



Predicting the Output From a Stochastic Computer Model When a Deterministic Approximation is Available

Evan Baker, Peter Challenor & Matt Eames

To cite this article: Evan Baker, Peter Challenor & Matt Eames (2020): Predicting the Output From a Stochastic Computer Model When a Deterministic Approximation is Available, Journal of Computational and Graphical Statistics, DOI: [10.1080/10618600.2020.1750416](https://doi.org/10.1080/10618600.2020.1750416)

To link to this article: <https://doi.org/10.1080/10618600.2020.1750416>



© 2020 The Author(s). Published with license by Taylor and Francis Group, LLC



[View supplementary material](#)



Accepted author version posted online: 02 Apr 2020.
Published online: 07 May 2020.



[Submit your article to this journal](#)



Article views: 96



[View related articles](#)



[View Crossmark data](#)

Predicting the Output From a Stochastic Computer Model When a Deterministic Approximation is Available

Evan Baker^a, Peter Challenor^a, and Matt Eames^b

^aDepartment of Mathematics, University of Exeter, Exeter, UK; ^bDepartment of Engineering, University of Exeter, Exeter, UK

ABSTRACT

Statistically modeling the output of a stochastic computer model can be difficult to do accurately without a large simulation budget. We alleviate this problem by exploiting readily available deterministic approximations to efficiently learn about the respective stochastic computer models. This is done via the summation of two Gaussian processes; one responsible for modeling the deterministic approximation, the other responsible for using such approximation to better statistically model the stochastic computer model. The developed method provides high predictive performance and increased confidence that complicated features of a stochastic computer model are captured, even when the simulation budget is small. Several synthetic computer models are used to outline the capabilities of this method, and two real-world examples are used to display its practical utility. Supplementary materials for this article are available online.

ARTICLE HISTORY

Received March 2019
Revised October 2019

KEYWORDS

Emulation; Gaussian process; Heteroscedastic; Multifidelity; Stochastic kriging; Stochastic simulation

1. Introduction

Complex real world systems can be modeled using computer models, also known as simulators, allowing experimentation to be conducted where physical experiments may be too costly or infeasible. Typically such simulators are deterministic, yielding the same output every time the simulator is run if the same input parameter values are used. Increasingly, however, simulators are becoming stochastic, including some random component in their code to account for perceived “randomness” in a system. For example, we would often not want to model a given animal population deterministically—whether, at a given time step, an individual encounters a potential mate, or a potential predator, cannot feasibly be represented without stochasticity. Running such simulators with the same input parameter values thus does not yield the same output value each time. Instead there is some intrinsic variance in the computer model output.

Simulators are often computationally expensive, and so to facilitate analysis, a statistical approximation of these models is used. Known as emulators, these use previously obtained values from the simulator to obtain fast predictions for new outputs of the simulator.

For deterministic models, the Gaussian process emulator is a common choice of emulator (O’Hagan 2006). The principal idea is that the outputs from the computer model $\eta(\cdot)$ are treated as a realization from a Gaussian process with a prior mean function $m(\cdot)$ and a prior covariance function $K(\cdot, \cdot)$. Predictions can then be obtained by conditioning on data $\mathbf{y} = \eta(X)$ (Rasmussen and Williams 2006).

This framework can be extended to model stochastic simulators by the addition of independent noise with variance $\delta(\cdot)$.

If the simulator is believed to be homoscedastic, this can simply be a constant. Alternatively, if the homoscedastic assumption is too strong, one can also model the (log) variance with another Gaussian process (Goldberg, Williams, and Bishop 1998; Boukouvalas and Cornford 2009; Binois, Gramacy, and Ludkovski 2018). The log variance is modeled rather than simply the variance to constrain predicted values of the variance to be greater than zero.

Such a model is then very flexible, with a nonparametric form for both the mean and variance, allowing for many different stochastic simulators to be emulated. The downside to this flexibility is that far more data is required to properly estimate the form of the mean (and variance). A rule of thumb for deterministic emulators is that at least 10 data points per input dimensions are required to fit an emulator (Loeppky, Sacks, and Welch 2009), whereas Binois et al. (2019) use 500 data points when comparing different methods of choosing data point locations for a one-dimensional toy stochastic simulator.

A larger amount of data is to be expected for more complicated simulators (and stochastic simulators are certainly a more complex class of simulator), but such a high number of required runs can be prohibitive in practice. In this article, we attempt to alleviate this problem by leveraging a unique tool that computer simulators provide over physical experiments: often deterministic approximations are available. Section 2 will intuitively justify and formally present the developed method. Section 3 outlines guidelines required to ensure the method works in practice. Section 4 then applies this method to a simple agent based model and a stochastic building performance simulator. Concluding remarks are given in Section 5.

CONTACT Evan Baker  e.baker@exeter.ac.uk  Department of Mathematics, University of Exeter, Laver Building, Exeter, UK, EX4 4QE.

 Supplementary materials for this article are available online. Please go to www.tandfonline.com/r/JCGS.

© 2020 The Author(s). Published with license by Taylor and Francis Group, LLC

This is an Open Access article distributed under the terms of the Creative Commons Attribution License (<http://creativecommons.org/licenses/by/4.0/>), which permits unrestricted use, distribution, and reproduction in any medium, provided the original work is properly cited.

2. Model

Consider the toy stochastic simulator given in Equation (1).

$$\eta(x) = (1 - x) \sin(\pi + 6\pi x) + \log(0.2 + x) + (1.2 - x)\epsilon, \quad \epsilon \sim N(0, 1). \quad (1)$$

Evaluating this toy simulator on 50 points sampled from $[0, 1]$ using a maximin Latin hypercube (McKay, Beckman, and Conover 2000), and standardizing the data, we obtain the plot on the left of Figure 1. The plot on the right of Figure 1 shows predictions for the mean and the 95% predictive intervals for the simulator output using a heteroscedastic Gaussian process emulator, as well as the true mean and 95% predictive intervals.

With only the plot on the left of Figure 1, the challenge of flexibly modeling a heteroscedastic process becomes clearer. Perceived trends can be “true,” or they can also just be artifacts of the stochasticity. This makes it difficult to discern the correct shapes for the mean and variance functions of a simulator without a large number of data points. The plot on the right shows a heteroscedastic Gaussian process emulator indeed struggling to identify the true mean, with the true mean being much more detailed than the estimated mean. This could be considered a similar issue to the problem of choosing the degrees of freedom for smoothing splines (Cantoni and Hastie 2002).

To properly use an emulator as a surrogate for the simulator, this issue should be resolved, but using an excessive number of data points can be computationally expensive. In situations where additional prior knowledge of the simulator is available, a more descriptive prior mean function $m(\cdot)$ can be used, providing information that can assist in the prediction of the true mean.

In practice, sufficient knowledge of the mean function can often be lacking, which is one reason why computer experiments are conducted in the first place. However, because stochastic computer models are completely artificial, based on theoretical understanding of a real world system, modifications to the computer model are possible, and thus sometimes deterministic versions are available. This can be because a deterministic version has intentionally been made; a different model of the same process exists but is deterministic; the computer model was once deterministic in its development history and stochasticity was added to the computer model after its initial creation; or because the underlying simulator is actually deterministic but some inputs are typically taken as random. Ultimately, both the deterministic approximation and the stochastic simulator are supposed to be modeling the same real world process, and so there is reason to believe that the deterministic approximation can contain key information about the stochastic simulator. Additionally, it can be easier to extract important information from deterministic systems than it can for stochastic systems, and so it logically follows that spending some of a simulation budget on a deterministic approximation can be worthwhile.

For our toy model, we obtain a hypothetical deterministic approximation by replacing the random component ϵ with a fixed number (in this case ϵ is replaced with 1; 0 has intentionally not been chosen to avoid an overly perfect approximation). The plot on the left of Figure 2 shows 12 runs of this toy deterministic approximation simulator (chosen via a maximin Latin hypercube design).

Because these runs are now samples from a deterministic simulator, a deterministic emulator can be fit to them, and the predictions in the right of Figure 2 are obtained.

The shape of the emulator produced from this appears visually similar in shape to the true mean of the stochastic simulator from Figure 1. This is to be expected, as the true mean of

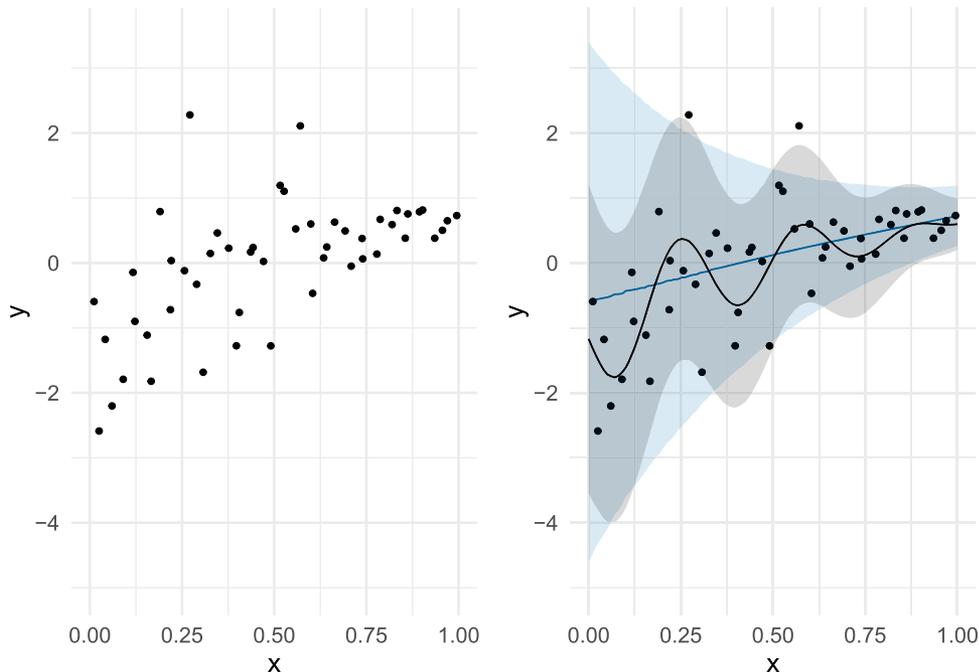


Figure 1. Fifty evaluations of the toy simulator from Equation (1) (left), and the respective heteroscedastic emulator predictions (right). The more periodic true mean and 95% interval are superimposed in black/dark gray, and the more linear emulator mean and 95% predictive interval are in blue (light gray in gray scale).

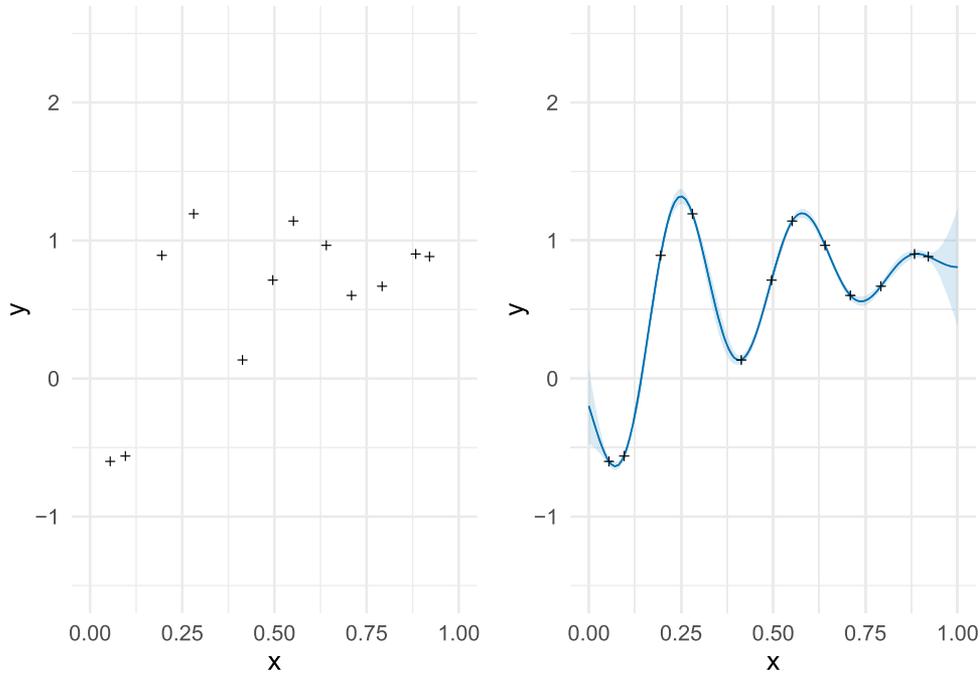


Figure 2. Ten evaluations of the toy deterministic approximation simulator from Equation (1), and the respective deterministic emulator predictions (right).

the deterministic emulator is the true mean of the stochastic simulator, offset by $(1.2 - x)$.

Visually observing the predictions from this deterministic emulator would suggest, even without knowledge of the truth, that the stochastic emulator's mean should not be (approximately) linear as it is in Figure 1. And so, we aim to formally incorporate this evidence.

We model the stochastic simulator as the sum of a deterministic Gaussian process DetGP and a heteroscedastic Gaussian process HetGP. Equations are given by Equation (2).

$$\eta(\cdot) = \text{HetGP}(\cdot) + \text{DetGP}(\cdot),$$

$$\text{where } \text{HetGP}(\cdot) \sim \text{GP}(m(\cdot), K(\cdot, \cdot) + \delta^2(\cdot)I),$$

$$\log(\delta^2(\cdot)) \sim \text{GP}(m_\delta(\cdot), K_\delta(\cdot, \cdot) + \sigma^2 I),$$

$$\text{and } \text{DetGP}(\cdot) \sim \text{GP}(m_{\text{det}}(\cdot), K_{\text{det}}(\cdot, \cdot)).$$

$m(\cdot)$, $m_{\text{det}}(\cdot)$, and $m_\delta(\cdot)$ are prior mean functions, $K(\cdot, \cdot)$, $K_{\text{det}}(\cdot, \cdot)$, and $K_\delta(\cdot, \cdot)$ are covariance functions, I is the identity matrix, $\delta^2(\cdot)$ is the heteroscedastic intrinsic simulator variance, and σ^2 is an additional variance term for the latent log variance Gaussian process. Although the methods described are completely general, in examples in this article, the mean functions are taken to be linear and the covariance functions are taken to be squared exponentials (e.g., for the HetGP Gaussian process, $m(\mathbf{x}) = \beta_0 + \mathbf{x}^T \boldsymbol{\beta}$ and $K(\mathbf{x}, \mathbf{x}') = \alpha^2 \prod_{i=1}^d \exp(-\frac{(x_i - x'_i)^2}{l_i})$, where d is the dimension of \mathbf{x} , α is a standard deviation parameter, and l_i are length scale parameters). Specific choices for a Gaussian process are often based on personal preference; some authors prefer more complicated mean functions (Vernon, Goldstein, and Bower 2010), others choose a zero mean function (Binois, Gramacy, and Ludkovski 2018), and then there are many choices for the covariance function with various justifications (Rasmussen and Williams 2006). These choices, while important, are outside the scope of this article.

The DetGP component models the deterministic approximation and is conditioned on observed deterministic approximation runs \mathbf{y}_{det} (with input values X_{det}). The HetGP component models the difference between the stochastic simulator's mean and the deterministic approximation's mean, and is thus conditioned on values of $\eta(\cdot) - \text{DetGP}(\cdot)$ for observed stochastic computer model runs \mathbf{y} (with input values X), that is, $\mathbf{y} - \text{DetGP}(X) | \mathbf{y}_{\text{det}}$. HetGP also incorporates the intrinsic variance modeled by $\delta^2(\cdot)$.

Using a sum of Gaussian processes to model a system is a common tool in deterministic emulation, having been used to include the discrepancy between the simulator and the real world (Kennedy and O'Hagan 2001; Brynjarsdóttir and O'Hagan 2014); modeling nonstationary simulators (Ba and Joseph 2012); modeling large-scale computer experiments (Haaland and Qian 2011); and modeling slow deterministic simulators when fast deterministic simulators are available (Kennedy and O'Hagan 2000).

In this article, parameters are taken fixed as their posterior modes, estimated via the optimizing function in Stan (Stan Development Team 2016). A full MCMC scheme could be used to incorporate parameter uncertainty, but this is prohibitively slow as for the base heteroscedastic Gaussian process (Kersting et al. 2007).

The priors for the linear mean function coefficients (β_0 and $\boldsymbol{\beta}$ for HetGP, β_{δ_0} and $\boldsymbol{\beta}_\delta$ for the $\log(\delta^2)$ Gaussian process, and β_{det_0} and $\boldsymbol{\beta}_{\text{det}}$ for DetGP) will be $N(0, 10)$; the standard deviation parameters (α , α_δ and α_{det} for the HetGP, $\log(\delta^2)$, and DetGP Gaussian process, respectively) will have Inverse-Gamma(2, 1) priors; and the length scales (l_i , l_{δ_i} , and l_{det_i}) will have Gamma(4, 4) priors. σ^2 will have a standard half-normal prior. Such choices can also change depending on personal preference. Our choices we argue are “semi-informative,” providing a good range of sensible values for the various parameters, based on an understanding

of what each parameter controls (Rasmussen and Williams 2006).

Predictions from this model are (comparatively) complex, requiring predictions from the latent log variance Gaussian process and the DetGP Gaussian process to be made before predictions from the HetGP Gaussian process, and thus the full model, can be made.

For the deterministic Gaussian process, predictions conditional on known deterministic runs \mathbf{y}_{det} are standard (Rasmussen and Williams 2006) and are given in Equation (3)

$$\text{DetGP}(X^*) | \mathbf{y}_{\text{det}} \sim N(\mathcal{M}_{\text{det}}(X^*), \mathcal{V}_{\text{det}}(X^*)), \quad (3)$$

where $\mathcal{M}_{\text{det}}(X^*) = m_{\text{det}}(X^*)$

$$+ K_{\text{det}}(X^*, X_{\text{det}})(K_{\text{det}}(X_{\text{det}}, X_{\text{det}})^{-1}(\mathbf{y}_{\text{det}} - m_{\text{det}}(X_{\text{det}})))$$

and $\mathcal{V}_{\text{det}}(X^*) = K_{\text{det}}(X^*, X^*)$

$$- K_{\text{det}}(X^*, X_{\text{det}})(K_{\text{det}}(X_{\text{det}}, X_{\text{det}})^{-1}K_{\text{det}}(X_{\text{det}}, X^*).$$

Predictions for the intrinsic (log) variance, conditional on (estimated) values at the input points $\delta^2(X)$ are also standard, and given by Equation (4)

$$\log(\delta^2(X^*)) | \log(\delta^2(X)) \sim N(\mathcal{M}_{\delta}(X^*), \mathcal{V}_{\delta}(X^*)), \quad (4)$$

where $\mathcal{M}_{\delta}(X^*) = m_{\delta}(X^*)$

$$+ K_{\delta}(X^*, X)(K_{\delta}(X, X) + \sigma^2 I)^{-1}(\log(\delta^2(X)) - m_{\delta}(X))$$

and $\mathcal{V}_{\delta}(X^*) = K_{\delta}(X^*, X^*) + \sigma^2 I$

$$- K_{\delta}(X^*, X)(K_{\delta}(X, X) + \sigma^2 I)^{-1}K_{\delta}(X, X^*).$$

Predictions for the HetGP(\cdot) component are the standard heteroscedastic predictions (Binois, Gramacy, and Ludkovski 2018) conditioned on values for $\mathbf{y} - \text{DetGP}(X) | \mathbf{y}_{\text{det}}$ rather than just \mathbf{y} . This leads to predictions for the heteroscedastic Gaussian process HetGP(\cdot) having the form in Equation (5)

$$\text{HetGP}(X^*) | \mathbf{y}, \delta^2(X), \mathbf{y}_{\text{det}} \sim N(\mathcal{M}_{\text{het}}(X^*), \mathcal{V}_{\text{het}}(X^*)), \quad (5)$$

where $\mathcal{M}_{\text{het}}(X^*) = m_{\text{het}}(X^*)$

$$+ K_{\text{het}}(X^*, X)((K_{\text{het}}(X, X) + \delta^2(X)I)^{-1}(\mathbf{y} - \text{DetGP}(X) | \mathbf{y}_{\text{det}} - m_{\text{het}}(X)))$$

and $\mathcal{V}_{\text{het}}(X^*) = K_{\text{het}}(X^*, X^*) + \delta^2(X^*)I$

$$- K_{\text{het}}(X^*, X)((K_{\text{het}}(X, X) + \delta^2(X)I)^{-1}K_{\text{het}}(X, X^*).$$

Predictions for the stochastic simulator output then become

$$\eta(X^*) | \mathbf{y}, \delta^2(X), \mathbf{y}_{\text{det}} = \text{DetGP}(X^*) | \mathbf{y}_{\text{det}} + \text{HetGP}(X^*) | \mathbf{y}, \delta^2(X), \mathbf{y}_{\text{det}}. \quad (6)$$

Applying this model to the toy simulator from before, using the previously obtained stochastic and deterministic data points, we obtain the predictions in Figure 3.

The overall shape of this emulator is closer to the truth, with the periodic feature now represented in the emulator.

The deterministic approximation yields outputs that are overall too large compared to the stochastic simulator's mean, and thus the same is true for the DetGP Gaussian process, the mean predictions of which are represented by the higher, orange, line. This DetGP Gaussian process does however contain useful information about the shape of the stochastic simulator's mean,

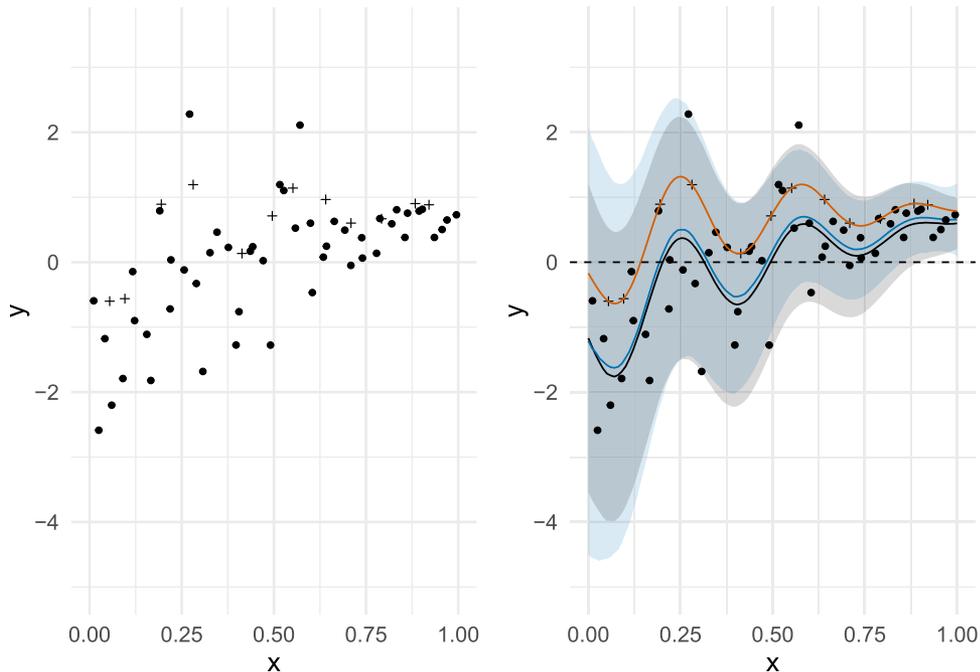


Figure 3. Emulator predictions for the toy simulator, using both stochastic and deterministic runs. The black interval and respective line are the true 95% interval and mean, and the blue interval and line are the emulator 95% predictive interval and mean. The stochastic data points are circles, and the deterministic data points are plus symbols. The mean of the DetGP component is in orange and interpolates the deterministic data points.

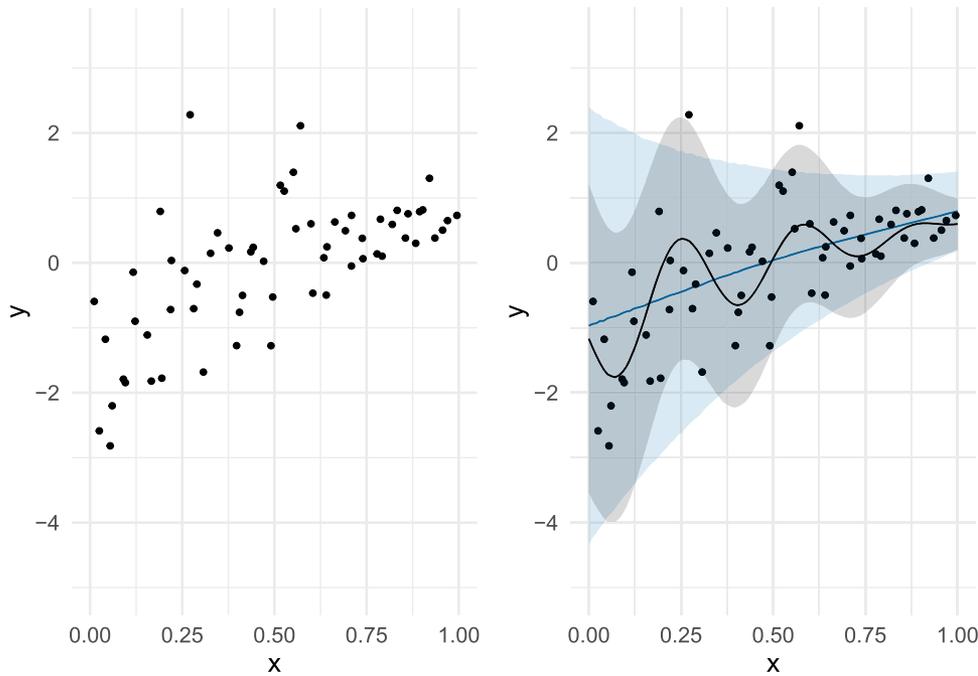


Figure 4. Sixty-two evaluations of the toy simulator from Equation (1) (left), and the respective heteroscedastic emulator predictions (right). The more periodic true mean and 95% interval are superimposed in black, and the more linear emulator mean and 95% predictive interval are in blue.

and thus it can be adjusted by the HetGP Gaussian process to ensure the emulator’s mean and the stochastic simulator’s mean match. The complexity of this adjustment Gaussian process HetGP is related to how good an approximation the deterministic simulator is. In this toy example, the deterministic approximation differs from the true mean of the stochastic simulator linearly, and thus the HetGP Gaussian process needs to be approximately linear for the full emulator to have good fit.

This “DetHetGP” emulator is evidently a better surrogate for the simulator than the previous heteroscedastic emulator, but it has used an additional 12 simulator runs to be so. To show that using deterministic runs was an efficient use of a simulator budget, Figure 4 shows the base heteroscedastic emulator fitted to the original 50 stochastic data points, plus an additional 12 stochastic data points generated from the same input values as the deterministic runs.

Here, the emulator remains substantially inferior to the emulator that uses some of the simulator budget to incorporate deterministic runs, failing to capture the periodic component, despite using the same total number of simulator runs. This example suggests that deterministic runs can indeed be a useful tool for modeling a stochastic simulator.

3. Guidelines

This section provides three criteria that must be satisfied for the outlined method to perform well, and advice on how to ensure they are satisfied. Also shown will be examples of what outcomes can occur if a criterion is not met. This section is not intended to downplay the outlined method; instead it serves to provide an intuitive understanding and actionable advice such that the method will work in practice.

The three criteria that we identify, in descending order of perceived importance are

1. The deterministic approximation must be informative in some way
2. There must be a sufficient number of deterministic simulation runs
3. There must be a sufficient number of stochastic simulation runs

3.1. Criterion 1

The first criterion is fairly straight forward: an assumption is made that the deterministic approximation provides some value with regard to the shape of the stochastic simulator. This criterion demands some degree of expert prior knowledge. One should hope that if the deterministic approximation and the stochastic simulator are both models of the same process, that common attributes should be present in both. It would be concerning if one model of a process, and a different model of the same process, would bear no similarities.

There is an additional way of obtaining a deterministic approximation that has not been mentioned up until now, which is most likely to fail this criterion: one can create a deterministic simulator by fixing the random seed. In many contexts, this does not create a deterministic approximation, but instead simply a deterministic code with no relation to the overall trends of the real world process.

To show what can occur should this criterion fail, we return to the toy simulator in Section 2, but this time use the following deterministic approximation:

$$\eta(x) = \log(0.1 + 4x). \tag{7}$$

This bears no resemblance to the stochastic simulator, and is ultimately just a different deterministic simulator, and not an approximation. Fitting the statistical model, using 50 stochastic

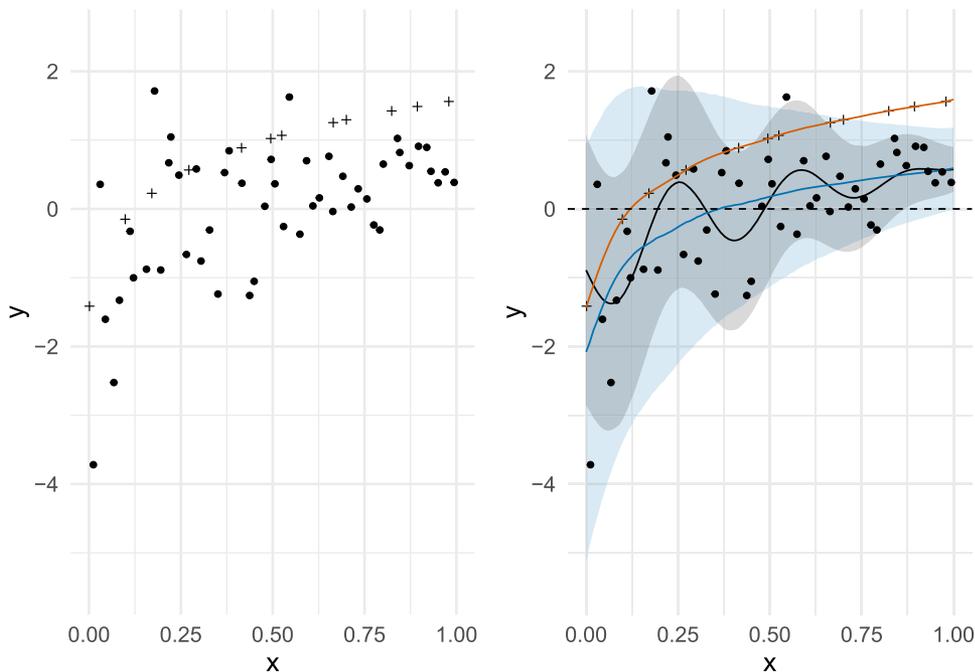


Figure 5. Emulator predictions for the toy simulator, using both stochastic and deterministic runs. The deterministic simulator here bears no resemblance to the stochastic simulator. The black interval and respective line are the true 95% interval and mean, and the blue interval and line are the emulator 95% predictive interval and mean. The stochastic data points are circles, and the deterministic data points are plus symbols. The mean of the DetGP component is in orange and interpolates the deterministic data points.

points and 12 deterministic points, provides the predictions in Figure 5.

One can see from this plot, that the deterministic points appear in an “r” shaped curve, and so the DetGP component tracks this. The resulting emulator uses this as a baseline for what the overall stochastic trend is; the outcome being that the emulator incorrectly believes the mean of the stochastic simulator is also an “r” shaped curve. It is in this way, that a deterministic code which does not approximate the stochastic simulator can misinform the emulator into believing something which is untrue. This does not appear to be too problematic in Figure 5, but were the standard heteroscedastic Gaussian process emulator able to capture the correct trend, a poor deterministic “approximation” might instead bias it into no longer learning said trend.

Our advice is to only use a deterministic approximation if there is a real reason to expect some similarity between the deterministic approximation and the stochastic simulator, such as with the four example situations suggested in Section 2.

3.2. Criterion 2

Failing the second criterion yields similar effects as failing the first criterion. Should there be too few deterministic simulations, DetGP can fail to capture the information contained within the deterministic approximation, and instead learn a different relationship. This different relationship may not be particularly informative to the shape of the stochastic simulator, but is nonetheless used to form predictions of the stochastic simulator.

Fitting the statistical model, using the exact same toy stochastic simulator and deterministic approximation from Section 2,

but now with only 4 deterministic points (and 58 stochastic points), provides the predictions in Figure 6.

In this plot, DetGP obtains a relationship which is not particularly informative to the overall mean of the stochastic simulator. The relationship that DetGP obtains still acts as a baseline for the overall mean, and there is not enough stochastic data to unlearn this relationship. This is effectively the same problem as when Criterion 1 fails; there is little difference between having a bad deterministic “approximation,” and having a good deterministic approximation but obtaining a bad representation of it.

Our advice here is based on the rule of thumb from Loepky, Sacks, and Welch (2009). Because satisfying Criterion 2 is essential, we recommend a more conservative 20 simulation runs per dimension, if possible, to capture the relationship of the deterministic approximation. We recommend this more conservative rule as the potential missed opportunity from failing to capture the deterministic approximation is large. This rule is followed in Section 4. A more thorough practice, which we recommend more strongly, is to fit a separate standard deterministic emulator to the deterministic simulator; and ensure enough deterministic data is obtained such that the deterministic emulator passes certain validation checks [such as those outlined in Bastos and O’Hagan (2009), or simpler leave-one-out cross-validation checks (see, e.g., Williamson, Blaker, and Sinha 2017)]; after this is done, one can feel confident that enough deterministic data has been obtained.

3.3. Criterion 3

The final criterion deals with the opposite problem to Criterion 2. Stochastic runs are essential to modify the deterministic approximation such that it agrees with the stochastic simulator.

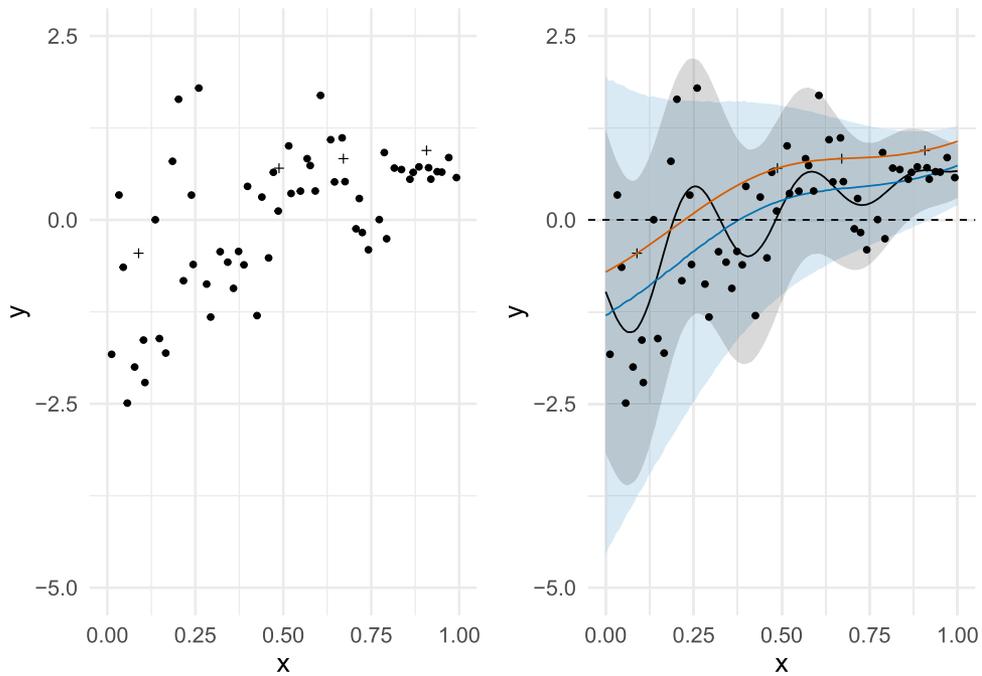


Figure 6. Emulator predictions for the toy simulator, using both stochastic and deterministic runs. Only four deterministic runs were used. The black interval and respective line are the true 95% interval and mean, and the blue interval and line are the emulator 95% predictive interval and mean. The stochastic data points are circles, and the deterministic data points are plus symbols. The mean of the DetGP component is in orange and interpolates the deterministic data points.

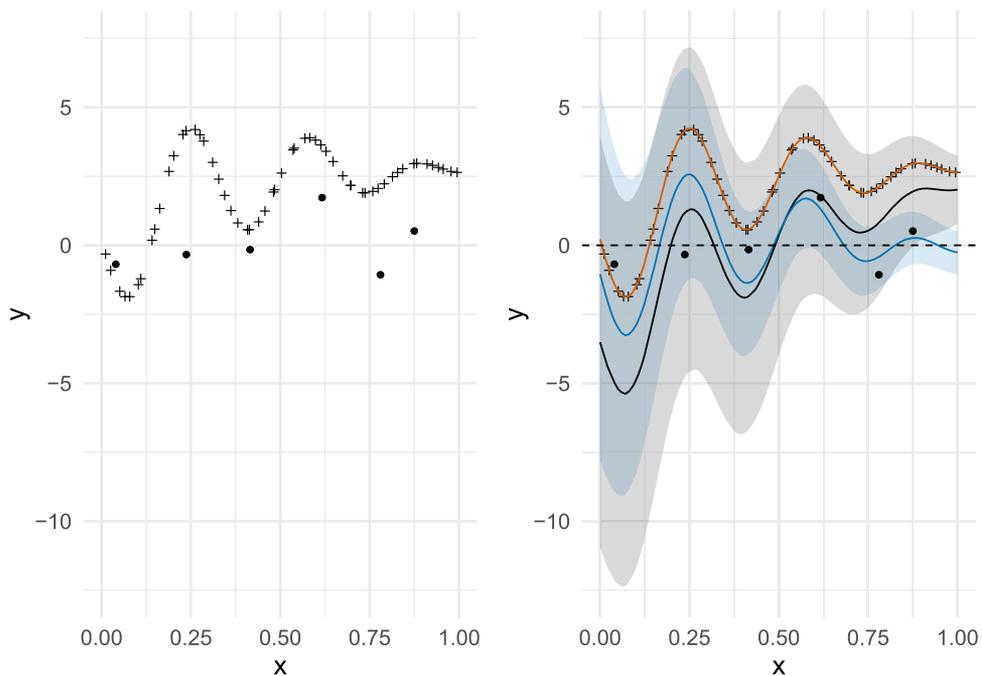


Figure 7. Emulator predictions for the toy simulator, using both stochastic and deterministic runs. Only six stochastic runs were used. The black interval and respective line are the true 95% interval and mean, and the blue interval and line are the emulator 95% predictive interval and mean. The stochastic data points are circles, and the deterministic data points are plus symbols. The mean of the DetGP component is in orange and interpolates the deterministic data points.

Stochastic runs are also needed to estimate the variance of the stochastic simulator.

Figure 7 presents the predictions obtained from fitting the emulator to the toy simulator and deterministic approximation when only 6 runs were stochastic (and 56 were deterministic).

This plot shows the emulator failing to adjust the location and shape of the deterministic approximation correctly—it has been decreased too little for small values of x and too much for

large values of x . Additionally, the variance is estimated poorly—especially for large values of x where it is estimated far too small. This serves as an extreme example, where such a huge percentage of the simulation budget is assigned as deterministic points, but it clearly establishes that having a sufficiently large number of stochastic runs is still important.

Our advice here is that, after the required number of deterministic runs for Criterion 2 to be satisfied is obtained, the

remainder of the simulation budget should be assigned as stochastic points, and no more assigned as deterministic points.

Since Criteria 2 and 3 can act against each other, the advice given should ensure the developed model performs acceptably well. In some circumstances, especially when the total simulation budget is larger, and thus the standard heteroscedastic Gaussian process model will perform less abysmally, a trade off can emerge; where using deterministic points leads to a better mean function estimate, but instead simply using more stochastic points leads to a better variance function estimate.

Decisions based on such a trade-off depend on subjective preferences, and hence the developed methodology lends itself more to situations where the mean is of greater interest. On the other hand, it is incredibly difficult to know when the simulation budget is large enough for the trade-off to arise; it depends on the complexity of the unknown true mean function and the unknown true variance function. Given that it is difficult to know a-priori when a simulation budget is large enough for the trade-off to arise, it remains a safer choice to use a deterministic approximation, regardless of whether the mean or the variance is of greater interest.

Additionally, when the simulation budget is very large, or the simulator is overly simplistic; it is also possible for the information provided by the deterministic approximation to be entirely learnt using only stochastic points; and so running any deterministic points can be wasteful. This makes deterministic approximations most useful when the simulation budget is limited

4. Examples

In this section, the method will be applied to two examples that are more realistic.

To compare the performance of the standard heteroscedastic Gaussian process (HetGP) and the developed method (DetHetGP), we will use two metrics: the mean squared error (MSE) and a “score” that scores predictions according to both their mean and variance. Both of these metrics use out-of-sample simulations runs to assess the predictions of a statistical model. The mean squared error is the mean squared difference between the observed out-of-sample simulator runs and the predictive mean of the emulator; and the score is from Gneiting and Raftery (2007, eq. 27), which is also used in Binois, Gramacy, and Ludkovski (2018).

Smaller values for the MSE indicate an improved mean function, and larger values for the score indicate better overall predictions.

All simulation datasets will be obtained with input points chosen by a maximin Latin hypercube design.

4.1. Susceptible-Infected-Recovered Simulator

The first simulator we investigate is a basic susceptible-infected-recovered (SIR) model using the individual contact model (ICM) from the EpiModel package (Jenness, Goodreau, and Morris 2018). This stochastically models a population, where individuals can be: susceptible to some disease; currently infected with the disease; or recovered from (and now immune to) the disease. This is a fast simulator, but it serves as a more

authentic example than the toy simulator in the previous sections. The two parameters we shall vary and use as our inputs for the emulators are: the probability of infection which will be allowed to vary between 0.5 and 1, and the recovery rate which will vary between 0 and 0.01. Other parameters exist in the model, such as the rate of interactions between individuals, the initial number of infected, the total population, and the time step to count the number of infected (all taken as fixed, respectively, as 0.01, 5, 1000, and 300). The package also includes a deterministic compartmental model (DCM), which is a different simulator, but takes the same inputs and provides the same outputs as the stochastic simulator, and thus shall be our deterministic approximation. Note that this is the case where a deterministic approximation is available because an alternative deterministic model of the same real world process exists.

HetGP is fit using 120 stochastic simulations, and DetHetGP is fit using 80 stochastic simulations and 40 deterministic simulations (i.e., 20 deterministic points per input dimension, with the rest of the budget assigned as stochastic runs). To compare the two emulators, we obtain 200 more simulation runs, and then calculate the MSE and score. Because this simulator is relatively cheap, we can repeat this entire procedure 100 times; smoothing out natural variation from specific Latin hypercube realizations and specific simulation runs. We can then present summary statistics of the MSE and score.

Table 1 gives these values.

Here, we can see that the score is marginally improved for DetHetGP, and the MSE is substantially improved, with the upper quartile for DetHetGP smaller than the median for HetGP. Not only does DetHetGP appear to be the preferred emulator here, but the deterministic approximation is much faster than the stochastic version. The stochastic simulator takes approximately 0.18 sec to run, whereas the deterministic approximation takes approximately 0.05 sec to run. It is not uncommon for a deterministic approximation to be cheaper, such simulators are likely to be simpler, and thus faster. This would imply that more simulation runs could be afforded if more deterministic runs were chosen; in this case one stochastic run costs more than three deterministic runs. To provide more conservative results, we ignore this advantage in these comparisons, despite the potential improvement it provides DetHetGP.

Because the score is only marginally improved (and potentially within the margin of natural variability from only repeating the comparison 100 times), but the mean squared error is substantially improved, this example is potentially a case where the trade-off between improved mean and improved variance occurs.

To visually show the improvements DetHetGP yields, Figure 8 shows predictions from both emulators with the infection

Table 1. The summary statistics of the MSE and score for both HetGP and DetHetGP from the simulation experiment conducted for the SIR simulator.

		Lower quartile	Median	Upper quartile
MSE	HetGP	0.326	0.418	0.495
	DetHetGP	0.292	0.341	0.399
Score	HetGP	116.4	145.1	172.4
	DetHetGP	117.0	146.9	180.0

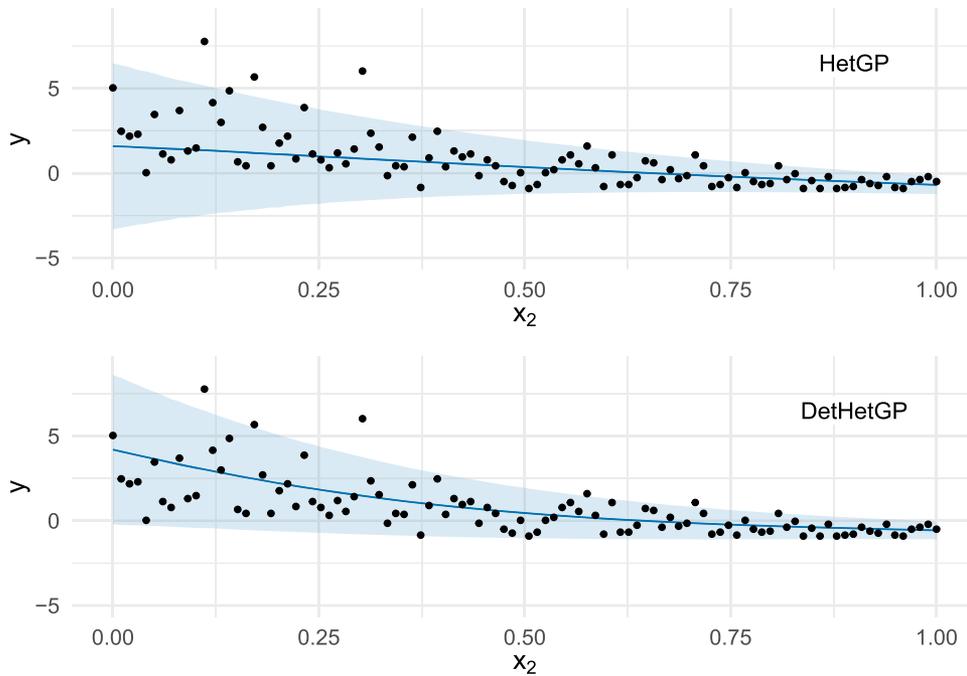


Figure 8. Emulator predictions for both HetGP (top) and DetHetGP (bottom). The predictions are for simulator runs where the infection rate is constant at 1, and only the recovery rate varies. Also superimposed are 100 additional simulator runs where the infection rate is kept constant at 1.

rate x_1 kept constant at 1, and only the recovery rate is varied. Superimposed on this plot are 100 out of sample simulator runs where the infection rate was also kept fixed at 1.

The mean predictions from HetGP are approximately linear, missing the sharper increase for lower values of the recovery rate, and instead a larger variance is predicted for these values—large observed simulator runs were probably interpreted by the emulator as existing because of a larger variance rather than because of a larger mean. DetHetGP on the other hand does estimate the sharp increase in the mean function, and does not feature an overly large variance estimate for low values of the recovery rate.

Other cross-sections of the emulator’s prediction surface could have been presented, and a similar pattern exists for when the recovery rate is fixed at 0 and the infection rate is allowed to vary. The given plots clearly show that DetHetGP can provide a significant advantage over HetGP.

4.2. Building Performance Simulator

The second, real-world, example we investigate is a simulator for modeling the energy usage of a building (Crawley et al. 2001). Various attributes of a building must be input, such as the buildings shape, properties of the building, and the weather. The specific modeled building in question is a hospital, taken from a reference hospital file (Deru et al. 2011), and the input variables considered are: wall concrete thickness, wall insulation thickness, roof insulation thickness, floor concrete thickness, and window size (as a percentage of the total wall height).

Often, a single year of weather data is input, and the simulator outputs the energy usage of the building if that year of weather were observed; efforts are then spent on creating typical weather files to use (Eames, Ramallo-Gonzalez, and Wood 2016). This is an odd way of treating weather, as the weather often does not

behave “typically.” We modify this process by sampling a new year of weather every time the simulator is run, converting the deterministic simulator into a stochastic simulator. Every time the building simulator is run, a new weather file is randomly generated by combining sampled weeks from historical records, stratified according to season. A deterministic approximation of this stochastic simulator is then easily available as the same simulator but with the weather fixed again. Note that this is simultaneously the case where a deterministic approximation is available because an old version of the simulator was deterministic, and because one of the inputs is actually random.

Two hundred data points are used to fit the two emulators, with each run taking roughly 65 sec. For DetHetGP 100 of the data points will be deterministic runs (20 per dimension), and 100 will be stochastic. For HetGP the same stochastic runs will be used, as well as an additional 100 stochastic points with input values the same as those for the deterministic points. The datasets will be standardized according to the sample mean and standard deviation of the shared 100 stochastic data points.

This simulator is more expensive than the SIR simulator, and so only one set of 500 out-of-sample simulation runs will be generated. DetHetGP receives a score of -189.0 and an MSE of 0.527 , whereas HetGP receives a score of -287.5 and an MSE of 0.809 . DetHetGP seems to perform better than HetGP on the building model in this case.

To further investigate this difference, Figure 9 shows the emulator predictive distribution for wall insulation thickness, keeping all other inputs fixed (at 0.5). 100 additional out-of-sample simulator runs are also shown, where all other input points are also fixed at 0.5.

The predictions from DetHetGP are characterized by a decrease in energy usage for very low values of wall insulation, after which improvements in energy efficiency seem to stabilize. This is consistent with our experience of the building model.

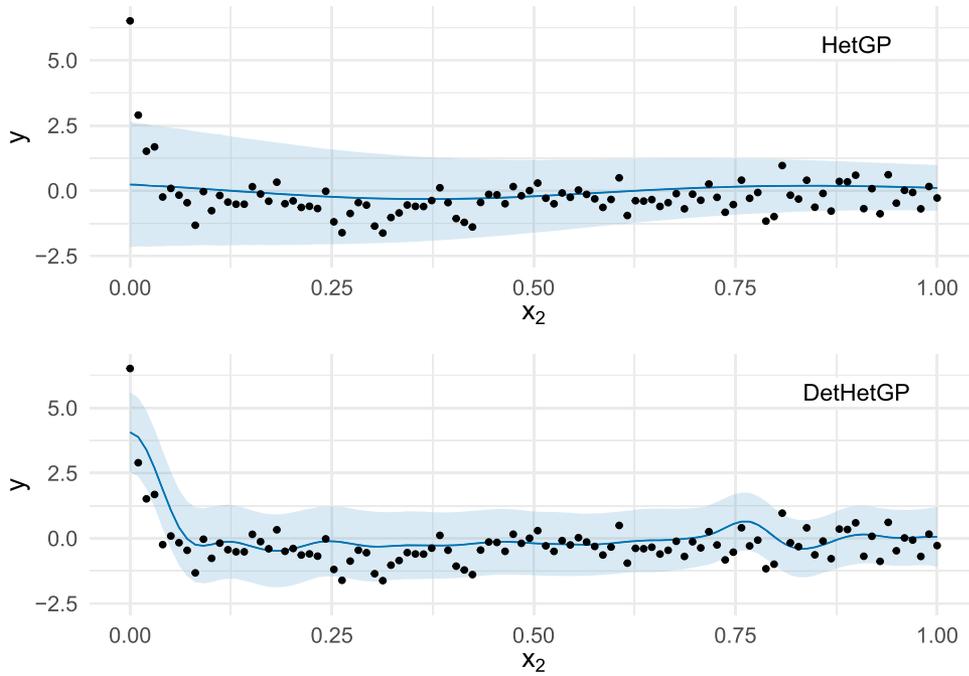


Figure 9. Emulator predictions for second input of the building model (wall insulation thickness), when all other inputs are kept fixed at 0.5. HetGP predictions are plotted above, and DetHetGP predictions are below. Also superimposed are 100 additional simulator runs where the remaining 4 inputs were kept constant at 0.5.

HetGP does not yield this characteristic, instead it is more defined by a decrease in the variability of the energy usage as wall insulation thickness increases. If we were to assume that the trend discovered by DetHetGP is more accurate, it is likely that extremely large observed values of energy usage were instead assumed by HetGP to be the result of an increased variance, rather than a sharp increase in the mean. The superimposed out-of-sample data points agree more with the predictions from DetHetGP, suggesting this emulator is better. The data implies the sharp increase for low values of wall insulation thickness, predicted by DetHetGP but not by HetGP, is correct and possibly even sharper than predicted by DetHetGP.

DetHetGP does not appear perfect from these plots, however, as around $x = 0.75$ there is a small, but sharp, decrease in the mean. This decrease does not seem sensible from the data, nor from prior understanding. Perhaps the deterministic approximation was modeled poorly due to a lack of deterministic data, or perhaps a limited amount of stochastic data caused this error. Using the more thorough advice from [Section 3](#), wherein a deterministic emulator is fit and validated beforehand, would have been more informative, and perhaps would have avoided this issue.

Although not definitive, the evidence suggests that DetHetGP performs better than HetGP. Including deterministic runs leads to a more accurately shaped mean function, and because the two emulators disagree with the mean function shape, it is probable that HetGP has estimated the mean function poorly.

5. Conclusions

We have presented a method for including deterministic runs in the emulation of stochastic simulation experiments. By using a deterministic approximation of the stochastic simulator, a

less noisy view into the general shape of the mean function can be learnt. Including deterministic runs can produce better emulators, especially when the mean function is complex and sufficient prior knowledge of the mean function is lacking, or the simulation budget is insufficiently large.

HetGP is not the only method for modeling a stochastic simulator with a Gaussian process. The variance could instead be modeled with a simple parametric form, as by Boukouvalas et al. (2014). Similarly, homoscedasticity may be an assumption a practitioner is willing to make, and thus a fixed variance, as in basic Gaussian process regression (Rasmussen and Williams 2006), may be viable. It would be interesting to see if DetHetGP remains useful in these cases, and whether it performs better in such scenarios. Similarly, other methods apart from Gaussian processes exist for flexibly modeling heteroscedastic systems, such as the method by Pratola et al. (2019). It would also be interesting to see if deterministic approximations can be incorporated into other such methods.

An interesting design question has been posed regarding how many deterministic points should be included: too few and the deterministic approximation will not be modeled well; too many and not enough stochastic data points will be generated, yielding a poor estimate of the variance and potentially the overall mean. The advice given in [Section 3](#) provides a good starting point for this.

Another modification would be the inclusion of replicated simulator runs in the fitting of the emulator. Replicates are often used in the fitting of heteroscedastic emulators (Ankenman, Nelson, and Staum 2010; Boukouvalas, Cornford, and Stehlik 2014; Binois et al. 2019), with the goal of obtaining a better understanding of the simulator’s mean and variance for the observed input points. With DetHetGP substantially improving the mean of stochastic emulator predictions, at the expense of fewer stochastic simulator runs, and sometimes at the expense

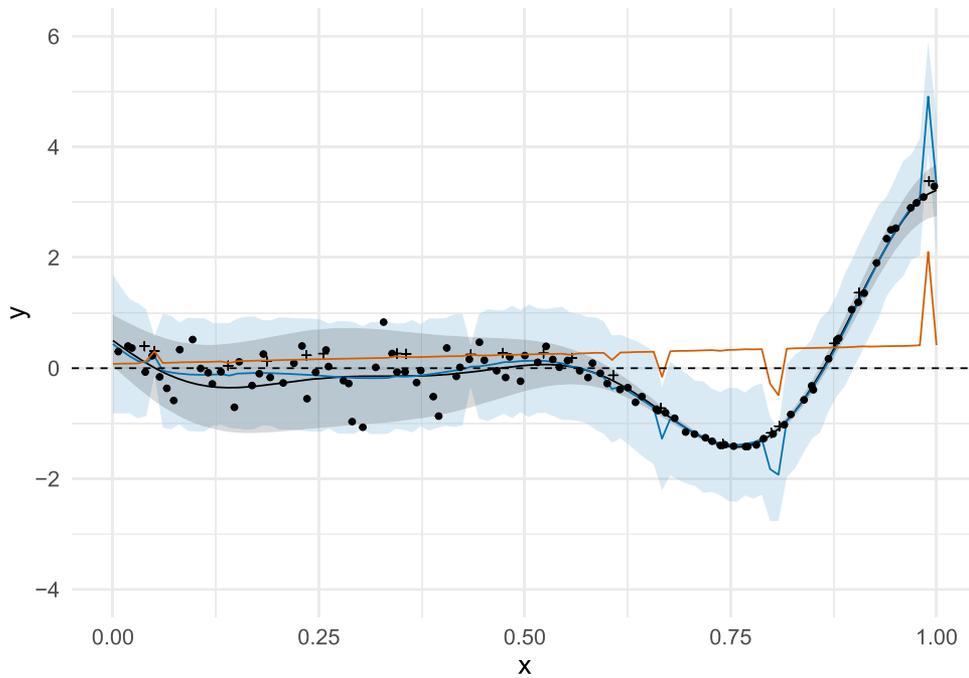


Figure A.1. Emulator predictions for the toy simulator from Binois et al. (2019), using the newly developed model that incorporates both stochastic and deterministic runs. The true mean and 95% interval are superimposed in black, and the emulator mean and 95% interval are in blue. The stochastic data points are circles, and the deterministic data points are plus symbols. The mean of the DetGP component is in orange.

of a worse variance prediction, it would be interesting to see whether combining replicates with incorporating deterministic approximations would yield an improved emulator overall.

Perhaps related, is the computational benefits that could be achieved by restricting stochastic runs to only be at locations where deterministic runs have been made (see Kennedy and O’Hagan 2000, for a similar implementation restriction). Such a restriction becomes more feasible when replicates are used, allowing the restriction to be obeyed, while also not forcing the number of stochastic runs to be less than the number of deterministic runs.

To conclude, we find that runs from a deterministic approximation can be used to help predict the output from a stochastic simulator. This can be done by modeling the stochastic simulator as a sum of two Gaussian processes, one of which is conditioned on deterministic approximation runs, and the other on runs from the stochastic simulator. The model proves powerful, yielding more accurate predictions of the output for multiple toy and real simulators explored in this article.

Appendix A. DetHetGP Prior

All the emulators fit in this article have had an additional “nugget variance” added for computational reasons (Neal 1997), including the deterministic Gaussian processes. A value of $1e-4$ was required for this nugget variance to prevent computational issues in inverting the covariance matrices.

For the stochastic emulator that incorporates deterministic runs, the l_{det_i} prior has been modified slightly. In practice $l_{\text{det}_i} = 0.05 + l_{\text{det}_i}^*$, and $l_{\text{det}_i}^*$ has a Gamma(4, 4) prior. This constrains l_{det_i} to be greater than 0.05. Figure A.1 gives an example of what can happen in practice if l_{det_i} is not explicitly constrained to be larger than zero.

l_{det_i} has an estimated value of 0.00150 here, which is very small. This results in the deterministic Gaussian process (plotted in orange) being

approximately a straight line, with steep jumps toward the observed deterministic points (which is an established problem discussed by Andrianakis and Challenor (2012), caused by the inclusion of a nugget variance). This leads to the final stochastic emulator also having steep unnecessary jumps because there is not enough stochastic data for the HetGP to smooth them out. Additionally, the deterministic emulator being approximately linear also leads to extraneous variance in the deterministic emulator predictions. This additional variance then also mostly accounts for the intrinsic variance of the stochastic simulator, leading to smaller estimates for $\delta^2(X^*)$, and a less flexible estimated variance process.

This issue could be prevented by decreasing the value of the nugget variance for the deterministic Gaussian process, but can lead to computational errors. An alternative solution is the one implemented, fixing l_{det_i} to be sufficiently larger than zero.

Supplementary Materials

R code: R scripts (and a Stan file) which contain functions to fit the developed model and make predictions from the model. Also contained are similar scripts for a deterministic GP and a heteroscedastic GP; which are then used in two R Markdown files to recreate the results in Sections 2 and 3. (.zip file)

Funding

The authors gratefully acknowledge funding provided by the Engineering and Physical Sciences Research Council.

References

- Andrianakis, I., and Challenor, P. G. (2012), “The Effect of the Nugget on Gaussian Process Emulators of Computer Models,” *Computational Statistics & Data Analysis*, 56, 4215–4228. [11]
- Ankenman, B., Nelson, B. L., and Staum, J. (2010), “Stochastic Kriging for Simulation Metamodeling,” *Operations Research*, 58, 371–382. [10]

- Ba, S., and Joseph, V. R. (2012), “Composite Gaussian Process Models for Emulating Expensive Functions,” *The Annals of Applied Statistics*, 6, 1838–1860. [3]
- Bastos, L. S., and O’Hagan, A. (2009), “Diagnostics for Gaussian Process Emulators,” *Technometrics*, 51, 425–438. [6]
- Binois, M., Gramacy, R. B., and Ludkovski, M. (2018), “Practical Heteroscedastic Gaussian Process Modeling for Large Simulation Experiments,” *Journal of Computational and Graphical Statistics*, 27, 808–821. [1,3,4,8]
- Binois, M., Huang, J., Gramacy, R. B., and Ludkovski, M. (2019), “Replication or Exploration? Sequential Design for Stochastic Simulation Experiments,” *Technometrics*, 61, 7–23. [1,10,11]
- Boukouvalas, A., and Cornford, D. (2009), “Learning Heteroscedastic Gaussian Processes for Complex Datasets,” Technical Report. [1]
- Boukouvalas, A., Cornford, D., and Stehlik, M. (2014), “Optimal Design for Correlated Processes With Input-Dependent Noise,” *Computational Statistics & Data Analysis*, 71, 1088–1102. [10]
- Boukouvalas, A., Sykes, P., Cornford, D., and Maruri-Aguilar, H. (2014), “Bayesian Precalibration of a Large Stochastic Microsimulation Model,” *IEEE Transactions on Intelligent Transportation Systems*, 15, 1337–1347. [10]
- Brynjarsdóttir, J., and O’Hagan, A. (2014), “Learning About Physical Parameters: The Importance of Model Discrepancy,” *Inverse Problems*, 30, 114007. [3]
- Cantoni, E., and Hastie, T. (2002), “Degrees-of-Freedom Tests for Smoothing Splines,” *Biometrika*, 89, 251–263. [2]
- Crawley, D. B., Lawrie, L. K., Winkelmann, F. C., Buhl, W. F., Huang, Y. J., Pedersen, C. O., Strand, R. K., Liesen, R. J., Fisher, D. E., Witte, M. J., and Glazer, J. (2001), “EnergyPlus: Creating a New-Generation Building Energy Simulation Program,” *Energy and Buildings*, 33, 319–331. [9]
- Deru, M., Field, K., Studer, D., Benne, K., Griffith, B., Torcellini, P., Liu, B., Halverson, M., Winiarski, D., Rosenberg, M., and Yazdanian, M. (2011), “US Department of Energy Commercial Reference Building Models of the National Building Stock.” [9]
- Eames, M. E., Ramallo-Gonzalez, A. P., and Wood, M. (2016), “An Update of the UK’s Test Reference Year: The Implications of a Revised Climate on Building Design,” *Building Services Engineering Research and Technology*, 37, 316–333. [9]
- Gneiting, T., and Raftery, A. E. (2007), “Strictly Proper Scoring Rules, Prediction, and Estimation,” *Journal of the American Statistical Association*, 102, 359–378. [8]
- Goldberg, P. W., Williams, C. K., and Bishop, C. M. (1998), “Regression With Input-Dependent Noise: A Gaussian Process Treatment,” in *Advances in Neural Information Processing Systems*, pp. 493–499. [1]
- Haaland, B., and Qian, P. Z. (2011), “Accurate Emulators for Large-Scale Computer Experiments,” *The Annals of Statistics*, 39, 2974–3002. [3]
- Jenness, S. M., Goodreau, S. M., and Morris, M. (2018), “EpiModel: An R Package for Mathematical Modeling of Infectious Disease Over Networks,” *Journal of Statistical Software*, 84, 8. [8]
- Kennedy, M. C., and O’Hagan, A. (2000), “Predicting the Output From a Complex Computer Code When Fast Approximations Are Available,” *Biometrika*, 87, 1–13. [3,11]
- (2001), “Bayesian Calibration of Computer Models,” *Journal of the Royal Statistical Society, Series B*, 63, 425–464. [3]
- Kersting, K., Plagemann, C., Pfaff, P., and Burgard, W. (2007), “Most Likely Heteroscedastic Gaussian Process Regression,” in *Proceedings of the 24th International Conference on Machine Learning*, ACM, pp. 393–400. [3]
- Loeppky, J. L., Sacks, J., and Welch, W. J. (2009), “Choosing the Sample Size of a Computer Experiment: A Practical Guide,” *Technometrics*, 51, 366–376. [1,6]
- McKay, M. D., Beckman, R. J., and Conover, W. J. (2000), “A Comparison of Three Methods for Selecting Values of Input Variables in the Analysis of Output From a Computer Code,” *Technometrics*, 42, 55–61. [2]
- Neal, R. M. (1997), “Monte Carlo Implementation of Gaussian Process Models for Bayesian Regression and Classification,” arXiv no. physics/9701026. [11]
- O’Hagan, A. (2006), “Bayesian Analysis of Computer Code Outputs: A Tutorial,” *Reliability Engineering & System Safety*, 91, 1290–1300. [1]
- Pratola, M., Chipman, H., George, E., and McCulloch, R. (2019), “Heteroscedastic BART via Multiplicative Regression Trees,” *Journal of Computational and Graphical Statistics*, 1–13. doi: 10.1080/10618600.2019.1677243 [10]
- Rasmussen, C. E., and Williams, C. K. I. (2006), *Gaussian Processes for Machine Learning*, Cambridge, MA: MIT Press. [1,3,4,10]
- Stan Development Team (2016), “Stan Modeling Language Users Guide and Reference Manual,” *Technical Report*. [3]
- Vernon, I., Goldstein, M., and Bower, R. G. (2010), “Galaxy Formation: A Bayesian Uncertainty Analysis,” *Bayesian Analysis*, 5, 619–669. [3]
- Williamson, D. B., Blaker, A. T., and Sinha, B. (2017), “Tuning Without Over-Tuning: Parametric Uncertainty Quantification for the NEMO Ocean Model,” *Geoscientific Model Development*, 10, 1789–1816. [6]