Proceedings of the 2020 Winter Simulation Conference
K.-H. Bae, B. Feng, S. Kim, S. Lazarova-Molnar, Z. Zheng, T. Roeder, and R. Thiesing, eds.

TUTORIAL: METAMODELING FOR SIMULATION

Russell R. Barton

Department of Supply Chain and Information Systems
Smeal College of Business
The Pennsylvania State University
University Park, PA 16802, USA

ABSTRACT

Metamodels are simpler computer models that are designed to mimic the input-output behavior of discrete-event or other complex simulation models. They are models of models, thus the name (provided by Jack P. C. Kleijnen). This introductory tutorial will highlight uses of metamodels, commonly used metamodel types, the linkage between metamodel type and the set of simulation model runs used to fit the metamodel, and basic issues in building and validating metamodels. The tutorial ends with a brief summary of recent research in metamodel types and use, and the implications for future metamodeling applications.

1 INTRODUCTION

Discrete event simulation (DES) provides behavioral modeling of complex stochastic systems. DES models are used for system performance prediction, system design, and policy formulation. DES models can be exercised at lower cost, in shorter time, and with less risk than exercise of a real system. In many cases exercise of the real system is not possible, either due to practical constraints or because the system does not yet exist. But if the DES system model is complex, the computational effort in system simulation can be substantial. Metamodels are useful in this context. Simulationists then often fit a computationally efficient approximation to the simulation model: a metamodel. The term metamodel was popularized and developed by Jack Kleijnen (for example, Kleijnen 1975), but the term and concept were both originated by Robert Blanning (1974, 1975). The term indicates a mathematical approximation that models the behavior of another model. These approximations are also called surrogate and response surface models. Metamodeling permits special insights into system performance, and enables extensive exercise of the system under different design conditions at relatively modest computational cost.

It has been five years since the last metamodeling tutorial at the Winter Simulation Conference. While there have been many advances in metamodeling during that time, the introductory body of knowledge is not greatly changed, and so this tutorial borrows heavily from its predecessor. The greatest changes have been in the use of neural networks (deep learning), regression trees, and Gaussian process (GP) metamodels. Aspects relevant in an introductory setting are presented in Section 3.

This tutorial aims to provide an accessible introduction to the beginner. This does not obviate a need for a familiarity with mathematical models and basic statistics. The intended learning outcome is for participants to understand the key tasks and tools needed to build, validate and use simple metamodels using tools like Excel, Minitab, SAS/JMP or R. Metamodeling process and metamodel types have greater emphasis than the experiment designs used to fit them. For more detail on the design of experiments see Barton (2013), Kleijnen (2008), Law (2017), and Sanchez et al. (2018), all of which provide different, complementary views. The tutorial focuses on common uses for metamodels, not on sophisticated applications such as quantile estimation (Batur, Bekki, and Chen 2018) or input uncertainty (Barton, Nelson, and Xie 2014). To permit easy accessibility to reference material, in some cases I give preference to a

Winter Simulation Conference *Proceedings* paper over a related journal publication. All WSC papers are available for free download at www.informs-sim.org.

The next section places the metamodeling activity in the overall process of building and using discrete-event simulation models. That is followed by a section reviewing and illustrating the basic metamodeling process for predicting average waiting time in a queue as a function of mean service time. The next section describes selecting and scaling the predictor variables (x) for the model and the selection and possible transformations of output variables (y) to be predicted, again illustrated for the queueing example. An introduction to metamodel types and associated experiment designs follows. The next section describes the basics of metamodel validation, and provides warnings about common metamodeling mistakes.

2 THE ROLE OF METAMODELS IN THE OVERALL DES MODELING PROCESS

There is general agreement about the major activities that are part of a simulation study; for examples see Banks et al. (2009) and Law (2014). Below is a summary in ten key steps:

1. Formulate the problem and objectives of the study.
2. Collect data, formalize assumptions and conceptual model.
3. Test validity of the conceptual model against objectives.
4. Identify design parameters and outputs; program the model.
5. Verify that the program output matches the conceptual model expectations.
6. Validate program: against real system data, via sensitivity analysis or other means.
7. Create DOE for decision support.
8. Run simulation program using decision support DOE.
9. Analyze results of experiments.
10. Use *simulation* results to inform decisions: through prediction, sensitivity analysis, optimization.

The role of metamodels fits in the last four steps of this process. With metamodeling, they are somewhat revised:

7. Create DOE for decision support, *and DOEs to fit and validate metamodel*. (Metamodel purpose, design parameters, outputs and metamodel type needed.)
8. Run simulation program *using DOEs to fit and validate metamodel*.
9. Analyze results of experiments *to fit and validate metamodel*. If metamodel passes, run metamodel under decision support conditions.
10. Use *metamodel* results (coefficients, additional predictions/analyses) to inform decisions: through prediction, sensitivity analysis, optimization.

The key assumption is that there is interest in predicting system performance under a large number of conditions; too many to explore directly using the DES model.

In the next section we expand a bit on steps 7, 8, and 9 in the metamodel process above. The sections following go into greater detail on model type, experiment design and validation.

3 METAMODELING: BASICS, PURPOSE, AND PROCESSES

3.1 Basics of Metamodeling

A metamodel is a function, say f, that takes some simulation model design parameters as inputs, represented here by a vector x, to an output, $f(x)$. Examples of model design parameters include input probability distribution parameters, such as arrival rate and mean service time; and system configuration parameters, such as the number of servers, service priority, operational protocols, and buffer capacity. For now, assume that any design parameter can be coded numerically, even if only as a 0-1 variable. There will be more

about how to do this later in the tutorial. The metamodel $f(x)$ produces an approximation to some characteristic of a simulation output Y, e.g., mean of Y, standard deviation of Y, 0.9-quantile of Y. Examples of simulation outputs are time in the system for a set of jobs or customers, utilization of a particular resource (e.g., operator, machine), or perhaps net revenue over a specific time period. Generally, these outputs are averaged over the length of a simulation run, but the averages vary randomly from run to run. If the value of the characteristic is $Y(x)$ for an actual simulation run with the design parameters set to the values in x, then we represent the fitted metamodel approximation $f(x)$, as:

$$h(Y(x)) \approx f(x), \tag{1}$$

where h represents some function of the random variable Y such as its mean, standard deviation, or a quantile. Note that the distribution of Y depends on the values of the design parameters. The simplest and most common metamodel type is linear regression. The term "linear" refers to the way the unknown coefficients come into the model. Linear regression can capture curvilinear relationships.

To illustrate the basics of metamodeling, we will fit two linear regression metamodels to data from a simple queueing simulation. Although simulation is not needed for an M/M/1 queue, such a system is easy to understand and has behavior that is frequently seen in simulations. In this case we want to explore how the average waiting time (not including service; the *output*) varies with mean time to service a customer (the *input*, or *design parameter*), assuming an arrival rate of 1 customer per unit time (in some scaled units of time). Figure 1, made using Minitab® (Minitab 2020) shows the results of simulations of 5000 customers in systems with mean service times of 0.7, 0.75. 0.8, 0.85, 0.9 and 0.95. There are three replications for each of the run conditions.

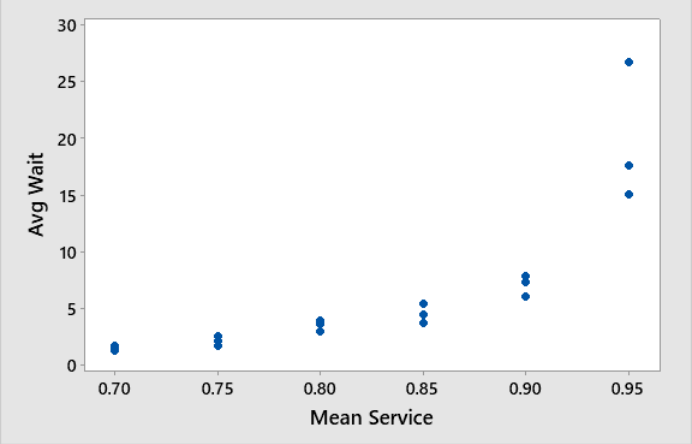

Figure 1: Output of M/M/1 experiments, three replications each at six mean service time settings.

If we fit a linear regression model to this data, the metamodel produces the fit seen in Figure 2. If we fit a quadratic metamodel to these data, it produces the fit shown in Figure 3. Clearly, there are problems with the fidelity of both of these fitted metamodels. We will return to this example throughout the tutorial to discuss each aspect of the metamodeling process.

For Figure 2, the metamodel corresponding to (1) has $Y \equiv$ average waiting time for the 5000 customers in a simulation run, $h(Y) \equiv$ the expected value of this run average, i.e., E(Y); $x \equiv$ mean service time (the x vector has only one component), and $f(x) = -44.19 + 61.42x$. For Figure 3, the fitted metamodel is $f(x) = 269.9 - 708.3x + 466.5x^2$.

Now that we have a basic understanding of what a metamodel is, we can discuss purposes for which metamodels are built.

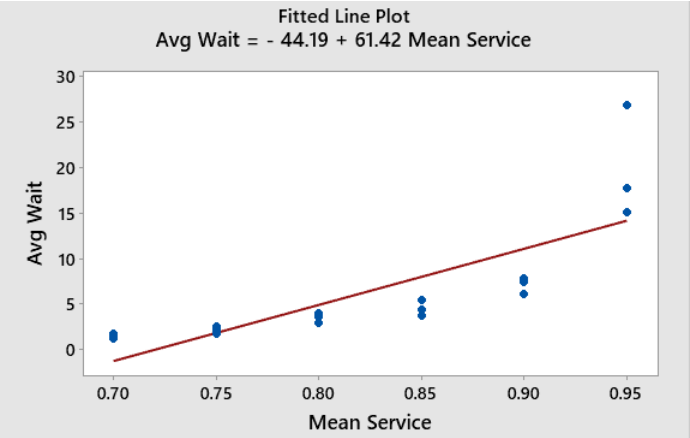

Figure 2: A regression metamodel fitted to the data, with intercept and linear term only.

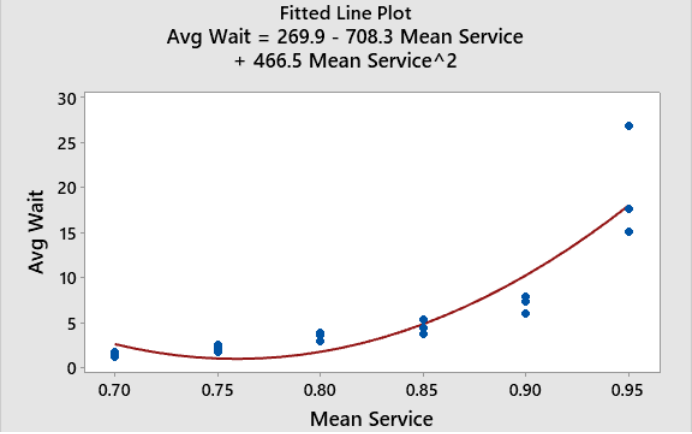

Figure 3: The fitted quadratic regression metamodel is an improvement.

3.2 The Metamodeling Process

The rest of this introductory tutorial is organized around the steps of the metamodeling process. An early formal description was given by Burdick and Naylor (1966). Table 1 shows a similar process.

Table 1: The metamodeling process.

Step	Activity
1	Determine Purpose(s) for Metamodeling
2	Identify Design Parameter(s) and Output(s)
3	Choose Metamodel Type
4	Based on Metamodel Type and on Purpose, Choose Experiment Design to Fit Metamodel
5	Conduct Simulation Runs Specified by the Experiment Design; Fit Metamodel
6	Validate Metamodel Adequacy: If Unsatisfactory Return (usually) to Step 3
7	Use Metamodel for Intended Purposes

4 METAMODELING PURPOSE

Metamodels generally have three characteristics that give them advantage over the simulation model for certain purposes. A metamodel f generally has *explicit form*, *deterministic output*, and, once fitted, is *computationally inexpensive* to evaluate.

Due to their explicit form, metamodels are often described as providing *insight* - that is, an understanding of the general relationship between design parameters and some performance measure related to the simulation output. For the example in Figure 2, we find a positive coefficient, which indicates both a *positive relationship* between service time and the expected value of average waiting time, and the magnitude of the slope coefficient gives the level of *sensitivity*. For a metamodel with one design parameter this sensitivity measure may not be very interesting, but when there are tens or hundreds of design parameters, the size of the coefficients can be used to identify the most influential design variables and *screen* the others. The coefficients of the metamodel shown in Figure 3 provide more insight: the relationship between waiting time and service time is *convex*. We know that because the coefficient of the squared term is positive. Further, it is easy to evaluate either of these metamodels for many values of service time, allowing one to characterize global variation in the mean value.

Metamodels, once fitted, can be used as a proxy, to evaluate instead of making (computationally expensive and stochastic) simulation runs. Further, because of their explicit form, they can be used in many computationally intensive operations, such as optimization (Barton and Meckesheimer 2006), input model uncertainty, quantiles and conditional value at risk, and robust design (Dellino, Kleijnen, and Meloni 2009).

5 IDENTIFYING DESIGN PARAMETERS AND OUTPUTS

5.1 Design Parameter and Output Selection

Typically, simulation models have many design parameters that might be included in a metamodel. Each parametric probability distribution used in the model has parameters. These are called input distribution parameters. They characterize randomly varying interarrival times, service times, times until machine breakdown, message lengths, routing choices, transport times, number of workers out sick, uncontrollable environmental factors, and other characteristics of the simulated system that have a stochastic nature.

In addition, there are design parameters that are deterministic in value: number of workers scheduled to work at a particular time, number of machines, processing protocols and other characteristics of the system being modeled that can be changed, either for a currently operating system or some system to be built in the future.

The particular parameters to include in the metamodel depend on three things: i) a desire to include parameters that the decision maker would like to explore changing; ii) the need to model the impact of any uncontrollable environmental factors that affect system performance; and iii) the recognition that more metamodel parameters generally means a larger fitting experiment, and more computational effort to fit the metamodel. Further, for regression metamodels, the inclusion of higher order terms to capture interaction effects and nonlinearity (x^2 for the M/M/1 example) also adds to the size of the experiment design. Cause-effect diagrams and a-priori plots can help identify the design parameters and any expected interactions or nonlinearity (Russell R. Barton 2013). For the M/M/1 example above, the experiment design was presented as given. In actuality, it would be selected after considering whether to include x^2 and perhaps x^3 in the metamodel.

Each function of the simulation outputs that is of interest is usually modeled separately, that is, a separate metamodel is fitted for each function of the simulation outputs needed for the purpose(s) identified in Step 1. Multiple response surface models were considered in the early paper by Burdick and Naylor (1966) and many subsequent authors. True multivariate metamodels are a recent development (Liu, Cai, and Ong 2018). For the M/M/1 example above, only one function of one simulation output was considered: h corresponds to the expected value, and Y is the average waiting time of 5000 simulated customers. Occasionally, h will be chosen to reduce the nonlinearity of the output with respect to the design parameters,

or to achieve equal (homogeneous) output variance across the design space. Methods for choosing such transformations are summarized in Barton and Meckesheimer (2006).

5.2 Continuous and Discrete Design Parameters and Outputs

Generally metamodeling assumes that all design parameters and outputs can take on continuously varying values. But many design parameters are discrete. Examples include numbers of servers, machines or other resources, buffer sizes, number of products, type of processing protocol, system configuration alternatives, and so forth. When the parameter has a numerical value that is restricted to a discrete set of values, then assuming that the parameter can take on a continuous set of values is often practical. The discrete nature places a restriction on the experiment design that is used for fitting, but the fitted metamodel can be evaluated on the discrete set of allowed values. When there are only two values for the parameter (e.g., *Protocol A* and *Protocol B*), a discrete numerical characterization can be assigned, for example, $x = 0$ for *Protocol A* and $x = 1$ for *Protocol B*. If the parameter does not have numerical value and there are more than two levels, numerical conversion is still possible, but requires additional x components. For example, for three protocols, let $x_1 = 1$ if *Protocol A* is used, and = 0 otherwise. Let $x_2 = 1$ if *Protocol B* is used, and = 0 otherwise. If *Protocol C* is used, then both x_1 and $x_2 = 0$. Then metamodel terms for x_1 (or x_2) will indicate the differences of *Protocol A* (or *Protocol B*) from *Protocol C*.

When the simulation output is discrete (e.g., success or failure), there is a restriction on the type of metamodel. These are generally called classifier or discriminant metamodels. Meckesheimer et al. (2001) consider metamodels with both continuous and discrete responses.

5.3 Scaling/Coding Parameters and Outputs

In addition to the output transformations and qualitative variable coding mentioned previously, the ability of the metamodel to provide insight depends on careful scaling and coding of all design parameters. Generally, code all numerical design parameters so that -1 is the smallest value taken, and +1 is the largest value. This coding is accomplished by the following:

$$x_{new} = 2[x - ((x_{max} + x_{min})/2)/(x_{max} - x_{min})]. \quad (2)$$

Compare the insight from the fitted metamodel in Figure 2 with that in Figure 4, based on the same data but rescaling the average service time using (2), to -1 for .7 and +1 for .95.

Insight from Figure 2: a unit increase in mean service time would result in an increase in average waiting time of approximately 61 time units (note: a unit increase in mean service time could not happen – the system would be unstable). Also, if the mean service time were reduced toward zero, the average waiting time would tend toward -44 time units (note: negative time is impossible). But Figure 4 coefficients are interpretable: moving from the middle service time (.825) to the highest service time (.95) will increase the average waiting time by approximately 7.7 time units. If the system is operated at the middle service time value (.825), the average waiting time will be approximately 6.5 time units. Clearly, basic insights from the regression model coefficients fail to materialize without careful coding of the design parameter value(s).

Further, Figure 5 shows that, unlike the difference between the models in Figures 2 and 3, the linear coefficient does not change between Figures 4 and 5. Note that the metamodel fit does not change – Figure 2 has the same shape as Figure 4 and Figure 3 as Figure 5 – only the interpretation of the coefficients changes. The intercept did change between Figure 4 and Figure 5, however. Fixing this requires more than coding of the design parameters, it requires coding the functions of those parameters used in the regression models. The topic of orthogonal polynomials is beyond this tutorial. See for example Montgomery (2012).

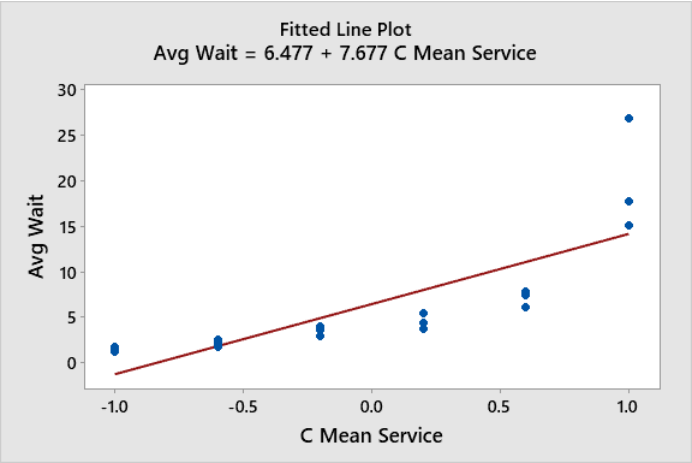

Figure 4: The linear regression metamodel with coded service time: fit is identical to that in Figure 2.

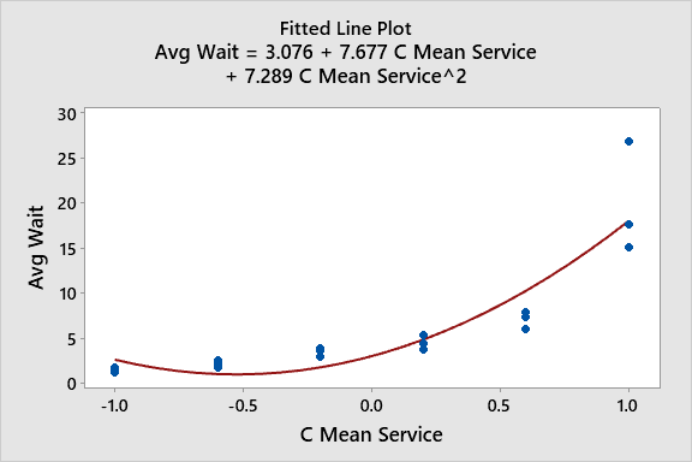

Figure 5: The quadratic regression metamodel with coded service time.

6 CHOOSE A METAMODEL TYPE

For this introductory tutorial we focus on two types of metamodels: linear regression and stochastic Kriging (spatial correlation) models. There are descriptions of other metamodel types in Barton (2009). We identify the metamodel type before selecting an experiment design for practical reasons: factorial and fractional-factorial designs, appropriate for linear regression, can cause significant numerical difficulties when used to fit Kriging models. Further, the complexity of the proposed regression model places (minimum) requirements on the type of factorial or fractional-factorial design that can be employed for fitting.

6.1 Linear Regression Metamodels

The form of the linear regression probability model that characterizes the simulation output is:

$$Y(x) = \beta_0 + \beta_1 g_1(x_1, x_2, \cdots, x_d) + \cdots + \beta_p g_p(x_1, x_2, \cdots, x_d) + \varepsilon \tag{3}$$

where ε are independent, normal random quantities with mean zero and unknown variance and there are d design parameters. Again, the model is *linear* because the unknown coefficients (β's) appear linearly (as multipliers) in the model. The g functions can be nonlinear in the x's, for example, $g_5(x_1, x_2, ..., x_d) = x_1^2$, or

$g_7(x_1, x_2, ..., x_d) = x_1x_5$. There are p terms in the model (not counting the intercept). The assumption is that the variance does not change depending on the values of $(x_1, x_2, ..., x_d)$. This model (3) implies that E(Y) has the same form as (3) but without the ε term. Further, the regression metamodel $f(x)$ will match (3) but with estimated values $b_0, b_1, \ldots, b_p$ for the unknown β coefficients. Since the estimated values $b_0, b_1, \ldots, b_p$ will vary randomly from one experiment to the next, conceptually the fitted metamodel depends randomly on the data. Given a set of data $\{x_i, y_i\}$ where $x_i = (x_{i1}, x_{i2}, \ldots, x_{id})$ is the vector of design parameter values for the i^{th} simulation run, and y_i is the corresponding output, let X be the matrix whose i^{th} row is x_i and let y be the column vector consisting of the elements $\{y_i\}$. Then the estimated coefficient vector $b = (b_0, b_1, \ldots b_p)$ is computed by:

$$b = (X'X)^{-1}X'y. \tag{4}$$

where the prime symbol denotes matrix transpose. The solution (4) minimizes the average squared deviation of the metamodel prediction from the observed simulation outputs. The intercept term b_0 has corresponding $g_0 \equiv 1$. For the M/M/1 example, the fitted metamodel in Figure 4 has $b_0 = 6.477$, and $b_1 = 7.677$. The fitted metamodel in Figure 5 has $b_0 = 3.076$, $b_1 = 7.677$, and $b_2 = 7.289$.

Linear regression models have simple form and so provide direct insight on the behavior of the simulation. When design parameters are coded over [-1, +1], then the magnitude of the linear coefficients indicate the relative sensitivities of the simulation output to all design parameters (over the defined ranges of parameter values). Similarly, quadratic coefficients can indicate nonlinearity, and convexity/concavity. Coefficients for cross-product terms indicate interaction effects – the sensitivity of the output to changes in one design parameter may vary, depending on the setting of another design parameter. Also, linear regression models are readily available in commercial statistical software.

Linear regression models using polynomial functions of the design parameters have limited flexibility, however. Figure 6 shows an attempt to find a better-fitting metamodel for the M/M/1 example by adding a cubic term. While the curve comes closer to the observed waiting times overall, it is no longer monotonically increasing, something we expect in a metamodel of mean waiting time versus mean service time. Adding more terms improves the fit near the six experiment design points but increases the excursions of the metamodel away from the design points. For the classic illustration of this "excursions" shortcoming of polynomial models see Figure 1 in Barton (1992).

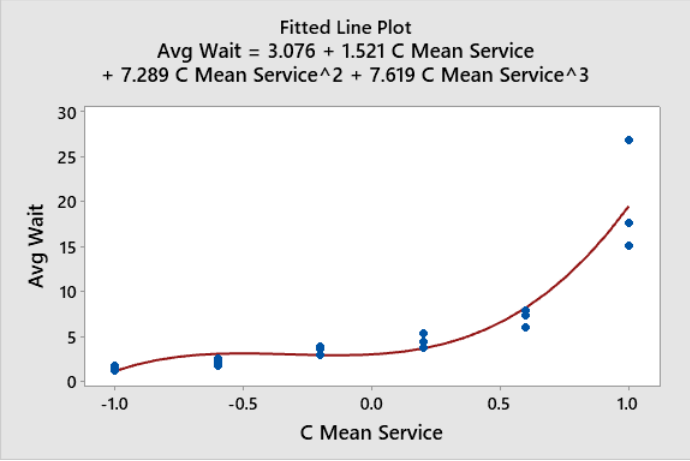

Figure 6: The fitted cubic regression metamodel is not monotonic.

Looking at Figures 1-6 it is apparent that the variance of the response is larger at higher average waiting times. This is particularly important for models where the output is some function of a queueing system, as

is often the case in discrete-event simulation. One approach to reducing this *heteroscedasticity* is to take different numbers of replications at different design points. By taking more replications at high-variance points, the variance of the average response at such points is reduced. This is an expensive proposition though: the spread you see is related to the standard deviation, which is only reduced as the square root, so to reduce the spread by a factor of two requires 4x the replications. For our M/M/1 data we would need approximately 120 additional replications for the x = .95 setting!

Often problems with heteroscedasticity *and* fitting can be reduced by transforming the dependent variable(s). A typical transformation for queueing output data is the logarithmic transformation. Figure 7 shows the same model as Figure 5 but using ln(Y) as the dependent variable. Note that the large differences in spread across the design points is reduced. In addition, the fit, as measured by R^2, is better.

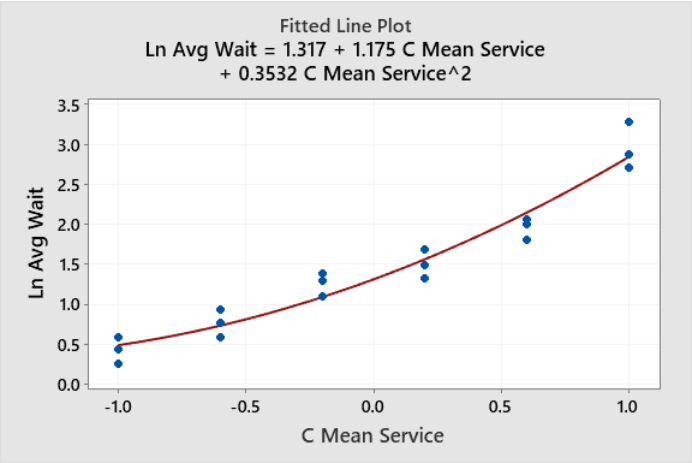

Figure 7: Transforming to ln(Avg Wait) reduces heteroscedasticity and improves fit.

6.2 Gaussian Process (Kriging) Metamodels

The simplest Kriging probability model is:

$$Y(x) = \beta_0 + M(x), \tag{5}$$

where M is the realization of a mean zero random field. That means it is a function drawn at random from the set of all functions whose nearby values are correlated according to a prespecified spatial correlation function. For that reason, these models are also called spatial correlation models. As in (3), the realization can be interpreted as $g_1(x) = M(x)$. Other regression-type g terms might be added, but typically the intercept β_0 is all that is needed for a good fit. When the randomness is assumed to be Gaussian, the models are called Gaussian Process (GP) regression models. GP models have been used to approximate deterministic response functions, since once the realization occurs, the model (5) has no intrinsic randomness. Since stochastic simulation models have an output with intrinsic randomness, Ankenman et al. (2010) added a normally distributed intrinsic error term, $\varepsilon(x)$ to (5), and allow covariance between $\varepsilon(x_i)$ and $\varepsilon(x_j)$. In this metamodel i indexes a unique set of design parameter values, $i = 1, \ldots, k$, with n_i replications run with the design parameter vector set to x_i. The fitted GP metamodel is:

$$\hat{Y}(x) = b_0 + [t^2 R_M(\hat{\theta}) + \hat{\Sigma}_\varepsilon]^{-1}(\bar{Y} - b_0 1_k) \tag{6}$$

where t^2 and $\hat{\theta}$ are spatial correlation parameters estimated from the experimental data, $\hat{\Sigma}_\varepsilon$ is the sample intrinsic error covariance matrix (sample variances and covariances across all experiment design points),

$R_M(\hat{\theta})$ is an approximate spatial correlation matrix computed using $\hat{\theta}$, and 1_k is a *k*-dimensional vector of ones. Popular stochastic GP modeling software implements a simple version of (5), with diagonal $\hat{\Sigma}_\varepsilon$.

GP models have great flexibility. They can model more complex response function shapes than is possible with polynomial regression metamodels. If one requires a global approximation to a nonlinear response, GP metamodels are an attractive alternative to linear regression models that use polynomial *g* functions. This comes at a cost. First the model is more complex to fit, and the capability is not included in some statistical packages. Second, fitted model coefficients give some indication of how rapidly the response changes as components of *x* change, but the detailed insight available from a fitted linear regression model cannot be obtained. Further, if experimental run conditions are scarce, predictions in design parameter space between experimental runs can be significantly in error due to mean reversion (see Figure 1 in Erickson et al. (2018). These models can be sensitive to parameter choices (Gramacy and Lee 2012) – the example below illustrates this. Because this is an introductory tutorial, the best advice here is to educate yourself thoroughly in these models, and their potential shortfalls.

6.3 Artificial Neural Network (ANN) Metamodels

Artificial neural network (ANN) metamodels develop the approximation function as a composition of networked simple nonlinear functions with adjustable coefficients. A feedforward neural network is an acyclic network of simple nonlinear (neuron) functions, each using a linear combination of outputs from the direct predecessor nodes; the output can be a single node or a set of nodes, so that multi-dimensional outputs are easily modeled. The ANN model can be thought of as a nested form of equation (3), where the *Y* output is defined node-by-node and the *g* functions are the outputs (*Y*s) of the immediate predecessor nodes. Predecessor nodes are usually grouped in layers with identical distance (path length) from the input nodes. There is one input node for each component of *x*. Layers between the input and output layers are called hidden layers. When there is more than one hidden layer the ANN process is often called "Deep Learning." More complex neural networks allow layer jumping and feedback, for an acyclic structure. Their fidelity and flexibility make them attractive for complex metamodeling and metamodel-based optimization (Dunke and Nickel 2020).

Although ANN software is readily accessible in R and other packages, implementation of artificial neural network (ANN) metamodels is beyond the level of this introductory tutorial. Coding the independent variables and choosing an ANN structure require care and sophistication. In an exploratory study (Fonseca, Navaresse, and Moynihan 2003) the authors note "the ANNs capabilities of effectively learning were highly restricted by the selected codification scheme." Further, ANNs are viewed as black boxes. While insight is possible from either linear regression coefficients or spatial correlation parameters, ANN coefficients generally do not provide insight on the impact of independent variables on simulation output.

7 CHOOSE EXPERIMENT DESIGN

In choosing an experiment design, one determines the number of distinct simulation settings to be run, and the specific values of the design parameters for each of these runs. Design types include random designs, optimal designs, combinatorial designs, Latin hypercube designs, orthogonal arrays, uniform designs, mixture designs, sequential designs, and factorial and fractional-factorial designs. For regression metamodels, the number and kinds of terms to be fitted place constraints on the minimum number of runs and minimum number of levels tested for each design parameter. Barton (2013) is a reference for this discussion, and also Sanchez et al. (2018).

7.1 Experiment Designs for Linear Regression

Experiment designs for regression are well-developed. There is a clear link between the form of the model being fitted and the kind of experiment design that is preferred. Typically, regression designs are either factorial designs or fractional factorial designs. Factorial designs are based on a grid, with each factor tested

in combination with every level of every other factor. Factorial designs are attractive for three reasons: i) the number of levels that are required for each factor is one greater than the highest-order power of that variable in the model, and the resulting design permits the estimation of coefficients for all cross-product terms ii) they are probably the most commonly used class of designs, and iii) the resulting set of run conditions are easy to visualize graphically for as many as nine factors.

The disadvantage of factorial designs is that they require a large number of distinct runs when the number of factors and/or the number of levels of the factors are large. In this case, fractional-factorials are often employed. Table 2 gives some guidance on experiment designs appropriate for regression modeling, depending on the purpose and nature of the mode, but is not comprehensive. For example, it does not include sequential designs, e.g. Wan et al. (2009).

7.2 Experiment Designs for GP Regression

Optimal experiment designs for Gaussian Process metamodels is an active field. Factorial designs have been found to work poorly with deterministic Kriging models. Latin hypercube designs are often used for GP models, but it is important to sample many Latin hypercube designs and choose one with good properties; for example the design that maximizes the minimum distance between any two points (maximin). Alternatives are Hammersley sampling sequences, orthogonal arrays, and uniform designs.

Table 2: Experiment designs for linear regression metamodels.

Objective	Minimum Size Factorial Designs	References
Initial screening	Saturated and supersaturated fractional factorial	Lin (1993), Li and Lin (2003)
Sensitivity	Saturated and resolution III Plackett-Burman fractional factorial	Sanchez and Sanchez (2005), Kleijnen (2008b)
Insight	3-level full or fractional factorial or central composite (more than 3 levels needed to check lack of fit)	Montgomery (2012), Kleijnen (2017), Sanchez et al. (2018)
Optimization	3- or more level fractional factorial	Montgomery (2012), Law (2014)

8 CONDUCT RUNS AND FIT MODEL

Unlike physical experiments, external environmental factors generally do not affect simulation results. Once the model type and design selection steps are complete, running the experimental conditions is straightforward, provided the experimenter keeps the simulation results attached to the correct values of the design parameter settings for each run.

Reusing random number streams for runs with different design parameter settings can induce correlation. Correlation can be hard to achieve, but it can result in better fits for regression metamodels (Schruben and Margolin 1978, Tew and Wilson 1992). Work by Chen et al. (2012) found that correlation induction using common random numbers was not effective for GP modeling, but more recent work seems to indicate an advantage for GP metamodel-assisted optimization (Pearce, Poloczek, and Branke 2019).

8.1 Software for Metamodeling

Virtually all simulation software allows input of a spreadsheet of simulation run parameters (created using your favorite DOE software, such as Matlab, Minitab, R, SAS or at the SEED Center for Data Farming) and spreadsheet output of results, for analysis using your favorite metamodeling software. Built-in metamodeling capabilities seem rare, as are specific design of experiments capability, based on the survey by Swain (2019). The survey had no specific entry for metamodeling, but software identifying DOE capability (batch run capability insufficient) includes AnyLogic, Flexsim, Frontline Solver, INCONTROL, ProModel, Siemens PLM, SIPmath Modeler. SAS/JMP and Arena were not participants in the 2019 survey,

but both SAS and JMP software identify the ability to link design of experiments capability with simulation and back to analysis/fitting metamodels.

When DOE and metamodeling are not supported, you should be prepared to provide spreadsheet input for the simulation experiment design and receive spreadsheet output from the runs to use in building your metamodel. The DOE and metamodel fitting can be managed by statistical software. Commercial software such as Excel-based packages, Minitab, or SAS/JMP provides an easy, menu-driven interface, while the freely available R software requires programming ability.

R provides high-quality statistical tools and includes many metamodel options. What makes R so powerful are its packages; there are over 5200 on the CRAN download site. The free package RStudio (RStudio 2020) provides a GUI that minimizes the R programming effort.

8.2 Using R to Fit a GP Metamodel

For Gaussian process regression, the R package *mlegp* is used (Dancik 2018). For this software the stochastic modeling is simpler than that in Ankenman et al. (2010). The stochastic variance terms are estimated directly at each design point using the replication sample variance. This is provided to *mlegp*; otherwise *mlegp* would use a pooled estimate, identical at every design point.

Below is the code to fit and plot the GP model. Commands include installing *mlegp* (do this only once to your machine), loading the *mlegp* library for this R session, grouping the replications, computing response means and variances, executing GP regression, and plotting actual vs predicted results. Note: the menu-driven interface in RStudio was used to import the Excel file containing the service mean and wait time data, placed in the R data object *WSC20QueueData*.

```
install.packages("mlegp")
library("mlegp")
x = WSC20QueueData[[1]]; y = WSC20QueueData[[2]]
reps = 3; nDOEpts = 6; groupID = rep(1:nDOEpts, each = reps)
grp_x = tapply(x, groupID, FUN=mean)
grp_meany = tapply(y, groupID, FUN=mean); grp_vary = tapply(y, groupID, FUN=var)
GPmodel = mlegp(grp_x,grp_meany,constantMean=1,nugget=grp_vary/reps,nugget.known=1)
xpreds = matrix((1:95)/100,95,1); ypreds = predict(GPmodel,xpreds)
par(mar = c(5,5,3,1)); plot(x,y,xlab="Mean Service", ylab="Avg Wait")
lines(xpreds,ypreds,col = "blue")
```

Figure 8 shows the plot. Notice that the prediction is poor at high mean service time. This is because the variance is allowed to differ across mean service values, and the high variance at high mean service means the prediction relies heavily on the spatial correlation, thus a predicted wait that is close to the waits at neighboring mean service times. If one uses a constant variance nugget, one might expect a closer fit at the high end, since the variance will be less.

The code is shown below, and Figure 9 shows the resulting fit. While the prediction at high mean service time is improved, the metamodel now exhibits undulation in its fit -- to a function known to be monotonic. Monotonicity for Gaussian process regression cannot be enforced in any straightforward way, but see Kleijnen and van Beers (2013). *This serves as a warning on the use of Gaussian process models. Note that in high dimensions, nonmonotonic structure is difficult to detect.*

```
anyReps(x)
GPmodel2 = mlegp(grp_x,grp_meany,constantMean=1,nugget=estimateNugget(x,y))
xpreds = matrix((1:95)/100,95,1); ypreds = predict(GPmodel2,xpreds)
par(mar = c(5,5,3,1)); plot(x,y,xlab="Mean Service", ylab="Avg Wait")
lines(xpreds,ypreds,col = "blue")
```

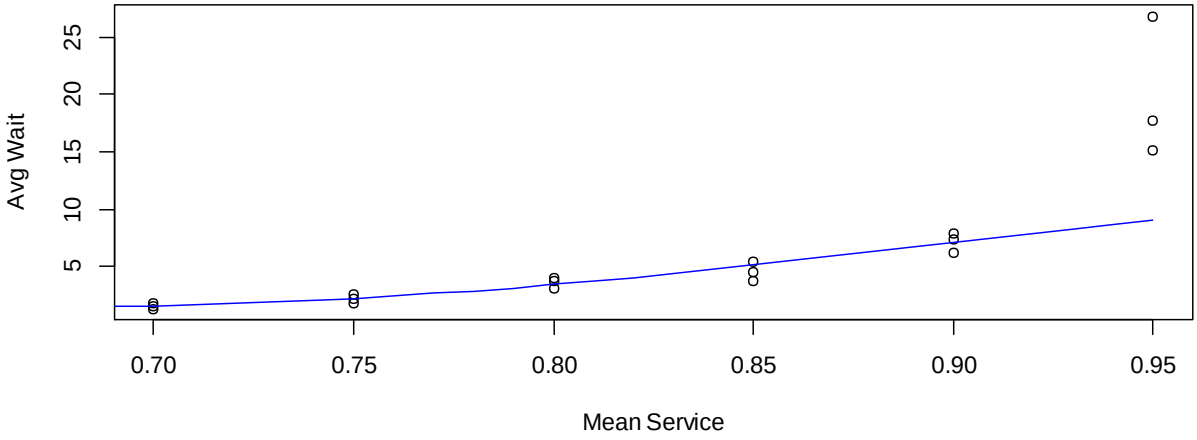

Figure 8. Gaussian process model fit to the M/M/1 simulation data.

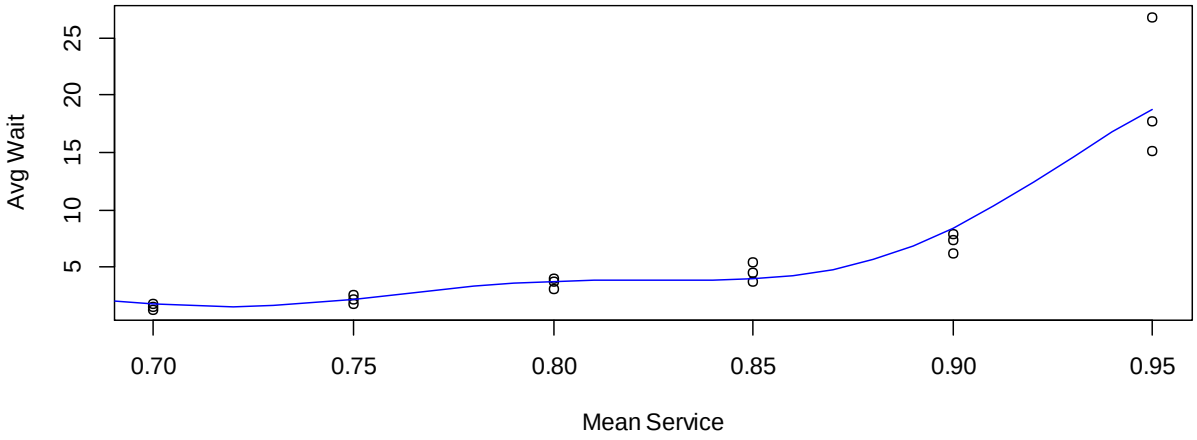

Figure 9: Gaussian process metamodel fit using common variance across all points.

9 VALIDATING METAMODEL ADEQUACY

The fitted model must be checked to see if the fidelity is adequate for the intended use. See Kleijnen and Sargent (2000) for more details on this process. For a regression metamodel for screening, simple statistical significance checks may be sufficient. For purposes where fidelity is important, additional goodness of fit tests are employed. There are well-developed goodness of fit tests for regression that appear in all commercial packages. To use them you will need to include more than the minimum number of design parameter levels.

For regression models, mean squared error (MSE) is provided automatically with the fit, as is R^2. High R^2 values can occur when there are just a few extreme values of x and correspondingly high or low values of the response. MSE will give a better assessment of fit in this case.

A general-purpose measure of fit that can be used outside the regression setting is to leave some design configurations and corresponding responses out of the set used to fit the model and then check the error of the fitted model at the design parameter settings left out of the fitting process. This process can be

computationally expensive if it is repeated for each possible omission. Meckesheimer et al. (2002) provide some efficient and effective assessment methods of this sort.

10 PUT THE METAMODEL TO USE – AND FURTHER STUDY

Assuming the fitted model passed the validation checks, it is ready to be used. Congratulations on successfully developing a metamodel! Remember though, that uses beyond the original purpose (e.g., using a screening metamodel for prediction or, worse, optimization) are not appropriate. This has been an introductory tutorial, the goal is to spur your interest in metamodeling with an understanding of the methods and their value. But it is only an introduction. Many books on simulation have a chapter on the design of experiments, which usually covers metamodeling. Three books with comprehensive coverage are Friedman (1996), Kleijnen (2008a - ebook available online through some university libraries), and Law (2014).

ACKNOWLEDGMENTS

I have had the good fortune to benefit from the wisdom of the authors cited in this paper. The Winter Simulation Conference has provided an opportunity to meet many of them (and many others) for direct discussions that helped my understanding of and appreciation for simulation metamodels. I thank the anonymous reviewer for helpful suggestions to improve this tutorial.

REFERENCES

Ankenman, B., B. L. Nelson, and J. Staum. 2010. "Stochastic Kriging for Simulation Metamodeling". *Operations Research* 58(2): 371-382.

Banks, J., J. S. Carson, B. L. Nelson, and D. M. Nicol. 2009. *Discrete-Event System Simulation*. 5th ed. Englewood Cliffs, NJ: Prentice Hall.

Barton, R. R. 1992. "Metamodels for Simulation Input-Output Relations". In *Proceedings of the 1992 Winter Simulation Conference*, edited by J. J. Swain, D. Goldsman, R. C. Crain and J. R. Wilson, 289-299. Piscataway, New Jersey: Institute of Electrical and Electronics Engineers, Inc.

Barton, R. R. 2013. "Designing Simulation Experiments". In *Proceedings of the 2013 Winter Simulation Conference*, edited by R. Pasupathy, S.-H. Kim, A. Tolk, R. Hill, and M. E. Kuhl, 342-353. Piscataway, New Jersey: Institute of Electrical and Electronics Engineers, Inc.

Barton, R. R., and M. Meckesheimer. 2006. "Metamodel-Based Simulation Optimization". In *Simulation*, ed. S. G. Henderson and B. L. Nelson. Vol. 13, 535-574. Handbooks in Operations Research and Management Science. New York: Elsevier.

Barton, R. R., B. L. Nelson, and W. Xie. 2014. "Quantifying Input Uncertainty via Simulation Confidence Intervals". *INFORMS Journal on Computing* 26(1): 74-87.

Barton, R. R. 2009. "Simulation Optimization Using Metamodels". In *Proceedings of the 2009 Winter Simulation Conference*, edited by M. D. Rossetti, R. R. Hill, B. Johansson, A. Dunkin and R. G. Ingalls, 230-238. Piscataway, New Jersey: Institute of Electrical and Electronics Engineers, Inc.

Batur, D., J. M. Bekki, and X. Chen. 2018. "Quantile Regression Metamodeling: Toward Improved Responsiveness in the High-Tech Electronics Manufacturing Industry". *European Journal of Operational Research* 264(1): 212-224.

Blanning, R. W. 1974. "The Sources and Uses of Sensitivity Information". *INFORMS Journal on Applied Analytics* 4(4): 32-38.

Blanning, R. W. 1975. "The Construction and Implementation of Metamodels". *Simulation* 24: 177-184.

Burdick, D. S., and T. H. Naylor. 1966. "Design of Computer Simulation Experiments for Industrial Systems". *Communications of the ACM* 9(5): 329-339.

Chen, X., B. E. Ankenman, and B. L. Nelson. 2012. "The Effects of Common Random Numbers on Stochastic Kriging Metamodels". *ACM Transactions on Modeling and Computer Simulation* 22(2): 7, 1-20.

Dancik, G. 2018. mlegp: An R Package for Gaussian Process Modeling and Sensitivity Analysis. https://cran.r-project.org/web/packages/mlegp/vignettes/mlegp.pdf, accessed 27th April 2020.

Dellino, G., J. P. C. Kleijnen, and C. Meloni. 2009. "Robust Simulation-Optimization Using Metamodels". In *Proceedings of the 2009 Winter Simulation Conference*, edited by M. D. Rossetti, R. R. Hill, B. Johansson, A. Dunkin and R. G. Ingalls, 540-549. Piscataway, New Jersey: Institute of Electrical and Electronics Engineers, Inc.

Dunke, F., and S. Nickel. 2020. "Neural Networks for the Metamodeling of Simulation Models with Online Decision Making". *Simulation Modelling Practice and Theory* 99(102016): 1-18.

Erickson, C. B., B. E. Ankenman, and S. M. Sanchez. 2018. "Comparison of Gaussian Process Modeling Software". *European Journal of Operational Research* 266(1): 179-192.

Fonseca, D. J., D. O. Navaresse, and G. P. Moynihan. 2003. "Simulation Metamodeling through Artificial Neural Networks". *Engineering Applications of Artificial Intelligence* 16(3): 177-183.
Friedman, L. W. 1996. *The Simulation Metamodel*. New York: Springer.
Gramacy, R. B., and H. K. H. Lee. 2012. "Cases for the Nugget in Modeling Computer Experiments". *Statistics and Computing* 22(3): 713-722.
Kleijnen, J. P. C. 1975. "A Comment on Blanning's 'Metamodel for Sensitivity Analysis: The Regression Metamodel in Simulation'". *INFORMS Journal on Applied Analytics* 5(3): 21-23.
Kleijnen, J. P. C. 2008a. *Design and Analysis of Simulation Experiments*. Berlin: Springer.
Kleijnen, J. P. C. 2008b. "Design of Experiments: Overview". In *Proceedings of the 2008 Winter Simulation Conference*, edited by S. J. Mason, R. R. Hill, L. Mönch, O. Rose, T. Jefferson, J. W. Fowler, 479-488. Piscataway, New Jersey: Institute of Electrical and Electronics Engineers, Inc.
Kleijnen, J. P. C. 2017. "Regression and Kriging Metamodels with Their Experimental Designs in Simulation: A Review". *European Journal of Operational Research* 256(1): 1-16.
Kleijnen, J. P. C., and R. G. Sargent. 2000. "A Methodology for Fitting and Validating Metamodels in Simulation". *European Journal of Operational Research* 120(1): 14-29
Kleijnen, J. P. C., and W. C. M. van Beers. 2013. "Monotonicity-Preserving Bootstrapped Kriging Metamodels for Expensive Simulations". *Journal of the Operational Research Society* 64(5): 708-717.
Law, A. M. 2014. *Simulation Modeling and Analysis*. 5th ed. New York: McGraw Hill Higher Education.
Law, A. M. 2017. "A Tutorial on Design of Experiments for Simulation Modeling". In *Proceedings of the 2017 Winter Simulation Conference*, edited by W. K. V. Chan, A. D'Ambrogio, G. Zacharewicz, N. Mustafee, G. Wainer, and E. Page, 550-564. Piscataway, New Jersey: Institute of Electrical and Electronics Engineers, Inc.
Li, R., and D. K. J. Lin. 2003. "Analysis Methods for Supersaturated Design: Some Comparisons". *Journal of Data Science* 1: 249-260.
Lin, D. K. J. 1993. "A New Class of Supersaturated Designs". *Technometrics* 35(1): 28-31.
Liu, H., J. Cai, and Y.-S. Ong. 2018. "Remarks on Multi-Output Gaussian Process Regression". *Knowledge-Based Systems* 144: 102-121.
Meckesheimer, M., R. R. Barton, F. Limayem, T. Simpson, and B. Yannou. 2001. "Metamodeling of Combined Discrete/Continuous Responses". *AIAA Journal* 39(10): 1950-1959.
Meckesheimer, M., A. J. Booker, R. R. Barton, and T. W. Simpson. 2002. "Computationally Inexpensive Metamodel Assessment Strategies". *AIAA Journal* 40(10): 2053-2060.
Minitab. 2020. Statistical & Data Analysis Software Package | Minitab 19. https://www.minitab.com/en-us/products/minitab/, accessed 27th April, 2020.
Montgomery, D. C. 2012. *Design and Analysis of Experiments*. 8th ed. Hoboken, New Jersey: Wiley.
Pearce, M., M. Poloczek, and J. Branke. 2019. "Bayesian Optimization Allowing for Common Random Numbers". http://arxiv.org/abs/1910.09259, accessed 27th April, 2020.
RStudio. 2020. RStudio | Open Source & Professional Software for Data Science Teams. https://rstudio.com/, accessed 27th April, 2020.
Sanchez, S. M., and P. J. Sánchez. 2005. "Very Large Fractional Factorial and Central Composite Designs". *ACM Transactions on Modeling and Computer Simulation* 15(4): 362-377.
Sanchez, S. M., P. J. Sánchez, and H. Wan. 2018. "Work Smarter, Not Harder: A Tutorial on Designing and Conducting Simulation Experiments". In *Proceedings of the 2018 Winter Simulation Conference*, edited by M. Rabe, A. A. Juan, N. Mustafee, A. Skoogh, S. Jain, and B. Johansson, 237-251. Piscataway, New Jersey: Institute of Electrical and Electronics Engineers, Inc.
Schruben, L. W., and B. H. Margolin. 1978. "Pseudorandom Number Assignment in Statistically Designed Simulation and Distribution Sampling Experiments". *Journal of the American Statistical Association* 73(363): 504-520.
Swain, J. J. 2019. "2019 Simulation Software Survey". *ORMS Today* 42(5): 36-49.
Tew, J. D., and J. R. Wilson. 1992. "Validation of Simulation Analysis Methods for the Schruben-Margolin Correlation-Induction Strategy". *Operations Research* 40(1): 87-103.
Wan, H., B. E. Ankenman, and B. L. Nelson. 2009. "Improving the Efficiency and Efficacy of Controlled Sequential Bifurcation for Simulation Factor Screening". *INFORMS Journal on Computing* 22(3): 482-492.

AUTHOR BIOGRAPHY

RUSSELL BARTON is Distinguished Professor of Supply Chain and Information Systems in the Smeal College of Business and Professor of Industrial Engineering at the Pennsylvania State University. His research interests include applications of statistical and simulation methods to system design and to product design, manufacturing and delivery. His email address is rbarton@psu.edu.