Cases for the nugget in modeling computer experiments

Robert B. Gramacy*
Booth School of Business
University of Chicago
rbgramacy@chicagobooth.edu

Herbert K. H. Lee
Applied Math & Statistics
University of California, Santa Cruz
herbie@ams.ucsc.edu

Abstract

Most surrogate models for computer experiments are interpolators, and the most common interpolator is a Gaussian process (GP) that deliberately omits a small-scale (measurement) error term called the *nugget*. The explanation is that computer experiments are, by definition, "deterministic", and so there is no measurement error. We think this is too narrow a focus for a computer experiment and a statistically inefficient way to model them. We show that estimating a (non-zero) nugget can lead to surrogate models with better statistical properties, such as predictive accuracy and coverage, in a variety of common situations.

Key words: computer simulator, surrogate model, Gaussian process, interpolation, smoothing

1 Introduction

To some, interpolation is *the* defining feature that distinguishes surrogate models (or emulators) for computer experiments from models for ordinary experiments. We think this is old-fashioned at best and misguided at worst. It is certainly true that a large swath of computer experiments are "deterministic", in the sense that once $y(x)$ is known there can

*Part of this work was done while RBG was at the Statistical Laboratory, University of Cambridge

be no uncertainty in the output $Y(x')$ if $x' = x$, because the simulator does not behave stochastically. Interpolation would seem natural in this case, and this is typically facilitated by a zero nugget in a Gaussian process (GP) prior for $Y(x)$. Our first observation is that many of the more recent computer experiments are indeed stochastic. A typical formulation is as an agent based model or finite element simulation where the purpose is to study cohort/community effects in independent organisms/agents whose behavior is governed by simple stochastic rules which cannot be understood analytically. It is in this sense that the defining feature of zero-nugget GPs for computer experiments is old-fashioned. Many computer experiments these days are not deterministic, so in those cases you would include a nugget without hesitation. The definition of *surrogate model* for a computer experiment needs to be updated.

But that is not what this paper is really about. We shall concentrate on those computer experiments that really are "deterministic"—in a sense similar to its usage above but whose decomposition of meaning in modern experiments is one of the main foci of this paper—and argue that you should use a nugget anyway. Our arguments for this are not computational, although the numerical instabilities of zero-nugget models are well-documented (Ababou et al., 1994; Neal, 1997). Another established criticism of zero-nugget models, upon which we will not focus, involves theoretical aspects of smoothness and derivatives. Stein (1999, pp. 96) proves that the smoother the spatial process, the smaller any error or variability needs to be in order for it to have negligible effect. Since the standard assumption in the computer modeling literature is a Gaussian correlation function, this assumption of infinite differentiability means that the results are highly sensitive to any possible deviations and thus Stein strongly cautions against omitting a nugget term.

As larger nugget values can impact the fitted values of other parameters (Gramacy and Lee, 2008b; Pepelyshev, 2010), some authors go to great lengths to reconcile numerical stability and zero-nugget-like interpolation, usually by using as small a nugget as possible (Ranjan

et al., 2010). Instead, we argue that issues of numerical stability, while they are strong arguments in favor of a nugget, are a bit of a red herring in the face of more serious conceptual issues. We aim to separate the ideology of forcing interpolation from some important (and undesirable) consequences of the zero-nugget model. We shall argue that when the data are sparse or when model assumptions are violated (e.g., stationarity)—and they typically are—the nugget is crucial for maintaining good statistical properties for the emulator (e.g., coverage). Essentially, when modeling computer experiments, we must be pragmatic about how assumptions map to conclusions (surrogate model fits), and this leads us to conclude that the most sensible default is to estimate a (nonzero) nugget.

The remainder of the paper is outlined as follows. We conclude this section with a brief review of GP basics, with further reference to their application as surrogate models for computer experiments. In Section 2 we elaborate on several conceptual problems with the zero-nugget approach. Section 3 provides numerical examples, showing how sparseness of the sample (Section 3.1) or violations of standard (and uncheckable) assumptions (Section 3.2) can lead to inferior predictive surfaces with the zero-nugget approach. The issue of "determinism" is explored in Section 3.3 to similar effect. And in Section 4 we revisit these points on a real-world computer experiment involving CFD simulations of a rocket booster re-entering the atmosphere. Finally, we conclude with a discussion.

1.1 GP basics

The canonical choice of surrogate model for computer experiments is the stationary Gaussian process (Sacks et al., 1989; O'Hagan et al., 1999; Santner et al., 2003), which defines a random process whose evaluation at any finite collection of locations has a multivariate Gaussian distribution with a specified mean and covariance function that depend only on the relative positions of the locations. A typical specification of the covariance function is the Gaussian correlation (used in all the references above), so that the covariance between any two points

is

$$C(\mathbf{x}_j, \mathbf{x}_k) = \sigma^2 K(\mathbf{x}_j, \mathbf{x}_k) = \sigma^2 \exp\left\{ -\sum_{i=1}^{m} \frac{|x_{ij} - x_{ik}|^2}{d_i} \right\},$$

where m is the dimension of the space and $\mathbf{d}$ is a vector (the range parameter) which scales the correlation length in each dimension. This model will interpolate the data, fitting a smooth curve between observed outputs of the computer simulator. When all elements of $\mathbf{d}$ are equal, the process is called *isotropic*.

An extension of this model is to include a nugget term in the model, specifying the covariance function as

$$C(\mathbf{x}_j, \mathbf{x}_k) = \sigma^2 K(\mathbf{x}_j, \mathbf{x}_k) = \sigma^2 \left[\exp\left\{ -\sum_{i=1}^{m} \frac{|x_{ij} - x_{ik}|^2}{d_i} \right\} + g\delta_{j,k} \right],$$

where $\delta_{\cdot,\cdot}$ is the Kronecker delta function and g is the nugget term. Originally introduced to model small-scale variation in geostatistical models, it is also mathematically equivalent to the inclusion of a random noise term in the likelihood. Thus with $g > 0$, this model no longer interpolates the data, and returns us to a situation analogous to fitting a mean function with noisy data.

In this paper we happen to take a Bayesian approach, but all of our arguments hold true under the frequentist paradigm as well. The implementation of our GP models is in R using the GP code from the `tgp` library from CRAN (Gramacy, 2007).

2 Examining the model assumptions

Most papers in the literature obsess on the zero-nugget model. When a nugget is needed for computational reasons, one aims to make it as small as possible while still maintaining numerical stability. The argument is that the closer the nugget is to zero, the more accurate the surrogate model approximation is to the computer code output. This may be true if

there is sufficient data, but is it even the right thing to be worried about? The measurement error captured by the nugget (which is presumed to be zero for deterministic computer simulations) is but one of many possible sources of error. Here we discuss four such sources of uncertainty which are likely to be of greater importance, so it is boggling why so much attention is paid to the nugget.

Simulator bias

No computer simulator is a perfect representation of the real world. All simulators are mathematical models and thus only approximate the real world, so they have some "bias". How we deal with this discrepancy depends on whether or not real world data are available. We take those two cases in turn.

When real data are available, it is well-established that the simulator can be calibrated using the data, that is, the discrepancy between the simulator and reality can be modeled using an additional Gaussian process (Kennedy and O'Hagan, 2001; Santner et al., 2003). While this addresses the simulator bias, it introduces a source of noise—that of the real data. Because all real world data are noisy, and because the data are being used to calibrate the simulator, it is no longer necessary for the model to interpolate the simulator exactly. A measurement error term in the likelihood can be shown to be a re-parameterization of a nugget term in the covariance function (Gramacy, 2005, Appendix B). If our model is equivalent to using a nugget, then we might as well embrace the nugget while fitting the model.

If real data are not available, then the bias cannot be estimated, and that term is typically ignored and swept under the rug. Yet pretending that the simulator is perfect, even though we know it is not is clearly ignoring a major source of error. Rather than insist that the statistical model interpolate the simulator, why not allow the model to smooth the simulator output? Just because the simulator errors cannot be quantified does not mean they should

be ignored completely.

The stationarity assumption

Nearly every analysis in the computer modeling literature makes an assumption of stationarity, second-order stationarity, or at least piece-wise stationarity. Typically there is not enough data available to fit a fully nonstationary model, and if there is enough data, then the model becomes too difficult to fit efficiently. While stationarity is often a reasonable assumption because it is a close approximation to the truth, in most cases it will not be exactly correct. Like the bias case, when there is unknown error, a general statistical principle is that smoothing (or shrinking) can give better results. Thus a nugget can help protect us in the case of moderate deviations from stationarity, which would be hard to detect in practice. In Section 3.2 we show that even minor violations in the stationarity assumption lead to emulators with poor statistical properties.

Correlation assumptions

There is an underlying assumption that the specified (typically Gaussian) correlation structure is correct. While this is a nice modeling assumption, it is yet another convenient approximation to reality. Parameters for the form of the correlation function can be difficult to fit in practice, and so it is often necessary to simply specify a reasonable guess. Since it is only an approximation, this is a further reason for allowing smoothing in the model.

The assumption of a deterministic simulator

The modeling assumptions addressed above may indeed be reasonable for a particular true physical process, but the implementation of the computer simulation may still behave in unpredictable ways. The assumption of a deterministic simulator may itself be a problem. Here we discuss two related possible issues, nonmodelable determinism and theoretical but not

numerical determinism, among other possible problems with the assumption of deterministic behavior in practice.

Some deterministic functions really are better treated as nondeterministic. As a simple example, consider a pseudo-random number generator where, for any given seed, a result is returned deterministically (if not also unpredictably unless you know a lot about numerical analysis). A version of a computer simulator approximating a function $g(x)$ numerically might effectively behave as follows (coded in R):

```
f <- function(x) {
  set.seed(x)
  return(g(x) + rnorm(1))
}
```

This function is theoretically deterministic, but knowing the true function, it would be irrational to interpolate it. Clearly one would want to smooth out the pseudo-random noise and just fit the underlying $g(x)$. In such a case, a nugget is needed even for this "deterministic" simulator. Two related examples would be a function with chaotic behavior, which can happen in complex systems of differential equations, or the Perlin noise function (Perlin, 2002), which is a deterministic method of generating random-looking smooth surfaces in computer graphics. Alternatively, the `rnorm(1)` term may stand in for the amount by which an iterative approximation algorithm steps over the convergence threshold, but usually we assume that this amount can be made to be arbitrarily small.

Now, this may seem pathological, but in Sections 3.3 & 4 we give a synthetic and real example, respectively, which are essentially the following adaptation:

```
f2 <- function(x) {
  set.seed(x)
  y <- runif(1)
  if(y < 0.9) return(g(x))
  else return(h(x))
}
```

for some new `h(x)`. The pseudo-random (but deterministic) `y` is intended to represent the chance that the computer code was (poorly) initialized such that it may end up converging to a sub-optimal (but locally converged) solution `h(x)` 10% of the time rather than the true globally converged approximation to $g(x)$. This is not an uncommon feature of a modern computer simulator, i.e., where the final output depends upon an initial "solution" for which there are defaults that usually work, but sometimes lead to a converged solution which is different from the one intended. It is clearly sub-optimal to use a zero-nugget model in this case, because some of the outputs are not the correct values. Despite their deterministic nature, we show in Section 3.3 that the uncertainty about the true function is best modeled with a random process that smooths rather than interpolates.

3 Statistically better fits with the nugget

3.1 Protecting against misfits with sparse data

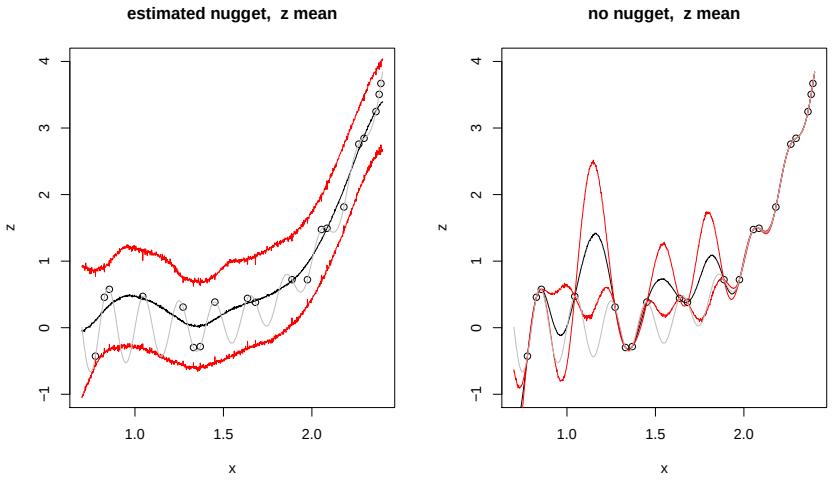
Many computer experiments are expensive to run and the number of datapoints is limited. As many experiments have higher dimensional input spaces, the curse of dimensionality implies that the data will be sparse in the input region. When the data are sparse, interpolation can have unpleasant results (Taddy et al., 2008, Sec. 2.2). We present here a simulated example where the data are sparse in one dimension, but this represents the concept of sparseness in higher dimensions with a simpler function or with more data.

Consider the function

$$Z = \frac{\sin(10\pi X)}{2X} + (X-1)^4 .$$

Suppose we only have 20 datapoints available (in practice, we would have more points but more dimensions). We randomly generated 10000 such datasets (with X generated from a uniform distribution each time) and fit models with both a nugget and without a nugget.

The table in Figure 1 gives the distribution of the mean square errors of fits under each model, and the model that includes a nugget does better on average (a paired t-test gives a p-value of less than 2.2×10^{-16}). The plots in Figure 1 show one of the runs. The data are too sparse to get a good fit of the function for smaller input values. While the nugget model smooths and produces reasonable confidence bands, in order to interpolate smoothly the no-nugget model ends up making predictions well outside the range of the actual data in that region, and its confidence bands are all over the place. Using a nugget gives a much more sensible fit. This sort of problem can be quite difficult to diagnose in higher dimensions, and the nugget provides good protection against the strange fits that interpolation can produce.

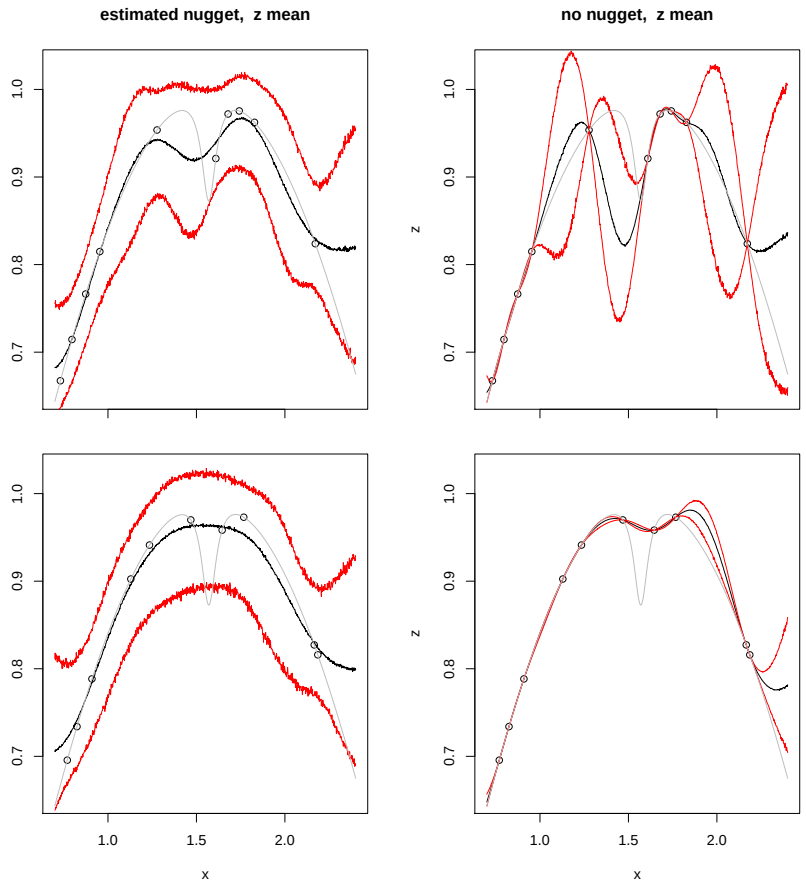


MSE	nug	nonug
Min.	0.0250	0.0057
1st Qu.	0.0999	0.0851
Median	0.1262	0.1399
Mean	0.1847	0.1929
3rd Qu.	0.1906	0.2290
Max.	1.2990	1.3510

Figure 1: Plots of one example of a fit (dark solid line), confidence bands (red), and true function (light grey). The *left* plot shows the fit using a nugget, the *right* plot without a nugget. The *table on the right* is the summary of the mean square errors under both models for 10,000 repeated uniform designs.

3.2 Poor coverage

In fact, the nugget offers protection from a slew of problematic scenarios. Here we shall illustrate the that no-nugget model under-covers the true computer simulator response when the stationarity assumption is not satisfied. We use three examples.



coverage	nug	nonug
Min.	0.3210	0.0650
1st Qu.	0.8028	0.5108
Median	0.8915	0.7230
Mean	0.8517	0.6531
3rd Qu.	0.9570	0.8068
Max.	1.0000	0.9960

Figure 2: The *plots on the left* are examples of fits under under two uniform designs in nugget (left column) and no-nugget (right column) models. The *table on the right* is the summary of the coverages under both models for 100 repeated uniform designs.

The first example is a 1-d function which is clearly nonstationary, but otherwise mimics typical features of a computer code. The response is given by $y(x) = \sin(x) - 0.02 \cdot t_1(x, 1.57, 0.05)$ where $t_1(\cdot, \mu, \sigma)$ is a Cauchy density with mean μ and spread σ. The two rows of Figure 2 show fits for two typical random uniform designs of size ten. The difference between smoothing (estimated nugget; *left panels*) and interpolation (no nugget; *right panels*) is clear. We see that the no-nugget model under-covers the truth (in gray) and can have wildly different (i.e., narrow or wide) 90% predictive credible intervals. This experiment was repeated 100 times and the percentage of the area of the input space where $y(x)$ was covered

by the 90% interval was recorded. A table showing the results is on the right in the figure. We see from these results that the under-coverage of the no-nugget model is drastic. For one of the random designs it only covered 6.5% of $y(x)$ and 3/4 of the trials under-covered by more than 10%. By contrast, the model which estimates a nugget has good coverage properties. Its median and mean coverages are close to 90% and the central 50% region tightly brackets the truth. Clearly, connecting the dots comes at the expense of other, arguably more important, statistical measures of goodness of fit.

	exp data	
coverage	nug	nonug
Min.	0.5479	0.3965
1st Qu.	0.8623	0.8242
Median	0.9185	0.8936
Mean	0.8962	0.8691
3rd Qu.	0.9492	0.9395
Max.	1.0000	1.0000

	fried data	
coverage	nug	nonug
Min.	0.5480	0.4580
1st Qu.	0.8930	0.8350
Median	0.9320	0.8890
Mean	0.9205	0.8762
3rd Qu.	0.9580	0.9310
Max.	0.9990	1.0000

Table 1: *Left* is coverage for the 2-d exponential data; *right* for the 5-d Friedman data.

We performed similar experiments on two higher-dimensional data sets. The first is a 2-d exponential function $y(x) = x_1 \exp\{-x_1^2 - x_2^2\}$, which less clearly violates the stationarity assumption. From the *left* side of Table 1 we see a similar under-coverage of the no-nugget model with repeated uniform designs of size 20. Our second experiment involved the first Friedman data function (Friedman, 1991) with five inputs where the response is $y(x) = 10\sin(\pi x_1 x_2) + 20(x_3 - 0.5)^2 + 10x_4 + 5x_5$. This function is better behaved (i.e., stationarity may be a reasonable assumption). However, it is apparent that correlation in the response would decay at different rates along the five coordinates—clear anisotropy. To illustrate how the effect of an inappropriate choice of correlation function is felt more strongly in the no-nugget model we used an isotropic Gaussian correlation function and uniform designs of size 25. The results are shown in the *right* in the table.

It is worth pointing out that as the size of the designs are increased, and/or as the

data less obviously violate assumptions, both models (nugget and no-nugget) will tend to over cover in practice. This is because we are fitting a model (GP) which always yields positive posterior predictive error away from the (discrete set of) design points, i.e., over an uncountably large region. Since the function we are modeling is deterministic, we know that as the size of the design tends to (countable) infinity we should be able to obtain a "perfect" fit with a high degree polynomial. So a GP is the wrong model in this case. Since over-coverage is inevitable, under-coverage should be our primary concern, and to avoid under-covering we can see that a nugget is needed.

3.3 Challenging determinism in computer simulation

Some computer experiments are deterministic in a technical sense, but not necessarily in a way that translates into sensible assumptions for the building of a surrogate model. We may reasonably presume that codes implementing the algorithms and calculations behind the experiment are nontrivial. They are expensive to program and expensive to execute, requiring long iterations to convergence and the (sometimes arbitrary) specification of tuning parameters and grid/mesh sizes. As a rule more than an exception, the resulting apparatus works better for some choices of inputs than for others. The most important issue is in detecting global convergence of the code, whose properties usually depend crucially on other implementation choices. It is essentially impossible to guarantee good global convergence properties, and so this the main target of our attack on the modeling of such "deterministic" computer simulations without a nugget.

Consider the following computer simulator coded in R below.

```
## 2-d function
f2d <- function(x1, x2) {
  w <- function(y) {
    return(exp(-(y-1)^2) + exp(-0.8*(y+1)^2) - 0.05*sin(8*(y+0.1)))
  }
```

```
  return(-w(x2)*w(x1))
}

## find the minimum of a projection of the 2-d function
f <- function(x) {
  return(optim(par=x, fn=f2d, x2=x)$value)
}
```

The true underlying function $f(x)$, evaluated by `f(x)` in R, is $\arg\min_{x_1} f(x_1, x)$ where

$$f(x_1, x_2) = -w(x_1)w(x_2), \qquad \text{where}$$
$$w(y) = \exp\left(-(y-1)^2\right) + \exp\left(-0.8(y+1)^2\right) - 0.05\sin\left(8(y+0.1)\right).$$

The optimization method used by the code above is the `optim` function in R initialized at $x_1 = x$.

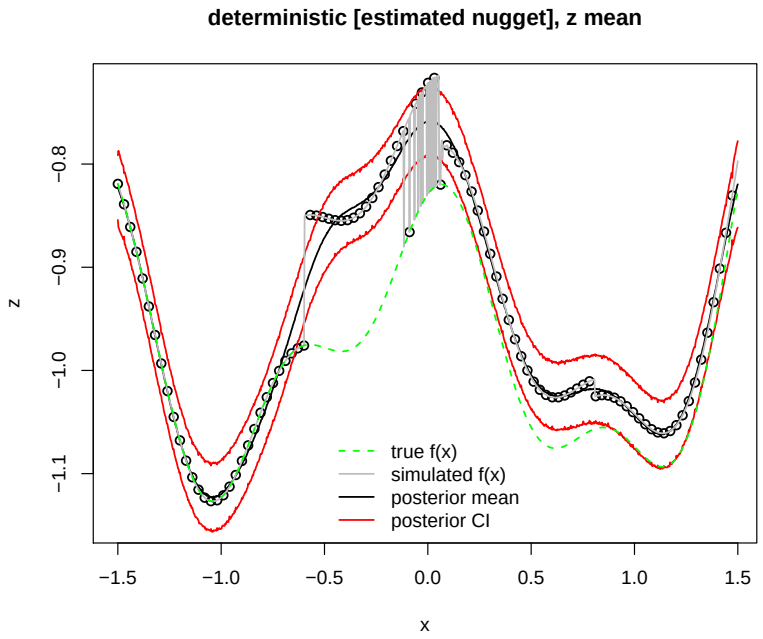


Figure 3: GP fit (with an estimated nugget) to a deterministic function which is the result of an iterative procedure.

Figure 3 shows the true $f(x)$ (dashed-green) and the output of the simulator `f(x)` (gray)

for $x \in [-1.5, 1.5]$. We can see that result of numerically finding the optimal value of the objective function (initialized somewhat arbitrarily, but not pathologically) is that the simulations `f(x)` are biased, and behave badly/unpredictably in some parts of the input space. It is worth noting that both behaviors persist with a different static initialization scheme; the $f(x_1, x_2)$ surface has about a dozen local minima. Also, the implementation `f(x)` is completely deterministic in a technical sense. However, `f(x)` is exhibiting "random" behavior of the sort alluded to in Section 2 as the initialization scheme causes the algorithm to converge to different local minima in a way that is not (easily) predictable. There are three places where the initialization causes it to have discontinuities (even though the true $f(x)$ is smooth everywhere), and it is particularly unstable near $x = 0$ since $(0, 0)$ is more or less equidistant from the many local minima of $f(x_1, x_2)$ in the 2-d space.

The figure also shows a fit to the computer simulator (`f(x)`) output obtained from a gridded design of 100 input output pairs using a GP with an estimated nugget. The fit is sensible given the discontinuities and otherwise "noisy" behavior of the simulator. It is not possible to fit this data without a nugget, or even with a small one, due to numerical instabilities. However, it is possible to do so with a reduced design of about 20 points or so. To connect with the coverage results in the last section (where stationarity was the issue) we

coverage	nug	nonug
Min.	0.433	0.2280
1st Qu.	0.787	0.6665
Median	0.875	0.7345
Mean	0.846	0.7276
3rd Qu.	0.938	0.8362
Max.	0.993	0.9760

Table 2: Coverage of `f(x)` for the "deterministic" data.

calculated the coverage of `f(x)` with 100 repeated uniform random designs of size 20 under the estimated nugget and no-nugget models, and the story is much the same as before. The results are shown in Table 2. Therefore, when "determinism" is challenged as an assumption

on the nature of the data-generating mechanism, a nugget for smoothing is clearly preferred to interpolation in the surrogate model.

4 A modern computer experiment

The Langley Glide-Back Booster (LGBB) is a rocket booster that underwent design phases at NASA primarily through the use of computational fluid dynamics simulators that numerically solve the relevant inviscid Euler equations over a mesh of 1.4 million cells (Rogers et al., 2003). The simulator models the forces felt by the rocket at the moment it is re-entering the atmosphere as a function of three inputs describing its state: speed (measured by Mach number), angle of attack (the alpha angle), and sideslip angle (the beta angle). As a free body in space, there are six degrees of freedom, so the six relevant forces/outputs are lift, drag, pitch, side-force, yaw, and roll. While theoretically deterministic, the simulator can fail to converge. Some nonconvergent runs are caught by an automated checker, and re-run with a new schedule of initial conditions, but some are erroneously accepted even after converging to a clearly inferior local mode. Input configurations arbitrarily close to one another can fail to achieve the same estimated convergence, even after satisfying the same stopping criterion.

Here we focus on the roll force output on a data set comprised of simulator runs at 3041 locations. See Figure 4 for a 2-d slice of this response. Previous work has focused on the lift force (Gramacy and Lee, 2008a) which exhibited many similar features, and on a sequential design task taking account of all outputs simultaneously (Gramacy and Lee, 2009). The experimental design is a combination of an initial grid followed by two hand-designed finer grids focused around Mach one, as the initial run showed that the most interesting part of the input space was generally around the sound barrier, where the physics in the simulator changes abruptly from a subsonic regime to a supersonic regime. What happens close to and along the boundary is the most difficult part of the simulation. The regime changes across this

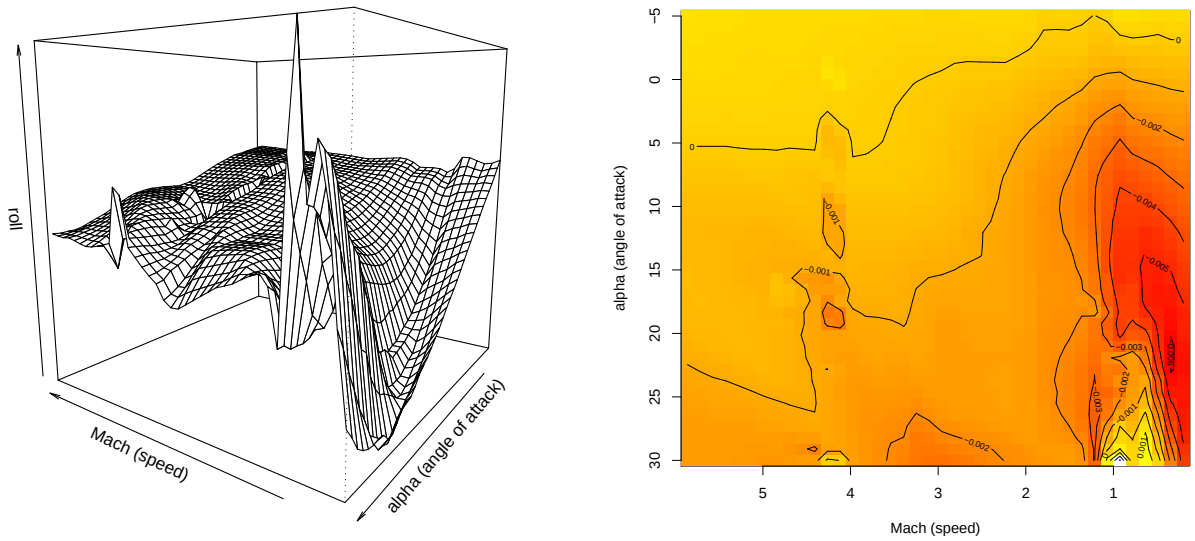


Figure 4: Linearly interpolated slice of the roll response plotted in perspective *(left)* and image/contour *(right)* as a function of speed (Mach) and angle of attach (alpha), with the slide slip angle (beta) fixed to 2. In the image plot, dark/red values are lower and light/yellow values higher in the image plot; the perspective plot is rotated for visualization purposes so that the closest corner corresponds to low speed and high angle of attack.

boundary cause the stationarity assumption to be violated. Also note the string of anomalies around Mach four, which appear to converge to local, rather than global, solutions. So this experiment comprises two challenging aspects—impractical determinism due to convergence issues and failed assumptions of stationarity due to physical regime changes—and we aim to show that the nugget is important in mitigating their effects when building a surrogate model.

Towards this end we calculated the coverages of predictive surfaces obtained with and without the nugget on a 20-fold partition of the 3041 input/output pairs. We iterated over the folds, training on $1/20^{\text{th}}$ of the data, about 159 pairs, and predicting at the remaining 3009-odd locations in an (inverse) cross-validation fashion. The results of this experiment, repeated 100 times for 2000 total coverages for each predictor, are shown on the *left* in Table 3. Note that this is not a uniform coverage rate (over the input area), since the design

	GP			TGP	
coverage	nug	nonug	coverage	nug	nonug
Min.	0.7547	0.5726	Min.	0.7627	0.5180
1st Qu.	0.9022	0.8427	1st Qu.	0.8757	0.7195
Median	0.9239	0.8793	Median	0.8978	0.7670
Mean	0.9187	0.8703	Mean	0.8954	0.7606
3rd Qu.	0.9396	0.9059	3rd Qu.	0.9186	0.8051
Max.	0.9777	0.9741	Max.	0.9771	0.9305

Table 3: Coverage of the roll response for the LGBB computer experiment data using a Gaussian process *(left)* and a treed Gaussian process *(right)*.

is more heavily concentrated around Mach one. However, the results here are as expected. The no-nugget model can severely under-cover in certain examples (with coverage as low as 57%) and, gives the target coverage of 90% less than 1/4 of the time. The model using an estimated nugget is much better behaved. However, it does seem to slightly over-cover. Although less of a concern, we think that the main cause of this is the nonuniformity of the design and our choice of priors for *both* the range and nugget parameters in the face of both nonstationarity and nonconvergence issues.

The treed Gaussian process (TGP, Gramacy and Lee, 2008a) model was designed to handle the axis-aligned nonstationarity that arises due to regime changes—exactly the sort exhibited by this data. In essence, the TGP model learns an axis-aligned partition of the data wherein the process is well-fit by separate stationary GP models. We performed an identical experiment using TGP and the results are summarized on the *right* in Table 3. We can see that the coverage of the version of TGP which estimates the nugget is improved (with better centering around 90%), but the no-nugget version is not (showing a more consistent tendency to undercover). We are left with the impression that the nugget is even more important when a nonstationary model is used, especially in the case of nonconvergent computer experiments where the assumption of "determinism", while technically valid, may be challenged from a practical standpoint.

5 Discussion

Several authors have previously argued in favor of a nugget term for reasons of numerical stability even when fitting a deterministic model. We go well beyond numerical convenience, raising fundamental issues of a variety of modeling assumptions and argue that the use of a nugget helps protect against many violations of assumptions. The focus on the assumption of determinism can be too single-minded. Ignoring the impacts of the many other modeling assumptions can lead to poor statistical properties of the surrogate model, that can be ameliorated with the use of a nugget.

Acknowledgments

We thank Tim Gustafson for introducing us to the Perlin function. This work was partially supported by National Science Foundation grant DMS-0906720 and EPSRC grant EP/D065704/1.

References

Ababou, R., Bagtzoglou, A. C., and Wood, E. F. (1994). "On the Condition Number of Covariance Matrices in Kriging, Estimation, and Simulation of Random Fields." *Mathematical Geology*, 26, 1, 99–133.

Friedman, J. H. (1991). "Multivariate Adaptive Regression Splines." *Annals of Statistics*, 19, No. 1, 1–67.

Gramacy, R. B. (2005). "Bayesian Treed Gaussian Process Models." Ph.D. thesis, University of California, Santa Cruz.

— (2007). "`tgp`: An `R` Package for Bayesian Nonstationary, Semiparametric Nonlinear

Regression and Design by Treed Gaussian Process Models." *Journal of Statistical Software*, 19, 9.

Gramacy, R. B. and Lee, H. K. H. (2008a). "Bayesian treed Gaussian process models with an application to computer modeling." *Journal of the American Statistical Association*, 103, 1119–1130.

— (2008b). "Gaussian Processes and Limiting Linear Models." *Computational Statistics and Data Analysis*, 53, 123–136.

— (2009). "Adaptive Design and Analysis of Supercomputer Experiment." *Technometrics*, 51, 2, 130–145.

Kennedy, M. and O'Hagan, A. (2001). "Bayesian Calibration of Computer Models (with discussion)." *Journal of the Royal Statistical Society, Series B*, 63, 425–464.

Neal, R. M. (1997). "Monte Carlo implementation of Gaussian process models for Bayesian regression and classification." Tech. Rep. 9702, Deptartment of Statistics, University of Toronto.

O'Hagan, A., Kennedy, M. C., and Oakley, J. E. (1999). "Uncertainty Analysis and Other Inference Tools for Complex Computer Codes." In *Bayesian Statistics 6*, eds. J. M. Bernardo, J. O. Berger, A. Dawid, and A. Smith, 503–524. Oxford University Press.

Pepelyshev, A. (2010). "The Role of the Nugget Term in the Gaussian Process Method." In *MODA 9 – Advances in Model-Oriented Design and Analysis*, 149–156. Berlin: Springer-Verlag.

Perlin, K. (2002). "Improving Noise." *ACM Transactions on Graphics*, 21, 681–682.

Ranjan, P., Haynes, R., and Karsten, R. (2010). "Gaussian Process Models and Interpolators for Deterministic Computer Simulators." Department of Mathematics and Statistics, Acadia University.

Rogers, S. E., Aftosmis, M. J., Pandya, S. A., N. M. Chaderjian, E. T. T., and Ahmad, J. U. (2003). "Automated CFD Parameter Studies on Distributed Parallel Computers." In *16th AIAA Computational Fluid Dynamics Conference*. AIAA Paper 2003-4229.

Sacks, J., Welch, W. J., Mitchell, T. J., and Wynn, H. P. (1989). "Design and Analysis of Computer Experiments." *Statistical Science*, 4, 409–435.

Santner, T. J., Williams, B. J., and Notz, W. I. (2003). *The Design and Analysis of Computer Experiments*. New York, NY: Springer-Verlag.

Stein, M. L. (1999). *Interpolation of Spatial Data*. New York, NY: Springer.

Taddy, M., Lee, H. K. H., Gray, G. A., and Griffin, J. D. (2008). "Bayesian Guided Pattern Search for Robust Local Optimization." Tech. Rep. ams2008-02, University of California, Santa Cruz, Department of Applied Mathematics and Statistics.