. Any smoothing capability usually has an associated tuning parameter that controls the extent of smoothing.

[79] There are two general types of problems involving function approximation models: physical experiments and computer experiments. There may exist substantial random errors in physical experiments due to different error sources, whereas computer simulation models are usually deterministic (noise-free), which means observations generated by a computer model experiment with the same set of inputs are identical. The inexact emulators are more suitable for physical experiments than computer experiments as the usual objective is to have an approximation that is insensitive (or less sensitive) to noise. An example application where inexact emulation is recommended is data-driven hydrologic and rainfall-runoff modeling as, e.g., neural networks have been extensively used in this context. Conversely, exact emulators are usually more advisable when approximating the deterministic response of a computer model.

[80] Figure 6 presents two real example experiments with inexact emulators to show how the noninterpolating behavior can be quite misleading in optimization especially in the vicinity of regions of attraction. Figure 6a shows a case where the set of design sites is relatively well distributed and includes a point very close to a local optimum, however, the quadratic polynomial fitted on this set is quite misleading and returns a point (surrogate function minimizer) on the plateau while ignoring the already found local region of attraction; evaluating the surrogate function minimizer and refitting would not noticeably change the polynomial. Figure 6b demonstrates a similar case where a single-hidden-layer neural network with 5 hidden neurons is trained to be used as the surrogate. In this case, the set of design sites is intentionally very well distributed such that there are design sites located at both regions of attraction (one global and one local). As can be seen, in our experiment with this neural network, the local region of attraction (local mode on the left) is easily ignored despite the fact that there is a design site very close to the local minimum, and second the location of the global region is misinterpreted. We believe that such a misleading behavior is not unlikely as we easily observed it in this simple experiment after a few trial-and-errors in network initialization and training. Notably, evaluating and adding the surrogate function minimizer to the set of design sites and refitting might not properly change the shape of the surrogate.

[81] In search-based metamodel-enabled analyses, it might be beneficial to have a smooth inexact emulator generating a surface passing smoothly across the design sites in the regions of the explanatory variable space with inferior quality as it may lead the search smoothly to the main regions of attraction. In contrast, inexact emulators can be very misleading in the regions of attraction (i.e., regions containing good quality local/global optima) where even marginal superiority of candidate solutions over each other is very important and the key to continue the search. Combining the two behaviors (i.e., exact emulation and inexact emulation) and adaptively switching from one to the other may appear promising, although how to implement this adaptive switching using the common function approximation techniques is not trivial.

[82] Exact emulation would seem to be the most appropriate way of approximating the deterministic response of computer simulation models. To our knowledge, the issues and shortcomings of inexact emulation for response surface modeling as described above have not been fully addressed in the literature, although the problems associated have been acknowledged to some extent in some publications, e.g., in the work of Sacks et al. [1989], Jones [2001], and Razavi et al. [2012]. For example, Jones [2001] points out that inexact emulators are unreliable because they might not sufficiently capture the shape of the deterministic underlying function. In particular, it is not clear to us from the water resources literature on surrogates for constraints, why the inexact emulation of a penalty function (which for large and sometimes continuous regions of decision space can be zero) is preferred or selected over an exact emulator.
2.6.3. Limits on Number of Design Sites

The number of design sites used for metamodel fitting can be a limiting factor affecting the suitability of a function approximation technique for a specific problem. The appropriate range (lower and upper bounds) for this number varies from one function approximation technique to another. Generally, the more design sites used for metamodel fitting, the higher the computational expense incurred in the fitting process. The computational expense associated with metamodel development and fitting should be taken into account in any metamodeling application. This expense may be limiting and directly affect the suitability of a function approximation technique for a specific problem especially when the total number of original model evaluations (and accordingly the number of design sites) in a metamodel-enabled application is relatively large.

The function approximation techniques utilizing basis (correlation) functions, such as kriging, RBFs, SVMs, and Gaussian Emulator Machine (GEM), are the most prone to the limitations arising from the large numbers of design sites. In these techniques, except for SVMs, the number of correlation functions is typically as many as the number of design sites, and as such, their structures and the computations associated for large sets of design sites become excessively large. GEM may suffer the most in this regard as the maximum number of design sites utilized in GEM applications in the literature is only 400 [Ratto et al., 2007]. Kriging has also limited applicability when the number of design sites is large, mostly because determining the kriging correlation parameters through the maximum likelihood estimation methodology can become computationally demanding for large sets. Practical numbers of design sites in kriging applications are typically less than a few thousands.

RBFs and SVMs can handle larger numbers of design sites. Least squares methods can efficiently fit RBFs even on large sets of design sites. SVMs are also capable of more efficiently handling larger numbers of design sites as the operator associated with the design site vectors in the SVM formulation is dot product [Yu et al., 2006]. However, both RBFs and SVMs may involve a relatively computationally demanding parameter tuning process for the correlation parameters and the other two specific parameters of SVMs.

Unlike the correlation functions existing in GEM, kriging, RBFs and SVMs, each of which only responds to a small region in the input space close to the corresponding design site, ANNs consist of sigmoidal units each of which is associated with a hidden neuron having an active part over a large domain of the input space. As such, even for large sets of design sites, ANNs may have relatively limited numbers of hidden neurons forming reasonably sized ANN structures. There are ANN applications for very large sets of design sites; for example, Broad et al. [2005] use 10,000 design sites in ANN fitting and Behzadian et al. [2009] utilize ANNs in the adaptive-recursive framework with 590,000 and 2,098,400 original function evaluations.

As opposed to the above function approximation techniques, which consist of a number of locally active building blocks, polynomials have a single global form covering the entire input space. As such, polynomial structure does not expand as the number of design sites increases. Polynomials can be fitted very fast even over very large sets of design sites. Similar to polynomials, the structure and complexity of multivariate adaptive regression splines (MARS) is not a function of the number of design sites, and instead, it is a function of the shape and complexity of the underlying function represented by the design sites. MARS builds multiple piecewise linear and nonlinear regression models (basis functions) to emulate the underlying function in the design sites [Friedman, 1991], and its main computational effort is to search over a variety of combinations by first adding the basis functions to the model (forward pass) and then extensively prunes the model (backward pass) to find a parsimonious model with a satisfactory generalization ability.

The minimum number of design sites required to develop a metamodel also varies for different function approximation techniques. For a polynomial, the minimum number equals the number of coefficients existing in the polynomial. Thus, for zeroth-, first- and second-order polynomials the minimum numbers are 1, $D + 1$ and $(D + 1) (D + 2)/2$, respectively, where $D$ is the dimension of the input space. As stated in section 2.6.2, when using these minimum numbers, the polynomials would act as exact emulator (i.e., zero degree of freedom). The minimum number of design sites in kriging and RBFs depends on the polynomials by which they are augmented; as zeroth- and first-order polynomials are commonly used in conjunction with kriging and RBFs, the minimum number of design sites in these techniques can be very small (e.g., 11 for $D = 10$ when augmented with first-order polynomials). GEM, SVM, and MARS also require reasonably small sets of design sites. For ANNs, although mathematically there is not any minimum limit for the number of design sites, it is commonly accepted that neural networks require relatively larger sets of design sites to be properly trained. The studies listed in Table 1 are consistent with this fact as the minimum number of initial design sites for neural network training in these studies is 150 [from Zou et al., 2007] and the second smallest number is 300 [from Yan and Minsker, 2006], while this number can be as small as 20–25 when the RBFs are acting as metamodels [from Regis and Shoemaker, 2007a].

2.6.4. Validation and Overfitting

Validation may be an important step in developing a response surface surrogate and reflects how the model performs in terms of generalizability. When a function approximation model exhibits a good fit to the design sites (i.e., zero error for exact emulators and satisfactorily small errors for inexact emulators), a validation measure is also required to ensure that the model performs consistently for unseen areas in the model input space. Cross validation strategies, such as $k$-fold cross validation and leave-one-out cross validation, are the commonly used means of validation of response surface surrogates [Wang and Shan, 2007], particularly when the set of design sites is not large. The
importance of validation differs for different approximation techniques. For example, the process of developing the polynomials, RBFs, or kriging approximation models is less dependent on validation as there are studies utilizing them without conducting a validation step; whereas, validation is an inseparable step in developing SVMs and ANNs. Bastos and O’Hagan [2009] claim that there has been little research on validating emulators before using them (i.e., in a surrogate-enabled analysis framework). While this statement, is accurate for some emulators (in particular Gaussian process emulators studied in the work of Bastos and O’Hagan [2009]), it is much more accurate when emulators such as SVMs and ANNs are considered. The studies in Table 1 that do not utilize SVMs or ANNs as response surface surrogates, also show little in the way of metamodel validation (although most do attempt to demonstrate the utility of the overall surrogate-enabled analysis). Bastos and O’Hagan [2009] propose some diagnostics to validate Gaussian process emulators.

[90] Overfitting (overtraining) degrades the generalization ability of approximation models. All approximation models are prone to overfitting. In statistics, overfitting is usually described as when the model fits the noise existing in the data dominantly rather than the underlying function. However, as discussed in the work of Razavi et al. [2012], surrogate models fitted on noise-free data (e.g., data generated from deterministic computer models) are also prone to overfitting, because there is another factor affecting the risk of overfitting, which is the conformability of the model structure with the shape of the available data. Overfitting due to conformability is more likely when the approximation model has a large degree of freedom (is overparameterized) compared to the amount of available data. Curve-fitting (regression analysis) practices are typically less prone to the negative effects of this factor, especially for low orders, because they have a global prespecified model form covering the entire input variable space. But in highly flexible approximation models, including ANNs, the problem associated with the conformability factor can be substantial.

[91] Neural networks are highly prone to the risk of overfitting. However, not all of the studies involving neural networks listed in Table 1 pay attention to overfitting/ generalizability. Seven studies listed in Table 1 [Broad et al., 2005; Broad et al., 2010; Johnson and Rogers, 2000; Khu and Werner, 2003; Razavi et al., 2012; Zou et al., 2007, 2009] apply the early stopping approach [Beale et al., 2010] to avoid overtraining, and one study [Kourakos and Mantoglou, 2009] applies the Bayesian regularization procedure [Foresee and Hagan, 1997]. The main problem with early stopping is that the available design sites have to be split into a training set, a testing set, and sometimes a validation set resulting in fewer data available to train ANNs. In contrast, the Bayesian regularization procedure does not require this splitting of design sites—all can be used for training. Razavi and Tolson [2011] recently proposed a neural network regularization measure that also does not require a testing/validation set.

[92] Overfitting due to the conformability factor may also occur in the function approximation techniques that are based on basis functions such as kriging and RBFs. However, the risk and extent of overfitting in kriging and RBFs is typically less compared to ANNs. The risk of overfitting is higher when there are very few design sites relative to the number of kriging and RBF parameters to be tuned. Note that, for example, a kriging model with Gaussian correlation functions has D correlation function parameters each of which is associated with one dimension in the D-dimensional input space and an RBF model with thin-plate splines has no correlation parameter to tune; as such, the number of parameters in these approximation models is typically small compared to the number of available design sites. As overfitting in kriging is not a major challenge, it has not been directly addressed in most kriging studies. To mitigate the possible overfitting problem in kriging, Welch et al. [1992] propose to initially keep all the correlation function parameters the same in all input space dimensions in the maximum likelihood estimation process, and then relax them one-by-one to identify the ones resulting in higher increase in the likelihood function and only let them be different.

2.6.5. Emulating Multiple Outputs or Multiple Functions

[93] The literature reviewed in Table 1 shows that even though the vast majority of response surface surrogate studies involve a simulation model with temporally and spatially varying outputs, the required number of model outputs or number of output functions (e.g., calibration error metrics) to approximate with surrogates is typically limited to only handful of outputs/functions (often just one). Recently, Bayesian emulation techniques have appeared that are tailored to approximate a time series of output (e.g., emulating a dynamic simulator in the work of Conti and O’Hagan [2010]). According to Conti and O’Hagan [2010], there are three approaches to emulating dynamic simulators: (1) multioutput emulators that are unique because they account for correlations among outputs, (2) multiple single-output emulators, and (3) time input emulators that use time as an auxiliary input. They conclude that the multiple single-output emulator approach is inappropriate because of its failure to account for temporal correlations and believe that multioutput emulators should eventually lead to the successful emulation of time series outputs. Fricker et al. [2010] also propose multioutput emulators considering correlation between multiple outputs based on multivariate Gaussian processes.

[94] In terms of the common function approximation techniques in Table 1, since ANNs have the ability to predict multiple outputs simultaneously, ANNs are a type of multioutput emulators. None of the other function approximation techniques reviewed in detail in section 2.3.1–2.3.4 can directly act as multioutput emulators. Thus, a single ANN model of multiple correlated outputs should conceptually be able to account for these correlations among outputs. Based on the work by Conti and O’Hagan [2010], we believe that the ability to account for correlations among outputs that are significantly correlated (even multiple output functions such as two model calibration objective functions) in the response surface surrogate should conceptually lead to increased surrogate accuracy. However, there are multiple studies demonstrating the need for multiple ANN surrogates to model multiple outputs (e.g., Kourakos and Mantoglou [2009], Yan and Minsker [2011], and Broad et al. [2005]). Yan and Minsker [2011] approximate six
outputs with three independent ANN surrogate models while Broad et al. [2005] model each output of interest with independent ANNs. Kourakos and Mantoglou [2009] utilize ANNs to approximate 34 outputs and they explain how their single ANN to model all outputs would lead to a practically infeasible ANN training procedure as nearly 2400 ANN parameters were to be specified. Instead they built and trained 34 modular subnetworks to circumvent this computational bottleneck in ANN training, assuming that the correlations between the outputs are negligible (justified based on the physics of their case study). Further studies could investigate scenarios where multiple-output ANNs are beneficial and to determine all the reasons they can fail relative to multiple single output ANNs.

3. **Lower-Fidelity Physically Based Surrogates**

[95] In contrast to response surface surrogates, which are data-driven techniques for approximating the response surface of high-fidelity (original) models based on a limited number of original model evaluations, lower-fidelity physically based surrogates are essentially cheaper-to-run alternative simulation models that are less faithful to the system of interest. For any real-world system, there may exist several simulation models with different levels of fidelity (accuracy). A high-fidelity model refers to the most accurate and as such the most desirable model available to users. As the high-fidelity models may typically be computationally intensive, there are frameworks concerned with efficiently utilizing the high-fidelity models in conjunction with lower-fidelity models (as surrogates of high-fidelity models) to enhance the overall computational efficiency; these surrogate modeling frameworks when applied in the field of optimization are also referred to as “multifidelity” or “variable-fidelity” optimization [Forrester et al., 2007; Gano et al., 2006; Leary et al., 2003; Madsen and Langthjem, 2001; Sun et al., 2010]. In some publications, low-fidelity models are called “coarse” models, and high-fidelity models are called “fine” models (e.g., in the work of Bandler et al. [1994]). As a simple example, a numerical model with very small numerical time steps may be deemed a high-fidelity model and its corresponding low-fidelity model may be one with larger numerical time steps. In this paper, we often refer to “lower-fidelity physically based surrogates” as “lower-fidelity surrogates” for simplicity.

[96] Lower-fidelity surrogates have two immediate advantages over the response surface surrogates: (1) they are expected to better emulate the unexplored regions in the explanatory variable (input) space (i.e., regions far from the previously evaluated points with the high-fidelity model) and as such perform more reliably in extrapolation, and (2) they avoid or minimize the problems associated with high-dimensional problems (see section 2.6.1), as they use domain-specific knowledge. There is a main assumption behind any lower-fidelity surrogate modeling practice: high-fidelity and low-fidelity models share the basic features and are correlated in some way [Kennedy and O’Hagan, 2000]. As such, the response of the low-fidelity model for a given input vector is expected to be reasonably close to the response of the high-fidelity model for the corresponding input vector in the high-fidelity model input space. This closeness enables the lower-fidelity model to relatively reliably predict the performance of the high-fidelity model in unexplored regions in the variable space. If this assumption is violated, the surrogate modeling framework would not work or the gains would be minimal.

[97] There are multiple strategies to reduce the number of expensive original model evaluations when a lower-fidelity model is available. The immediate strategies include the use of the lower-fidelity model first to reduce the variables’ feasible space and/or to identify unimportant variables to reduce the problem dimensionality [Madsen and Langthjem, 2001]. Most studies utilizing lower-fidelity models are concerned with developing optimization strategies where low- and high-fidelity models are adaptively chosen to be evaluated in the course of optimization [Forrester et al., 2007; Gano et al., 2006; Huang et al., 2006; Leary et al., 2003; Viana et al., 2009]. There are also lower-fidelity surrogate applications in uncertainty analysis [Allaire, 2009; Kennedy and O’Hagan, 2000].

[98] Lower-fidelity surrogate modeling has only very recently started to gain popularity in the water resources literature. Although it is a well-established area of research in the broader research community, formal terminologies and common methods available for lower-fidelity surrogate modeling in other disciplines seem to be largely unused in water resources literature. This section first reviews and categorizes the research efforts for lower-fidelity surrogate modeling in the broader research community and then reports the research efforts accomplished in water resources literature.

3.1. **Types of Lower-Fidelity Physically Based Models**

[99] Depending on the original model type, at least three different general classes of strategies may be used to yield lower-fidelity models. In the first class of strategies, the lower-fidelity models are fundamentally the same as the original models but with reduced numerical accuracy. For example, in numerical simulation models of partial differential equations, a lower-fidelity model can be a variation of the original model but with larger (coarser) spatial/temporal grid size [Leary et al., 2003; Madsen and Langthjem, 2001; Thokala and Martins, 2007]. Finite element models with simpler basis functions can also be a low-fidelity model of an original model involving more complex basis functions. Whenever applicable, lower-fidelity models can be essentially the same as the original model but with less strict numerical convergence tolerances. Forrester et al. [2006] employ a partially converged CFD model as a lower-fidelity surrogate of a fully converged (original) model.

[100] A second class of strategies to derive lower-fidelity models involves model-driven approximations of the original models using model order reduction (MOR) techniques [Gugercin and Antoulas, 2000; Rewienski and White, 2006; Willcox and Megretski, 2005]. MOR aims to reduce the complexity of models by deriving substitute approximations of the original complex equations involved in the original model. These substitute approximations are systematically obtained by rigorous mathematical techniques without the need of knowing the underlying system.

[101] In the third class of strategies, lower-fidelity models can be simpler models of the real-world system of interest in which some physics modeled by the high-fidelity model is ignored or approximated. Strategies such as
considering simpler geometry and/or boundary conditions in the model, utilizing a lumped-parameter model in lieu of a distributed model, and utilizing a two-dimensional model instead of a three-dimensional model lie under this class. For example in fluid dynamics, numerical models solving Navier-Stokes equations are the highest-fidelity and the most expensive models, models based on Euler equations are the lower-fidelity and less expensive models, and analytical or empirical formulations are the lowest-fidelity and cheapest models [Alexandrov and Lewis, 2001; Simpson et al., 2008; Thokala and Martins, 2007]. Note that for any real-world system, there may be a hierarchy of models with different levels of fidelity.

[102] Thokala and Martins [2007] conduct multiple experiments utilizing different types of lower-fidelity models and conclude that the lower-fidelity models that share the same physical components with the original models (i.e., lower-fidelity models lying under the first and second classes) are more successful than those having different/simplified physical bases (i.e., lower-fidelity models lying under the third class), as no correction (see section 3.2.1 for definition of correction functions) can compensate for ignoring a component of the physical characteristics of a system.

3.2. Lower-Fidelity Model Enabled Analysis Frameworks

3.2.1. Variable-Fidelity Models With Identical Variable Space

[103] Models with different levels of fidelity may be defined over the same variable/parameter space. There are multiple frameworks selectively utilizing lower-fidelity models as substitutes of the original models to reduce the number of expensive evaluations of the original model. These frameworks have mostly arisen from the optimization context but have also applied for other purposes including uncertainty analysis. The main challenge to be addressed in these frameworks is that the response landscapes of the original and lower-fidelity models are somewhat different. Figure 7 presents an illustrative hypothetical example of high- and low-fidelity response landscapes in a one-dimensional space. As can be seen, there are discrepancies between the two response landscapes; the low-fidelity function underestimates the response on the left part of the plot and overestimates in most of the right part. Moreover, both functions have two regions of attractors (modes), but the global minimizer of the low-fidelity function coincides with the local minimizer of the high-fidelity function which is far from the global optimum.

[104] There are three main independent (but sometimes complimentary) approaches to formally address the discrepancies between low- and high-fidelity models: correction functions, space-mapping, and hybrid strategies. The most common approach is to build a correction function to correct the response landscape of the lower-fidelity model and align it with the response landscape of the original model. This process in the multifidelity modeling literature is referred to as correction, tuning, scaling, or alignment. Suppose that $f_h(x)$ and $f_l(x)$ represent the response surfaces of the high- and low-fidelity models of the real-world system of interest, respectively; the two general strategies for defining a correction function are the additive approach [Gano et al., 2006; Leary et al., 2003; Viana et al., 2009] in equation (10) and the multiplicative approach [Alexandrov and Lewis, 2001; Madsen and Langthjem, 2001; Thokala and Martins, 2007] in equation (11):

\[ g_a(x) = f_h(x) - f_l(x), \]  

\[ g_m(x) = f_h(x)/f_l(x), \]  

where $g_a(x)$ is the additive correction function directly emulating the discrepancies between the high- and low-fidelity response surfaces and $g_m(x)$ is the multiplicative correction function. As the exact form of the correction function is typically unknown for any given problem, an approximate correction function is to be built by a limited number of high- and low-fidelity model evaluations. Then, the surrogate model can be

\[ f(x) = f_l(x) + g_a(x), \]  

\[ f(x) = f_l(x)g_m(x), \]

where $g_a(x)$ and $g_m(x)$ are the approximate correction functions that are designed to correct the low-fidelity model response by offsetting (additive) and scaling (multiplicative), respectively. Notably, the multiplicative form is prone to ill-conditioning when the model response approaches zero; to avoid this ill-conditioning, a constant can be added to both the numerator and denominator of equation (11). Eldred et al. [2004] demonstrate the superiority of the additive form over the multiplicative form across multiple test problems.

[105] The process of developing the approximate correction function is analogous to response surface surrogate modeling; however, the correction function is supposedly less complex than typical response surface surrogates, as the response surface of a lower-fidelity model is supposed to be reasonably close to the response surface of the original model. Different approaches or tools have been proposed to develop the approximate correction function. Linear regression [Alexandrov and Lewis, 2001; Madsen and Langthjem, 2001; Vitali et al., 2002] and quadratic
polynomials [Eldred et al., 2004; Sun et al., 2010; Viana et al., 2009] are two common, simple approaches to build the correction functions. More flexible function approximation models have also been used for this purpose, including kriging [Gano et al., 2006] and neural networks [Leary et al., 2003].

[106] Note that limitations and considerations raised in section 2.6 for response surface surrogates may also hold when using complex function approximation models to build the approximate correction functions. However, the limitations of response surface surrogates used in this context for high dimensional problems (see section 2.6.1) are not as important. This is because the correction function for a good quality lower-fidelity surrogate is only of secondary importance (it adjusts the lower-fidelity model output). As a result, less complex approximate correction functions may be more desirable in practice. Building correction functions that are correlated for multiple outputs is similarly not as important (see section 2.6.5).

[107] The general correction-function-based framework utilizing lower-fidelity surrogates is as follows: the framework, in Step 1, starts with an initial DoE to generate sample points and then evaluates them by both the original and lower-fidelity models. In Step 2, a global correction function is developed to emulate the discrepancies (errors) between the responses of the original and lower-fidelity models at the sampled points. In Step 3, a search or sampling is applied on the corrected response surface of the lower-fidelity model to identify the regions of interest in the explanatory variable space and screen out one or multiple points. In cases where the search algorithm is for optimization, this step returns the optimal/near-optimal point (or multiple high-quality points) of the corrected response surface of the lower-fidelity model. In Step 4, the candidate points from Step 3 are evaluated by the original function. If needed, the framework goes back to Step 2 to modify the correction function and repeat the analyses in Step 3.

[108] Trust-region approaches for optimization have also been applied in correction-function-based framework [Alexandrov and Lewis, 2001; Eldred et al., 2004; Robinson et al., 2006]. In such a framework, an initial DoE is not required and the framework may start with any (but desirable a good quality) initial solution (Step 1). The initial trust region size is also specified in this step. In Step 2, the current solution is evaluated by both original and lower-fidelity models. In Step 3, the correction function is locally fitted around the current (best) solution. In Step 4, the corrected lower-fidelity response surface is optimized within the trust region (centered at the current best solution) and the best solution found is evaluated by the original model. In Step 5, depending on how close this high-fidelity response is to the low-fidelity response, the trust region is expanded, remains the same, or is reduced. Steps 3 through 5 are repeated until convergence or stopping criteria are met. The trust-region based framework is provably convergent to a local optimum of the original model response surface if the corrected lower-fidelity response surface is at least first-order accurate at the center of the trust region [Robinson et al., 2006]. Eldred et al. [2004] demonstrate that second-order accurate corrections can lead to more desirable convergence characteristics.

[109] The second main approach to tackle the discrepancies between low- and high-fidelity response surfaces is the so-called space mapping approach. Initially introduced by Bandler et al. [1994] for optimization purposes, space mapping aims to locally establish a relationship between the original model variables and the lower-fidelity model variables. By definition, space mapping can be used to make use of any sufficiently faithful lower-fidelity model even if it is defined on a different variable space (see section 3.2.2). To establish a space mapping relationship, multiple points on the original response surface and their corresponding points on the lower-fidelity response surface are required. For any given point \( x \) in the original variable space, a corresponding point \( \hat{x} \) in the lower-fidelity variable space is defined as the point where \( f_h(x) \) is equal (or reasonably close) to \( f_h(\hat{x}) \). To find each point required in the lower-fidelity variable space, one optimization subproblem is to be solved with the objective of minimizing \( |f_h(\hat{x}) - f_h(x)| \) by varying \( \hat{x} \). Notably, there may exist multiple points \( \hat{x} \) having the same lower-fidelity response value equal to \( f_h(x) \) leading to the failure of space mapping. Many approaches have been proposed to address this problem of nonuniqueness [Bakr et al., 1999; Bandler et al., 1996; Bandler et al., 2004]. Once the corresponding points in the two spaces are available, different linear or nonlinear functions may be used to relate the two spaces by fitting over these points [Bandler et al., 2004]. Then any solution in the lower-fidelity variable space obtained in analyses with the lower-fidelity model can be mapped to the original variable space. The space mapping relationships can be updated as the algorithm progresses. As many optimization subproblems are to be solved on the lower-fidelity model to adaptively establish/update the mapping relationships, the total number of lower-fidelity model evaluations is relatively high. As such, if the low-fidelity model is not much cheaper than the original model, space mapping would not be computationally feasible.

[110] There are hybrid strategies following a third general approach to make use of lower-fidelity models jointly with original models to build response surface surrogates. These strategies may be used with any of the frameworks utilizing response surface surrogates (detailed in section 2.4) with the main difference that there are (at least) two sets of design sites with different levels of fidelity. These two sets of design sites are used to either build a single response surface surrogate formed by the sets or two response surface surrogates representing the two sets independently. Forrester et al. [2007] develop a single response surface surrogate using cokriging, which is an exact emulator on the high-fidelity design sites and an inexact emulator on the lower-fidelity design sites—such a response surface surrogate can capture the exact behavior of the underlying function where high-fidelity design sites are available in the variable space and only extract the trends and curvatures in the unexplored regions from the cheaply available lower-fidelity design sites that are far from the high-fidelity design sites. Leary et al. [2003] propose a heuristic way to incorporate the lower-fidelity model response into ANN and kriging-based response surface surrogates. Huang et al. [2006] develop a methodology to incorporate the data obtained by a lower-fidelity model to enhance the EGO algorithm (see section 2.4.4 for EGO). Vitali et al. [2002]
and Sun et al. [2010] use a fourth-order polynomial and an RBF model, respectively, to develop response surface surrogates of their developed lower-fidelity models and then utilize a correction function approach to correct these response surface surrogates to be used in optimization problems with their original computationally expensive models.

### 3.2.2. Variable-Fidelity Models With Different Variable Spaces

[111] The variable space associated with a lower-fidelity model may differ from the original model variable space. In such cases, the number of variables associated with the lower-fidelity model can be unequal to (typically less than) the number of original variables. Since their application is not trivial, multifidelity models when defined on different variable spaces have been less appealing in surrogate modeling literature [Simpson et al., 2008]. When the spaces are different, a mapping must be established such that \( \hat{x} = P(x) \) or \( \hat{x} = Q(x) \) where \( x \) and \( \hat{x} \) are corresponding points in the original and lower-fidelity variable spaces, and \( P \) and \( Q \) are mapping functions that transform points from one space to the other. In some cases, the space mapping relationship is clear based on the physics of the problem, and the lower-fidelity variable vector is a subset or interpolation of the original variable vector. However, when such knowledge is not available, empirical relationships are to be derived. The space mapping approach, explained in section 3.2.1, is a means to derive these empirical relationships [Bandler et al., 1994; Bandler et al., 2004]. The main procedure in space mapping between different spaces is essentially the same as the space mapping procedure when the spaces are identical. Another mapping approach has been proposed based on proper orthogonal decomposition (also called principal component analysis) and compared against space mapping in a trust-region optimization framework [Robinson, 2007; Robinson et al., 2006]. Robinson et al. [2006, 2008] incorporate the idea of correction function into space mapping to match the gradients of the lower-fidelity and original response surfaces at points of interest.

### 3.3. Related Research in Water Resources Literature

[112] The basic idea of replacing a high-fidelity simulation model with a lower-fidelity model of the system for the purposes of optimization and uncertainty or sensitivity analysis is a concept that exists in the water resources literature. However, the vast majority of such studies have applied this idea independent of the lower-fidelity surrogate modeling studies, methods, and terminologies from other disciplines described in section 3.2. This section reviews the research efforts arising from water resources modeling and relates them to the general methodologies for lower-fidelity surrogate modeling. The terms in quotations below represent the terminology used in the associated publications.

[113] There are research efforts to reduce the complexity level of various water resources models to be typically used in optimization or calibration. Ulanicki et al. [1996] and Muschler and Savic [1999] develop “model reduction methods” to simplify water distribution network models by eliminating less important pipes and nodes and allocating their demands to the neighboring nodes. Shamir and Salomons [2008] and Preis et al. [2011] utilize the model reduction method developed by Ulanicki et al. [1996] to create “reduced models” of water distribution networks to be used in optimization frameworks. McPhee and Yeh [2008] develop a “reduced model” of a groundwater model and link it with optimization in lieu of the original model for the purpose of groundwater management — this “reduced model” is defined over a different parameter space based on empirical orthogonal functions (ordinary differential equation instead of partial differential equation). Vermeulen et al. [2004] and Siade et al. [2010] propose methods to develop “reduced models” of high-dimensional groundwater models based on proper orthogonal decomposition. Vermeulen et al. [2005] and Vermeulen et al. [2006] utilize such “reduced models” as substitutes of the original groundwater models for model inversion (calibration).

[114] The well-established idea of using response matrices in place of full groundwater models for groundwater management (e.g., in the work of Reichard [1995]), where the same methods for developing response surface surrogates are typically used, can also be classified as lower-fidelity surrogates. Cheng et al. [2011] also refer to the response matrix as a “reduced model” replacing a groundwater model in their optimization problem formulated for groundwater management. Pianosi and Soncini-Sessa [2009] use a “reduced model” of a reservoir system to improve the computational efficiency of stochastic dynamic programming for designing optimal reservoir regulation plans. Crout et al. [2009] develop a methodology to “reduce” water resources models by iteratively replacing model variables with constants.

[115] Notably, in all the above studies that are for optimization purposes, reduced models (i.e., lower-fidelity models) after being developed are treated as if they are high-fidelity representations of the underlying real-world systems and fully replace the original models (i.e., high-fidelity models) in their analyses. In other words, the discrepancies between the high- and low-fidelity models are ignored.

[116] Simplified surrogate models have been also used with different Markov Chain Monte Carlo (MCMC) frameworks for uncertainty analysis of water resources models. Keating et al. [2010] develop a simplified groundwater model as a “ surrogate” of a computationally intensive groundwater model, defined over a different parameter space. They compare the null-space Monte Carlo (NSMC) method and an MCMC for autocalibration and uncertainty assessment of the surrogate instead of the original model, and then the NSMC, with techniques developed for the surrogate, is used on the original model. Efendiev et al. [2005] develop a two-stage strategy employing a groundwater model with a coarse grid, referred to as “coarse-scale model,” as a lower-fidelity surrogate of the original “fine-grid model” to speed up an MCMC experiment. In their methodology, both the surrogate and original models are defined over the same parameter space, and the surrogate is first evaluated to determine whether the original model is worth evaluating for a given solution.

[117] There are very few surrogate modeling studies in water resources addressing the discrepancies between the response surfaces of the lower-fidelity surrogate and the original model. Mondal et al. [2010] develop “coarse-grid models” (also called “upscaled models”) of a
(high-fidelity) groundwater model to speed up MCMC computations for uncertainty quantification; in their study, the discrepancies are recognized and quantified by a linear correction function built off-line before MCMC experiments to avoid biased approximated posterior distribution. Cui et al. [2011] also develop a “reduced order model” of a “fine model” by coarsening the grid structure and employ it within a correction function framework to enhance the efficiency of an MCMC algorithm for groundwater model inversion; an adaptive local correction function is used in their study in the course of MCMC sampling to improve the accuracy.

4. Efficiency Gains of Surrogate-enabled Analyses

[118] The most important question in assessing a surrogate-enabled analysis is how efficient or effective it is in comparison with other efficient alternative tools without surrogate modeling, especially because the computational efficiency achieved is the main factor motivating the research and application of surrogate modeling. Surrogate-enabled analyses typically sacrifice accuracy for efficiency as they utilize approximate models (less accurate than the original models) to more efficiently achieve the analysis objectives. As such, there is always a risk that surrogate models yield misleading results; this risk is higher when the original response landscape is complex and deceptive and is minimal for simple original response landscapes (e.g., almost negligible for smooth unimodal functions being optimized). A thorough discussion of this matter in the context of optimization is available in the work of Razavi et al. [2012] where a comparative assessment framework for metamodel-enabled optimizers is developed presenting a computational budget dependent definition for the success/failure of the metamodeling strategies. The careful selection of a benchmark alternative analysis or decision-making procedure without surrogate modeling is a vital step for fair assessment of a given surrogate-enabled analysis. To be clear, a benchmark alternative analysis has available at least the same number of original model simulations as utilized in the surrogate-enabled analysis. Although Broad et al. [2005] note that “Metamodels should only be used where time constraints prohibit the possibility of optimizing a problem with a simulation model.” In our view, the determination of such prohibition is not always clear cut. In a sampling context, one may take fewer samples than they would prefer, while in a search context, the algorithm can be terminated before it converges.

[119] In an optimization or search context, the most tangible measure of efficiency gains over a benchmark alternative is computational saving. For a single metamodel-enabled optimization analysis (e.g., optimal design) this can be calculated as

\[
\text{Saving} = 100 \times \frac{t - t_s}{t_s},
\]

where \( t \) is the computational budget or time required to reach a desired solution quality through an algorithm without surrogate modeling, and \( t_s \) is the computational budget or time a surrogate-enabled algorithm requires to reach a solution with the same quality. Regis and Shoemaker [2007b] present such comparative assessments in terms of efficiency by quantifying \( t \) and \( t_s \) as the numbers of original function evaluations the algorithms require to reach within 1% of the optimal value of the original function. Computational budgets may be quantified as the total CPU clock time (as in the work of Behzadian et al. [2009], Broad et al. [2005], Kourakos and Mantoglou [2009]) or the number of original function evaluations (as in the work of Mugunthan and Shoemaker [2006], Regis and Shoemaker [2007a], Zou et al. [2007, 2009]). As stated in Table 1, 15 (out of 32) studies present quantitative information demonstrating the efficiency of the surrogate modeling strategies used. Some of these studies report the associated computational savings explicitly; in the other ones, savings are not clearly reported and we interpreted them based on the published results. According to Table 1, computational savings achieved through the use of surrogate models can vary significantly, ranging from 20% of CPU time in the work of Zhang et al. [2009] to 97% in the work of Zou et al. [2007].

[120] Razavi et al. [2012] demonstrate that the failure of metamodel-enabled optimizers is a function of not only the degree of complexity and deceptiveness of the original landscape but also the available computational budget. In very limited computational budgets surrogate modeling is expected to be very helpful, whereas when the computational budget is not severely limited, surrogate modeling might not be as helpful, as equivalent or better solutions can be achieved by the benchmark optimizer. We believe similar findings are probable for all other types of metamodel-enabled analyses. However, the details of the comparative efficiency assessment framework for each of these other types of analysis (i.e., sensitivity analysis or reliability assessment), along with developing meaningful variants for equation (14), would need to be determined. An example variation to the efficiency assessment procedure for metamodel-enabled GLUE is demonstrated in the work of Khu and Werner [2003].

[121] In any surrogate-enabled analysis, the available computational budget or time is divided between three main parts: (1) budget or time required to run the original model, (2) budget or time required to develop, run, and update the surrogate model, and (3) budget or time the analyst needs to identify and create an appropriate surrogate-enabled analysis framework. Parts 2 and 3, which are referred to as “metamodeling time” and “analyst time,” respectively, should consume a small portion of the available computational budget leaving the majority for part 1. Nonetheless, metamodeling time and analyst time should ideally be taken into account when assessing the computational efficiency of a surrogate-enabled analysis. The computational and perhaps the analyst’s efforts are typically higher for lower-fidelity surrogates than response surface surrogates. As such, it is difficult to imagine any comparison involving lower-fidelity surrogates on the basis of the number of original function evaluations required as is commonly done with response surface surrogate comparisons. When a developed surrogate is to be used in repeat applications the importance of the analyst time is reduced.

[122] Any conclusion on the efficiency of a developed algorithm with surrogate models must be based on performing multiple replicates as any single application of such an algorithm (as with any other stochastic algorithm) is a single performance level observation from a statistical
population of possible performance levels. Despite the obvious computational burden of performing multiple replicates, it is the only way to conduct valid numerical assessments and comparisons.

5. Summary and Final Remarks

[123] There is a large body of literature, from different contexts and disciplines, developing and applying a wide variety of surrogate modeling strategies typically to improve the computational efficiency of sampling or search-based modeling analyses. The surrogate modeling literature was reviewed with an emphasis on research efforts in the field of water resources modeling. A set of publications including 48 references on surrogate modeling in water resources problems were analyzed and summarized in this paper and more than 100 other references from other disciplines were also reviewed. We overviewed the components involved in a surrogate-enabled modeling analysis framework and detail different framework designs.

[124] The most important observations and available guidance on the alternative methods and surrogate modeling frameworks that have been applied to the water resources studies reviewed here are as follows:

1. It is not trivial to suggest the best function approximation technique for the purpose of response surface modeling, and metamodel developers typically pick a technique based on their preference and level of familiarity as well as software availability. Function approximation techniques that are able to (1) act as exact emulators, (2) provide a measure of approximation uncertainty, and (3) efficiently and effectively handle the size of the data set (design sites) of interest, conceptually seem to be the most appealing for modeling the deterministic response of computer simulation models.

[126] 2. The metamodel-enabled optimization frameworks that utilize metamodel approximation uncertainty estimates (section 2.4.4) are conceptually the most robust strategies in comparison with the other three frameworks, especially for problems with highly multimodal and deceptive original response surfaces. In our view, this framework, particularly the statistics based approach [e.g., Jones, 2001] is underutilized in the water resources literature as the metamodel uncertainty characterization should prove useful in Bayesian model calibration studies and traditional Monte Carlo-based reliability or uncertainty analysis.

[127] 3. When evidence is available suggesting the original function is a relatively simple/unimodal function, using the basic sequential framework or adaptive-recursive framework would be the most appropriate as they would be successful and more efficient.

[128] 4. Difficulties are introduced moving from unconstrained (or just box-constrained) surrogate-enabled single optimization to surrogate-enabled constrained or multi-objective optimization (see sections 2.5.5 and 2.5.6).

[129] 5. Probably the most important limitation of surrogate modeling in applicability, especially response surface surrogate modeling, is when the number of dimensions in the problem variable space is large (successful surrogate-enabled analyses reviewed here were limited to 50 at most and typically less than 20 explanatory variables). Lower-fidelity surrogates are much less vulnerable to this limitation.

[130] 6. Lower-fidelity models are conceptually more reliable in exploring the unseen regions in the explanatory variable space compared to response surface surrogates. This reliability directly relates to the level of fidelity of a surrogate model and diminishes for the surrogates with very low fidelity. As there is typically a trade-off between the level of fidelity of a model and its computational demand, the lower-fidelity model developers should create a level of fidelity that is sufficiently faithful to the original model while being efficient enough to permit the case study specific analysis required.

[131] 7. Lower-fidelity models have advantages when there is an interest in emulating multiple model outputs or in multiobjective optimization with two or more emulated objectives as the lower-fidelity models would inherently account for output/objective function correlations.

[132] The following remarks summarize some suggestions on future research directions, many of which are inspired from ideas not commonly used in water resources literature:

[133] 1. Since the level of complexity of the original response function (which is typically unknown a priori) plays a key role in the determination of an appropriate function approximation technique, future research may be directed at developing methods to preanalyze the original response landscapes with a very limited number of samples to measure their level of complexity. The study by Gibbs et al. [2011] is an example of research efforts for the response landscape analysis in the optimization context (not involving surrogate modeling).

[134] 2. The strategies utilizing lower-fidelity surrogate models are relatively new and seem very promising as they circumvent many of the limitations accompanying response surface surrogates and although the lower-fidelity surrogate concept is slowly making its way into the water resources literature, there are multiple advanced strategies in the broader research community that are unseen or underutilized in the water resources literature we reviewed here (e.g., correction functions and space mapping).

[135] 3. A recent review by Viana and Haftka [2008] argues against applying a single surrogate model and instead suggests that multiple surrogate models should be fit and even used in conjunction with one another. The body of multisurrogate model methods they review should prove useful in water resources applications.

[136] 4. Building new/revised model analysis specific computational efficiency assessment frameworks, similar in concept to the one proposed in the work of Razavi et al. [2012], for Bayesian model calibration studies, sensitivity analysis, multiobjective optimization and traditional Monte Carlo based uncertainty analysis would help formalize metamodel-enabled methodological comparisons. In particular, such frameworks should account for the challenges noted in section 4 associated with the comparison of response surface surrogates with lower-fidelity modeling.

[137] Our final direction for future research is the most important one. Studies demonstrating a methodology for validation, or perhaps a case study specific proof of concept, of the entire metamodel-enabled analysis would be invaluable. In practical situations, the metamodel-enabled modeling analysis would not be repeated without a metamodel and the analyst either hopes the metamodel-enabled analysis yields helpful results (e.g., the list of most sensitive model parameters is mostly correct) or provides a better answer than they started with (e.g., the optimization
solution is better than the initial solution before any modeling. Such complete uncertainty about the quality or accuracy of the final analysis result after such a time consuming procedure does not breed confidence—in particular given that the success or failure of the entire metamodel-enabled analysis depends on many subjective decisions, the computational budget, the case study original model characteristics, the random number generator, etc. Imagine suggesting and defending a worldwide policy decision to combat climate change on the basis of a single metamodel-enabled analysis without rigorous validation showing the entire procedure could reliably accomplish what it was intended to do. The real question is how to build relevant case study specific examples for proof of concept—clearly, this would involve developing lower-fidelity models of the system.

[138] Acknowledgments. The authors would like to thank the reviewers, Holger Maier, Uwe Ehret, and the two anonymous reviewers, for their extensive and very helpful comments and suggestions, which significantly improved this review paper.

References


this would involve developing lower-fidelity models of the case study specific examples for proof of concept—clearly, intended to do. The real question is how to build relevant system optimization using metamodels, J. Water Resour. Plann. Manage., 131(3), 172-180.


Kennedy, M. C., and A. O’Hagan (2000), Predicting the output from a complex computer code when fast approximations are available, Biometrika, 87(1), 1-13.
Maschler, T., and D. A. Savic (1999), Simplification of water supply net-
McKay, M. D., R. J. Beckman, and W. J. Conover (1979), A comparison of
three methods for selecting values of input variables in the analysis of output
from a computer code, Technometrics, 21(2), 239–245.
Mckinney, D. C., and M. D. Lin (1994), Genetic algorithm solution of
McPhee, J., and W. W. G. Yeh (2008), Groundwater management using
Manage., 134(2), 161–170.
Melching, C., S. B. C. Yen, and H. G. Wenzel (1990), A reliability estima-
tion in modeling watershed runoff with uncertainties, Water Resour. Res.,
26(10), 2275–2286.
Michalski, R. S. (2000), Learnable Evolution Model: Evolutionary proc-
Mondal, A., Y. Efendiev, B. Mallick, and A. Datta-Gupta (2010), Bayesian
uncertainty quantification for flows in heterogeneous porous media using
reversible jump Markov chain Monte Carlo methods, Adv. Water Resour.,
33(3), 241–256.
Wiley, New York.
Mugunthan, P., and C. A. Shoemaker (2006), Assessing the impacts of pa-
rameter uncertainty for computationally expensive groundwater models,
Mugunthan, P., C. A. Shoemaker, and R. G. Regis (2005), Comparison of
function approximation, heuristic, and derivative-based methods for
automatic calibration of computationally expensive groundwater bio-
2005WR004134.
Muller, A. A., and A. Messac (2006), Metamodeling using extended radial
Myers, R. H., and D. C. Montgomery (2002), Response Surface Methodol-
Nakayama, H., M. Arakawa, and R. Sasaki (2002), Simulation-based opti-
O’Hagan, A. (2006), Bayesian analysis of computer code outputs: A tuto-
Papadrakakis, M., N. D. Lagaros, and Y. Tsompanakis (1998), Structural
Pianosi, F., and R. Soncini-Sessa (2009), Real-time management of a multi-
purpose water reservoir with a heteroscedastic inflow model, Water Resour.
Pichon, V., N. H. Kim, R. T. Haftka, and N. V. Queipo (2008), Conserva-
tive predictions using surrogate modeling, in 49th AIAA/ASME/ASCE/AHS/ASC Structures, Structural Dynamics,
Ponweiser, W., T. Wagner, D. Biermann, and M. Vincze (2008), Multiobjec-
tive optimization on a limited budget of evaluations using model-assisted
draulic state estimation technique using reduced models of urban water
Ratto, M., and A. Pagano (2010), Using recursive algorithms for the effi-
94(4), 367–388.
Ratto, M., A. Pagano, and P. Young (2007), State dependent parameter meta-
modelling and sensitivity analysis, Comput. Phys. Commun., 177(11),
863–876.
Ratto, M., A. Castelletti, and A. Pagano (2011), Editorial - Emulation tech-
niques for the reduction and sensitivity analysis of complex environmen-
Razavi, S., and B. A. Tolson (2011), A new formulation for feedbackward
neural networks, IEEE Trans. Neural Networks, 22(10), 1588–1598,
Razavi, S., B. A. Tolson, and D. H. Burn (2012), Numerical assessment of
metamodeling strategies in computationally intensive optimization,


Simpson, T. W., V. V. Toropov, V. Balabanov, and F. A. C. Viana (2008), Design and analysis of computer experiments in multidisciplinary design optimization: A review of how far we have come—or not, paper presented at 12th AIAA/ISSMO multidisciplinary analysis and optimization conference, Victoria, B.C., Canada.


