RobustCalibration: Robust Calibration of Computer Models in R

by Mengyang Gu

Abstract Two fundamental research tasks in science and engineering are forward predictions and data inversion. This article introduces a new R package RobustCalibration for Bayesian data inversion and model calibration using experiments and field observations. Mathematical models for forward predictions are often written in computer code, and they can be computationally expensive to run. To overcome the computational bottleneck from the simulator, we implemented a statistical emulator from the RobustGaSP package for emulating both scalar-valued or vector-valued computer model outputs. Both posterior sampling and maximum likelihood approach are implemented in the RobustCalibration package for parameter estimation. For imperfect computer models, we implement the Gaussian stochastic process and scaled Gaussian stochastic process for modeling the discrepancy function between the reality and mathematical model. This package is applicable to various other types of field observations and models, such as repeated experiments, multiple sources of measurements and correlated measurement bias. We discuss numerical examples of calibrating mathematical models that have closed-form expressions, and differential equations solved by numerical methods.

1 Introduction

Complex processes are often represented as mathematical models, implemented in computer code. These mathematical models are often called *computer models* or *simulators*, and are widely used to simulate different processes given a set of parameters and initial conditions (Sacks et al., 1989). The initial conditions and model parameters may be unknown or uncertain in practice. Calibrating the computer models based on real experiments or observations is one of the fundamental tasks in science and engineering, widely known as *data inversion* or *model calibration*.

We follow the notation in Bayarri et al. (2007a) for defining the model calibration problem. Let $y^F(\mathbf{x})$ be the real-valued field observation with a p_x-dimensional observable input vector $\mathbf{x}$. The field observation is often decomposed by $y^F(\mathbf{x}) = y^R(\mathbf{x}) + \epsilon$, where $y^R(\cdot)$ denotes a function of unknown reality, and ϵ denotes the noise, with the superscript 'F' and 'R' denoting the field observations and reality, respectively. Scientists often model the unknown reality by a computer model, denoted by $f^M(\mathbf{x}, \boldsymbol{\theta})$ at the p_x-dimensional observable input $\mathbf{x}$, and p_θ-dimensional calibration parameters $\boldsymbol{\theta}$, unobservable from the experiments, with superscript 'M' denoting the computer model.

When the computer model describes the reality perfectly, a field observation can be written as:

$$y^F(\mathbf{x}) = f^M(\mathbf{x}, \boldsymbol{\theta}) + \epsilon, \tag{1}$$

where ϵ is a zero-mean Gaussian noise. We call the method by Equation (1) the *no-discrepancy calibration*.

When computer models are imperfect, the statistical calibration model below is often used:

$$y^F(\mathbf{x}) = f^M(\mathbf{x}, \boldsymbol{\theta}) + \delta(\mathbf{x}) + \epsilon, \tag{2}$$

where $\delta(\cdot)$ is the unobservable discrepancy between the mathematical model and reality. Equation (2) implies that the reality can be written as $y^R(\mathbf{x}) = f^M(\mathbf{x}, \boldsymbol{\theta}) + \delta(\mathbf{x})$ at any observable input $\mathbf{x} \in \mathcal{X}$. The model calibration framework has been studied extensively. In Kennedy and O'Hagan (2001), the discrepancy function $\delta(\cdot)$ is modeled via a Gaussian stochastic process (GaSP), leading to more accurate prediction based on the joint model of calibrated computer model and discrepancy compared to either using the computer model or discrepancy model for prediction alone. We call this method the *GaSP Calibration*. The GaSP calibration has been applied in a wide range of studies of continuous and categorical outputs (Bayarri et al., 2007a; Higdon et al., 2008; Paulo et al., 2012; Chang et al., 2016, 2022). Both no-discrepancy calibration and GaSP calibration are implemented in the **RobustCalibration** package.

The calibrated computer model output can be far from the observations in terms of L_2 distance when the discrepancy function is modeled by the GaSP, (Arendt et al., 2012), since a large proportion of the variability in the data can be explained by discrepancy function. To solve this problem, we implemented the scaled Gaussian stochastic process (S-GaSP) calibration, or *S-GaSP Calibration* in the **RobustCalibration** package. The S-GaSP model of discrepancy function was first introduced in Gu and Wang (2018), where more prior probability mass of the L_2 loss is placed on small values. Consequently, the calibrated computer model by the S-GaSP fits the reality better in terms of L_2 loss than the GaSP calibration. Furthermore, historical data may be used for reducing the parameter space

(Williamson et al., 2013).

A few recent approaches aim to minimize the L_2 distance between the reality and computer model (Tuo and Wu, 2015; Wong et al., 2017), where the reality and discrepancy function are often estimated in two steps separately. The S-GaSP calibration bridges the two-step L_2 calibration with the GaSP calibration. Compared with two-step approaches, parameters and discrepancy $(\boldsymbol{\theta}, \delta(.))$ are estimated in GaSP calibration and S-GaSP calibration jointly, and the uncertainty of these estimation can be obtained based on the likelihood function of the sampling model. Furthermore, compared with the orthogonal calibration model (Plumlee, 2017), the S-GaSP process places more prior mass on inputs leading to small values of the random L_2 loss, instead of more prior mass on inputs close to the critical points of the loss function.

Another concern is the computational cost of model calibration, as computer models may involve numerical solutions of differential equations, which can be expensive to run. We utilize a statistical emulator for approximating computationally expensive computer models. The GaSP emulator is a well-developed framework for emulating computationally expensive computer models and it was implemented in a few R packages, such as **DiceKriging** (Roustant et al., 2012), **GPfit** (MacDonald et al., 2015) and **RobustGaSP** (Gu et al., 2019). Here we specify the functions `rgasp`, `ppgasp` and `predict` in **RobustGaSP** for emulating computer models with scalar-valued and vector-valued outputs, respectively.

Building packages for Bayesian model calibration is much more complicated than packages of statistical emulator, as both field observations and computer models need to be integrated. A few statistical packages are available for Bayesian model calibration, such as **BACCO** (Hankin, 2005), **SAVE** package (Palomo et al., 2015), and **CaliCo** (Carmassi et al., 2018). The **CaliCo** package, for example, integrates **DiceKriging** for emulating computational expensive simulations. It offers different types of diagnostic plots based on **ggplot2** (Wickham, 2011), and distinct prior choices of the parameters. In the **BACCO** and **SAVE**, the GaSP model of the discrepancy function (Kennedy and O'Hagan, 2001) is assumed, and GaSP emulators can be used to approximate expensive computer models.

Although a large number of studies were developed for Bayesian model calibration, to the authors' best knowledge, many methods implemented in **RobustCalibration** have not been implemented in another software package previously. We highlight a few unique features of the **RobustCalibration** package. First of all, we allow users to specify different types of output of field observations, for different scenarios through the `output` argument in the `rcalibration` function. The simplest scenario is to have a vector of fields observations. We also allow users to input a matrix or a list of fields observations with the same or different number of replications, respectively. Efficient computation from sufficient statistics was implemented for model calibration with replications, which can improve computation up to k^3 times, where k is the number of repeated experiments. Second, distinct models can be calibrated with different sets of parameters for data from multiple sources by the `calibration_MS` function. Third, we implemented emulators for approximating expensive computer models with both scalar-valued and vector-valued outputs. Emulators for vector-valued outputs, capable of handling a large number of temporal or spatial coordinates, were not implemented in other packages for model calibration. Here we use the parallel partial Gaussian stochastic process (PP-GaSP) emulator from **RobustGaSP** package (Gu et al., 2019) as a computationally scalable emulator of vector-valued output. The **RobustGaSP** package has been applied to various applications, such as emulating geophysical models of ground deformation (Anderson et al., 2019), and storm surge simulation (Ma et al., 2022). Furthermore, users can choose no-discrepancy calibration, GaSP or S-GaSP models of discrepancy functions for different scenarios. Statistical inferences by both posterior samples and the maximum likelihood estimator (MLE) are made available for these calibration approaches by the `method` argument in the `rcalibration` function.

The rest of the paper is organized below. We first give an overview of the **RobustCalibration** package to introduce the main functions and their usage. We further introduce methodology and numerical examples to illustrate the implemented methods in the **RobustCalibration** package. Closed form expressions of likelihood functions, derivatives, posterior distributions and computational algorithms are outlined in the Appendix.

2 An overview of RobustCalibration

Figure 1 gives a schematic overview of the **RobustCalibration** package. We consider predicting the reality in two ways: using both calibrated computer model and discrepancy (predictive accuracy), and the calibrated computer model alone (calibration accuracy), both evaluated in terms of the L_2 loss. The left panel gives comparison of accuracy and computational cost between the implemented methods. The no-discrepancy calibration is faster than GaSP and S-GaSP calibration, as one does not need to compute the inversion and log determinant of the covariance matrix of the field data,

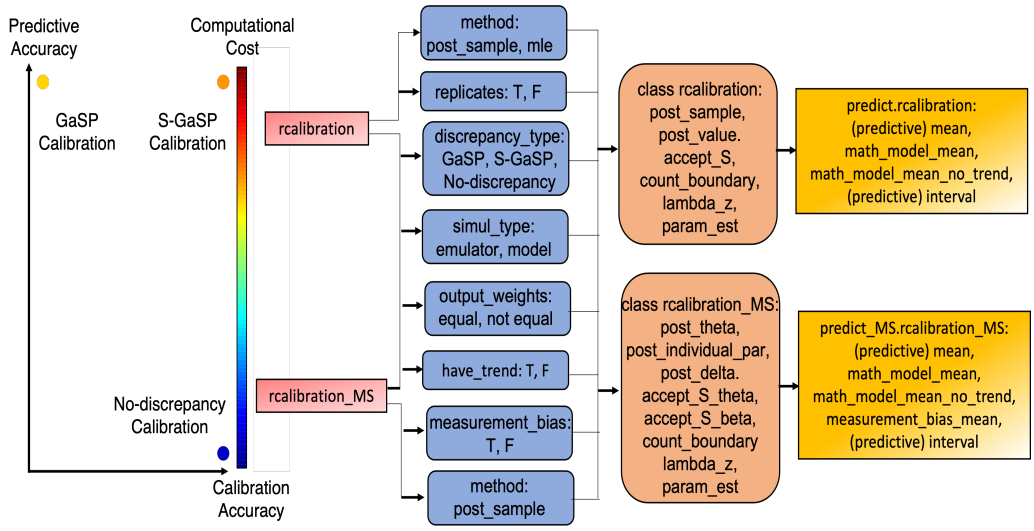

Figure 1: Schematic overview of the **RobustCalibration** package. The left panel compares the expected predictive accuracy and calibration accuracy by different calibration methods. The `calibration` and `calibration_MS` in the right panel are two main functions for parameter estimation for observations from single and multiple sources, respectively. The arguments of these functions are given in blue boxes. The object classes `calibration` and `calibration_MS` (orange boxes) can be supplied in `predict.rcalibration` and `predict_MS.rcalibration_MS` functions for making predictions.

unlike GaSP calibration and S-GaSP calibration. Because model discrepancy is modeled, predictive accuracy by GaSP and S-GaSP calibration is typically higher than no-discrepancy calibration. The **RobustCalibration** package can handle model calibration and prediction at a wide range of settings. We first introduce the main functions of the package.

2.1 Main functions

The **RobustCalibration** package can be used for estimating unobservable parameters from computer models and predicting reality. The conventional model calibration and prediction from a single source of field observations are achieved by `rcalibration` and `predict.rcalibration`, as shown in upper blue and yellow boxes in Figure 1, respectively. The function `rcalibration` allows users to call either a Markov chain Monte Carlo (MCMC) algorithm for posterior sampling or a numerical optimization algorithm for computing the maximum likelihood estimator (MLE) of the parameters. Additional arguments can be specified to handle observations from repeated experiments, select a different trend or discrepancy function, and build a surrogate model for approximating expensive computer models, shown in Figure 1. The `rcalibration` function returns an object of the `rcalibration` S4 class with posterior samples or MLE of the parameters, shown in the upper orange box in Figure 1. Then `rcalibration` S4 class is used as the input in the `predict.rcalibration` function to perform predictions on a set of test inputs. The `predict.rcalibration` function returns a `predictobj.rcalibration` S4 class, which contains the predictive mean of the calibrated computer model, estimated trend, and discrepancy function. The estimated interval at a given quantile can also be returned by specifying the vector of the `interval_est` argument in the `predict.raclibration` function.

In some applications, we have different sources or types of observations. For instance, in calibrating geophysical models, time series of continuous GPS observations and satellite radar interferogram can be jointly used to estimate unobservable parameters (Anderson et al., 2019). In the **RobustCalibration** package, we implement parameter estimation for observations from multiple sources by the `rcalibration_MS` function, which returns a `rcalibration_MS` S4 class that contains posterior samples, shown in Figure 1. Since each source of observations can induce a separate set of model parameters, making the dimension of the parameter space large, the MLE is unstable, and as such is not implemented in `rcalibration_MS`. The `rcalibration_MS` class can be used as an input into the `predict_MS.rcalibration_MS` function for predicting the reality.

2.2 The rcalibration function

To use the `rcalibration` function, users need to specify 4 inputs: 1) `design`, an $n \times p_x$ matrix of observable inputs, 2) `observations`, field observations that can have three types specified below,

3) `theta_range`, a $p_\theta \times 2$ matrix of the lower and upper bounds of calibration parameters, and 4) either `math_model`, a function of mathematical model, or `input_simul` and `output_simul`, simulation runs by for building the emulator. The **RobustCalibration** package can handle three different kinds of field observations as specified by the argument `observations`. First, one can input an n-`vector` $(y^F(\mathbf{x}_1), ..., y^F(\mathbf{x}_n))^T$, when one field datum is available at each observable input. Second, when k repeated measurements are available for each observable input, one can input an $n \times k$ `matrix` of field observations, where each row contains k replicates of one observable input. Third, one can input a `list`, where each element contains k_i replications at the ith observable input, for $i = 1, ..., n$.

An important feature of the **RobustCalibration** package is the flexible specification of the computer model. Users can specify a mathematical model by supplying a generic `R` function through the argument `math_model` and selecting the default choice of the simulator `simul_type=1`. The second choice is to use an emulator to predict the computer model by choosing `simul_type=0`. Users need to input a set of simulation runs by `input_simul` and `output_simul` to call an emulator by the **RobustGaSP** package for this choice. Users can either give a $D \times (p_x + p_\theta)$ matrix of input and a D-vector of output to `input_simul` and `output_simul` arguments, for calling `rgasp` function from **RobustGaSP** for emulation, or give a $D \times p_\theta$ matrix of input and a $D \times p_x$ matrix of output to `input_simul` and `output_simul`, to call for `ppgasp` function from **RobustGaSP** for emulation. To account for stochastic error or numerical approximation error in the simulator, we also allow users to specify a nugget parameter estimated by the simulator data by the argument `simul_nug=T`.

The `rcalibration` function contains a few optional arguments. First, users can specify GaSP or S-GaSP discrepancy models by the argument `discrepancy_type="GaSP"` or `discrepancy_type="S-GaSP"`, respectively. The default choice is to assume a S-GaSP discrepancy model. The model without a discrepancy can be specified by the argument `discrepancy_type="no-discrepancy"`. Second, a trend can be specified by the argument `X`. By default, the trend (or mean) of the calibration model is zero. Third, the parameter estimation approach can be specified by the `method` argument. The default choice is `method="post_sample"`, where the posterior samples will be drawn, and stored in the slot `post_sample` of the `rcalibration` class. The MLE can be specified by the argument `method="mle"`, and the estimated parameters are stored in the slot `param_est` of the `rcalibration` class. Users can specify the number of MCMC samples and burn-in samples by the argument `S` and `S_0`, respectively. A vector containing the standard deviation of the proposal distribution for the calibration parameters, the logarithm of the inverse range parameters, and the logarithm of the nugget parameter can be specified by the `sd_proposal` argument. Furthermore, users can "thin" the MCMC samples and only record a subset by the argument `thinning`. For instance, `thinning=5` means only 1/5 of the posterior samples will be recorded. Besides, the inverse variances of noises in the field observations can be specified by the `output_weights`. Finally, arguments `initial_values` and `num_initial_starts` can be specified in numerical optimization to find MLE, and when posterior sampling is used, initial values of the calibration parameters can be specified through the argument `initial_values`.

The object `rcalibration` created by the `rcalibration` function have a few key properties. First of all, if we have `method='post_sample'`, the after burn-in posterior samples of parameters will be stored in the slot `post_sample`, where each row contains posterior samples in one iteration. The first p_θ columns contain posterior samples of the calibration parameters $\boldsymbol{\theta}$. In the no-discrepancy calibration, the $p_\theta + 1$ column of `post_sample` contains posterior draws of the noise variance σ_0^2 and the $p_\theta + 2$ to $p_\theta + q + 1$ columns contain the trend parameters $\boldsymbol{\theta}_m$, for a non-zero mean basis. In the GaSP or S-GaSP calibration, the $p_\theta + 1$ to $p_\theta + p_x + 1$ columns record posterior samples of the log inverse range and log nugget parameters. The $p_\theta + p_x + 2$ to $p_\theta + p_x + q$ columns record the noise variance parameter and trend parameters if they are specified. The parameter λ_z in S-GaSP are recorded in slot `lambda_z`. The indices of the accepted proposed samples are recorded in slots `accept_S` as one of the diagnostic statistics.

2.3 The predict.rcalibration function

After calling the `rcalibration` function, an object of the `rcalibration` S4 class will be created, and it can be used as an input into the `predict.rcalibration` function for predicting the reality on a specified matrix of test input using the argument `testing_input`. Besides, if the mean basis is specified by the argument `X` in `rcalibration` function, the user also needs to specify the mean basis of the reality at the test inputs by the argument `X_testing` in function `predict.rcalibration`. If the emulator was not constructed in `rcalibration`, users also need to specify the mathematical model by the argument `math_model` in the `predict.rcalibration` function. Finally, a predictive interval can be specified by the argument `interval_est`. For instance, the 95% predictive interval can be obtained by the `interval_est=c(0.025,0.975)` argument. The interval of the field data will be computed if `interval_data=T`. Otherwise, the predictive interval of the reality will be computed. If the variance parameter of the noise in field data was specified by the `output_weights`, users should also specify the variance of test data by `testing_output_weights`, which affects the predictive interval of test data.

After calling `predict.rcalibration` function, an object `predictobj.rcalibration` will be created and it contains three different predictors for reality. The slot `math_model_mean_no_trend` gives the predictive mean based on the calibrated computer model (f^M). The slot `math_model_mean` gives the predictive mean based on the calibrated computer model and the estimated trend ($f^M + \mu$), if the basis functions of the trend at the observable input and the test input are specified through `X` and `X_testing` in `rcalibration` and `rcalibration.predict`, respectively. The slot `mean` gives the predictive mean based on the calibrated computer model, estimated trend, and discrepancy ($f^M + \mu + \delta$). If `interval_est` is specified, a matrix of the intervals will be created for quantifying the uncertainty of predictions.

2.4 The rcalibration_MS function and the predict_MS.rcalibration_MS function

Model calibration and predictions using multiple sources or different types of data are implemented by `rcalibration_MS` and `rcalibration_MS.predict_MS` functions, respectively. One needs to specify 4 inputs: `design`, `observations`, `theta_range` and a form of mathematical model. Suppose we have a data set produced by k sources of observations. The `design` is a list of k elements, where each element is a matrix of observable input for each source. The argument `observations` takes a list of field observations, where each element is a vector of field observations for each source. In principle, one can have different number of observations from each source and the type of the observations may not be the same. The argument `theta_range` is a $p_\theta \times 2$ matrix of range of parameters, where the ith row contains the minimum and maximum values of the ith coordinate of the calibration parameter vector $\boldsymbol{\theta}$. Furthermore, if closed form expressions of mathematical models are available for each source of data, users can input a list of functions into the `math_model` for each source of data. We also allow users to emulate the expensive simulation. In this scenario, one needs to let `simul_type` be a vector of 1s, indicating emulators will be called for each computer model.

Additional arguments can be specified in `rcalibration_MS`. For instance, the argument `index_theta` takes a list of indices for the associated calibration parameter for each computer model. Suppose we have three calibration parameters $\boldsymbol{\theta} = (\theta_1, \theta_2, \theta_3)$, and the computer model for first source of data is related to the first two calibration parameters and the computer model for second source of data is related to second two calibration parameters. Then `index_theta` is a list where `index_theta[[1]]=c(1,2)`, and `index_theta[[2]]=c(2,3)`. More arguments will be introduced along with the examples. After calling `rcalibration_MS`, an S4 class `rcalibration_MS` will be built, and used as an input to `predict_MS.rcalibration_MS` for making predictions.

3 Methods and examples

3.1 No-discrepancy calibration

Let us start with the simplest method: no-discrepancy calibration. First, suppose we have a vector of real-valued field observations $\mathbf{y}^F = (y^F(\mathbf{x}_1), ..., y^F(\mathbf{x}_n))^T$ at n observable inputs, and the corresponding computer model output is denoted by $\mathbf{f}^M_{\boldsymbol{\theta}} = (f^M(\mathbf{x}_1, \boldsymbol{\theta}), ..., f^M(\mathbf{x}_n, \boldsymbol{\theta}))^T$ for any calibration parameters $\boldsymbol{\theta}$. The vector of measurement noise is assumed to follow $\boldsymbol{\epsilon} \sim \mathcal{MN}(0, \sigma_0^2 \boldsymbol{\Lambda})$, where the covariance of the noise is diagonal with the ith term being $\sigma_0^2 \Lambda_{ii}$ and $\mathcal{MN}$ denote the multivariate normal distribution. In the default setting, σ_0^2 is estimated from the data and $\boldsymbol{\Lambda} = \mathbf{I}_n$, where $\mathbf{I}_n$ is an $n \times n$ identity matrix. Users can manually specify the inverse of the diagonal terms of $\boldsymbol{\Lambda}$ by the `output_weights` argument in `rcalibration` and `rcalibration_MS` functions, as the variances of measurement errors can be different. We assume the diagonal matrix $\boldsymbol{\Lambda}$ is given in this section.

For a no-discrepancy calibration model in (1), the MLE of the variance of the noise is $\hat{\sigma}_0^2 = S_0^2/n$ with $S_0^2 = (\mathbf{y}^F - \mathbf{f}^M_\theta)^T \boldsymbol{\Lambda}^{-1} (\mathbf{y}^F - \mathbf{f}^M_\theta)$. The likelihood function is given in the Appendix. As $\boldsymbol{\Lambda}$ is a diagonal matrix with diagonal terms denoted as $\Lambda_{ii} := 1/w_i$, the profile likelihood follows

$$\mathcal{L}(\boldsymbol{\theta} \mid \hat{\sigma}_0^2) \propto \left\{ \sum_{i=1}^{n} w_i \left(y^F(\mathbf{x}_i) - f^M(\mathbf{x}_i, \boldsymbol{\theta}) \right)^2 \right\}^{-n/2},$$

where $w_i = 1$ is used in the default scenario. Maximizing the profile likelihood of a no-discrepancy model is equivalent to minimizing the weighted squared error $\sum_{i=1}^n w_i \left(y^F(\mathbf{x}_i) - f^M(\mathbf{x}_i, \boldsymbol{\theta}) \right)^2$. We numerically find $\boldsymbol{\theta}$ by the low-storage quasi-Newton optimization method (Liu and Nocedal, 1989) implemented in `lbfgs` function in the **nloptr** package (Ypma (2014)) for optimization.

In the **RobustCalibration** package, we allow the users to specify the trend or mean function modeled as $\boldsymbol{\mu}(\mathbf{x}_i) = \mathbf{h}(\mathbf{x})\boldsymbol{\theta}_m = \sum_{t=1}^q h_t(\mathbf{x})\theta_{m,t}$ for any $\mathbf{x}$, where $\boldsymbol{\theta}_m = (\theta_{m,1}, ..., \theta_{m,q})^T$ is a vector of trend

parameters. Maximizing the profile likelihood of a no-discrepancy model with a trend is equivalent to minimizing the squared error loss:

$$\hat{\boldsymbol{\theta}}_m^{LS} = \text{argmin}_{\boldsymbol{\theta}} \sum_{i=1}^{n} w_i \left(y^F(\mathbf{x}_i) - f^M(\mathbf{x}_i, \boldsymbol{\theta}) - \mathbf{h}(\mathbf{x}_i)\hat{\boldsymbol{\theta}}_m^{LS} \right)^2,$$

where the solution follows $\hat{\boldsymbol{\theta}}_m^{LS} = (\mathbf{H}^T\boldsymbol{\Lambda}^{-1}\mathbf{H})^{-1}\mathbf{H}^T\boldsymbol{\Lambda}^{-1}(\mathbf{y}^F - \mathbf{f}_\theta^M)$ with $\mathbf{H} = (\mathbf{h}^T(\mathbf{x}_1), ..., \mathbf{h}^T(\mathbf{x}_n))^T$ being an $n \times q$ matrix of the mean basis.

After calling the `rcalibration` function with `method='mle'`, the point estimator $\hat{\boldsymbol{\theta}}_{LS}$ is recorded in the slot `param_est` in the `rcalibration` class. The MLE is a computationally cheap way to obtain estimates of calibration parameter vector $\boldsymbol{\theta}$. After obtaining the MLE, we can plug the MLE into the computer model for predicting reality: $f^M(\mathbf{x}_i, \hat{\boldsymbol{\theta}}) + h(\mathbf{x}_i)\hat{\boldsymbol{\theta}}_m^{LS}$, using the `predict` function.

We also implement the MCMC algorithm for Bayesian inference. The posterior distribution follows

$$p(\boldsymbol{\theta}_m, \sigma_0^2, \boldsymbol{\theta} \mid \mathbf{y}^F) \propto p(\mathbf{y}^F \mid \boldsymbol{\theta}, \boldsymbol{\theta}_m, \sigma_0^2)\pi(\boldsymbol{\theta})\pi(\boldsymbol{\theta}_m, \sigma_0^2). \tag{3}$$

We assume an objective prior for the mean and variance parameters $\pi(\boldsymbol{\theta}_m, \sigma_0^2) \propto 1/\sigma_0^2$. The posterior distribution $p(\boldsymbol{\theta}_m, \sigma_0^2 \mid \boldsymbol{\theta}, \mathbf{y}^F)$ can be sampled by the Gibbs algorithm, since the prior is conjugate. We assume that the default choice of the prior of the calibration parameters $\pi(\boldsymbol{\theta})$ is uniform over the parameter space and sample $(\boldsymbol{\theta} \mid \mathbf{y}^F, \boldsymbol{\theta}_m, \sigma_0^2)$ using the Metropolis algorithm. The posterior distributions and predictions from the posterior samples will be discussed in the Appendix.

Here we show one example from Bayarri et al. (2007b), where data are observed from unknown reality: $y^F(x) = 3.5\exp(-1.7x) + 1.5 + \epsilon$, where $\epsilon \sim \mathcal{N}(0, 0.3^2)$. We have 30 observations at 10 inputs in the domain $x_i \in [0, 3]$, for $i = 1, 2, ..., 10$, each containing 3 replications. The computer model is $f^M(x, \theta) = 5\exp(-\theta x)$ with $\theta \in [0, 50]$. The goal is to estimate the calibration parameters and predict the unknown reality $(3.5\exp(-1.7x) + 1.5)$ at $x \in [0, 5]$. The observable inputs and observations from Bayarri et al. (2007b) can be generated by the code below.

```
R> input=c(.110, .432, .754, 1.077, 1.399, 1.721, 2.043, 2.366, 2.688,3.010)
R> n=length(input)
R> k=3
R> output=t(matrix(c(4.730,4.720,4.234,3.177,2.966,3.653,1.970,2.267,2.084,2.079,
+                2.409,2.371,  1.908,1.665,1.685, 1.773,1.603,1.922,1.370,1.661,
+                1.757, 1.868,1.505,1.638,1.390, 1.275,1.679,1.461,1.157,1.530),k,n))
R> Bayarri_07<-function(x,theta){
+   5*exp(-x*theta)
+ }
R> theta_range=matrix(c(0,50),1,2)
```

Here the mathematical model has a closed-form expression so we code it as a function called `Bayarri_07`. The parameter range of calibration parameter θ is given by a vector called `theta_range`.

The no-discrepancy calibration is implemented by the code below.

```
R> set.seed(1)
R> X=matrix(1,n,1)
R> m_no_discrepancy=rcalibration(input, output,math_model = Bayarri_07,
+                                theta_range = theta_range, X =X,
+                                have_trend = T,discrepancy_type = 'no-discrepancy')
R> m_no_discrepancy
type of discrepancy function:  no-discrepancy
number of after burn-in posterior samples:  8000
0.1385 of proposed calibration parameters are accepted
median and 95\% posterior credible interval of calibration parameter  1 are
2.951997 2.264356 3.930725
```

Here we use a constant mean basis using the mean (or trend) argument `X`, such that the mathematical model is $f^M(x, \theta) + \mu$. We draw $10,000$ posterior samples with the first $2,000$ samples used as burn-in samples. One can adjust the number of posterior and burn-in samples by argument `S` and `S_0` in the `calibration` function, respectively. We found that the median of the posterior samples of θ is 2.95, and the 95% posterior interval is around $(2.26, 3.93)$. Around 14% of the samples were accepted by the Metropolis algorithm. Here the first p_θ terms in `sd_proposal` is the standard deviation of the proposal distribution of the calibration parameter, where the default choice is 0.05, and the next $p_x + 1$ parameters are the standard deviation of the inverse logarithm of the range parameter and nugget parameters in GaSP and S-GaSP calibration, discussed in the next subsection. The code below reduces `sd_proposal` to 0.025 of the range of the parameter space to make a larger proportion of the posterior

samples accepted by the algorithm:

```
R> m_no_discrepancy_small_sd=rcalibration(input, output,math_model = Bayarri_07,
+                           theta_range = theta_range,
+                           sd_proposal = c(rep(0.025,dim(theta_range)[1]),
+                                           rep(0.25,dim(as.matrix(input) )[2]),0.25),
+                           X =X, have_trend = T,discrepancy_type = 'no-discrepancy')
R> m_no_discrepancy_small_sd
type of discrepancy function:  no-discrepancy
number of after burn-in posterior samples:  8000
0.2613 of proposed calibration parameters are accepted
median and 95\% posterior credible interval of calibration parameter  1 are
2.935007 2.19352 3.933492
```

Around 26% of the posterior samples are accepted. In this example, the median and 95% posterior credible intervals are similar to the one having a proposal distribution with smaller standard deviation.

The following code gives the predictions and the 95% predictive interval of the reality for 200 test inputs equally spaced at $x \in [0,5]$. Since we use a constant trend in model calibration, we also specify a constant trend for making predictions, through the argument `X_testing` in the `predict` function:

```
R> testing_input=seq(0,5,5/199)
R> X_testing=matrix(1,length(testing_input),1)
R> m_no_discrepancy_pred=predict(m_no_discrepancy,testing_input,math_model=Bayarri_07,
+                                interval_est=c(0.025, 0.975),X_testing=X_testing)
```

Posterior samples of the calibration parameter θ and the mean parameter θ_m in the no-discrepancy calibration are plotted as the red rectangles in the right panel in Figure 2. The predictive mean and 95% predictive interval are graphed as the red curve and grey shaded area in the left panel, respectively. Compared to models with an estimated discrepancy function, the posterior samples from the no-discrepancy calibration are more concentrated, and the average length of 95% predictive interval is shorter. However, a large proportion of reality at $x \in [0,1.5]$ is not covered by the 95% posterior credible interval, due to large discrepancy between the mathematical model and reality in this domain. Next, we introduce two discrepancy models to solve this problem.

3.2 Gaussian stochastic process models of discrepancy functions

The discrepancy function can be modeled as a GaSP, meaning that for any $\{\mathbf{x}_1, ..., \mathbf{x}_n\}$, the marginal distribution of the discrepancy function follows a multivariate normal distribution:

$$(\delta(\mathbf{x}_1), ..., \delta(\mathbf{x}_n))^T \mid \boldsymbol{\mu}, \sigma^2, \mathbf{R} \sim \mathcal{MN}(\boldsymbol{\mu}, \sigma^2\mathbf{R}),$$

where $\boldsymbol{\mu} = (\mu(\mathbf{x}_1), ..., \mu(\mathbf{x}_n))^T$ is a vector of the mean (or trend), σ^2 is a variance parameter, and $\mathbf{R}$ is the correlation matrix between outputs.

Similar to the no-discrepancy calibration, we allow user to model the trend as $\boldsymbol{\mu}(\mathbf{x}_i) = \mathbf{h}(\mathbf{x}_i)\boldsymbol{\theta}_m = \sum_{t=1}^{q} h_t(\mathbf{x}_i)\theta_{m,t}$, where $\mathbf{h}(\mathbf{x}_i)$ is a row vector of mean basis and $\boldsymbol{\theta}_m$ is a vector of trend parameters. Since the trend is often modeled in the computer model, the default value of the trend parameters is $\mathbf{0}$.

The (i,j)th element of the correlation matrix is modeled by a kernel function $R_{i,j} = K(\mathbf{x}_i, \mathbf{x}_j)$. We implement the separable kernel $K(\mathbf{x}_i, \mathbf{x}_j) = \prod_{l=1}^{p_x} K_l(\mathbf{x}_i, \mathbf{x}_j)$, where K_l models the correlation between the lth coordinate of the observable input, for $l = 1, ..., p_x$. Some frequently used kernel functions are listed in Table 1. The range parameters γ are estimated by default.

We denote the nugget parameter by the noise variance to signal variance ratio $\eta = \sigma_0^2/\sigma^2$. Marginalizing out the random discrepancy function $\delta(\cdot)$, one has $\mathbf{y}^F \sim \mathcal{MN}(\mathbf{f}_\theta^M + \mathbf{H}\boldsymbol{\theta}_m, \sigma_0^2\tilde{\mathbf{R}})$, where $\tilde{\mathbf{R}} = \mathbf{R}/\eta + \boldsymbol{\Lambda}$. Here by default $\boldsymbol{\Lambda} = \mathbf{I}_n$, and we allow users to specify the inverse diagonal terms of $\boldsymbol{\Lambda}$ by the argument `weights`: $w_i = 1/\Lambda_{ii}$, for $i = 1, ..., n$.

The MLE of the trend and noise variance parameters in the GaSP calibration follows $\hat{\boldsymbol{\theta}}_m = \left(\mathbf{H}^T\tilde{\mathbf{R}}^{-1}\mathbf{H}\right)^{-1}\mathbf{H}^T\tilde{\mathbf{R}}^{-1}(\mathbf{y} - \mathbf{f}_\theta^M)$ and $\hat{\sigma}_0^2 = S_K^2/n$, where $S_K^2 = (\mathbf{y} - \mathbf{f}_\theta^M - \mathbf{H}^T\hat{\boldsymbol{\theta}}_m)^T\tilde{\mathbf{R}}^{-1}(\mathbf{y} - \mathbf{f}_\theta^M - \mathbf{H}^T\hat{\boldsymbol{\theta}}_m)$. Plugging in the MLE of trend and noise variance parameters, one can numerically maximize the profile likelihood to obtain the calibration, kernel and nugget parameters in the GaSP calibration:

$$(\hat{\boldsymbol{\theta}}, \hat{\boldsymbol{\gamma}}, \hat{\eta}) = \underset{(\boldsymbol{\theta},\boldsymbol{\gamma},\eta)}{\operatorname{argmax}} \mathcal{L}_K(\boldsymbol{\theta}, \boldsymbol{\gamma}, \eta \mid \hat{\boldsymbol{\theta}}_m, \hat{\sigma}_0^2) \tag{4}$$

where the closed form expression of the profile likelihood $\mathcal{L}_K(\boldsymbol{\theta}, \boldsymbol{\gamma}, \eta \mid \hat{\boldsymbol{\theta}}_m, \hat{\sigma}_0^2)$ with kernel $K(.,.)$ is given in the Appendix.

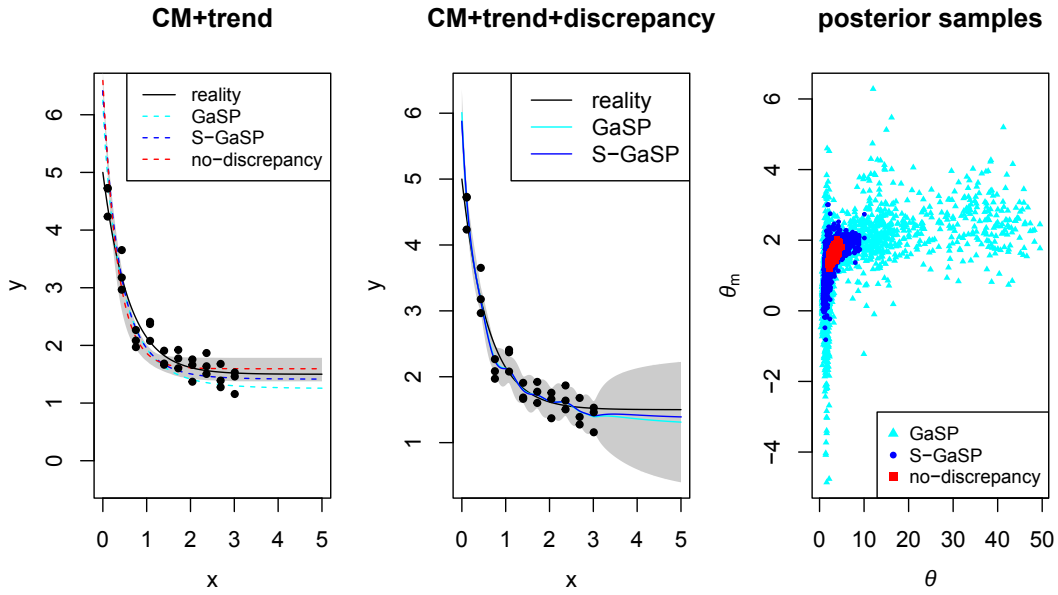

Figure 2: Predictions and posterior samples by the **RobustCalibration** package for the example introduced in Bayarri et al. (2007b). In the left and middle panels, the unknown reality is graphed as black curves, and field observations that contain with 3 replicates at each observable input are plotted as black dots. In the left panel, predictions of reality based on calibrated computer model with trend by GaSP, S-GaSP and no-discrepancy calibration are graphed as light blue, blue and red curves, respectively. 'CM' denotes the computer model and the shaded area is the 95% posterior predictive interval by the no-discrepancy calibration. In the middle panel, predictions based on the calibrated computer model, discrepancy, and the trend by GaSP and S-GaSP are graphed as light blue and blue curves, respectively. The shaded area is the 95% predictive interval by the GaSP calibration. After burn-in posterior samples of calibration parameter θ and mean parameter θ_m are plotted in the right panel.

After obtaining the estimation of the parameters, the predictive distribution of reality on any $\mathbf{x}^*$ can be obtained by plugging in the estimator

$$y^R(\mathbf{x}^*) \mid \mathbf{y}^F, \hat{\boldsymbol{\theta}}_m, \hat{\sigma}_0^2, \hat{\boldsymbol{\theta}}, \hat{\gamma}, \hat{\eta} \sim \mathcal{N}(\hat{\mu}(\mathbf{x}^*), \hat{\sigma}_0^2(K^*/\hat{\eta} + \Lambda^*)), \tag{5}$$

where Λ^* is the weight at $\mathbf{x}^*$ set to be 1 by default, and

$$\begin{aligned}\hat{\mu}(\mathbf{x}^*) &= f^M(\mathbf{x}^*, \hat{\boldsymbol{\theta}}) + \mathbf{h}(\mathbf{x}^*)\hat{\boldsymbol{\theta}}_m + \mathbf{r}^T(\mathbf{x}^*)\tilde{\mathbf{R}}^{-1}(\mathbf{y}^F - \mathbf{f}(\mathbf{x}_{1:n}, \hat{\boldsymbol{\theta}}) - \mathbf{H}\hat{\boldsymbol{\theta}}_m),\\ K^* &= K(\mathbf{x}^*, \mathbf{x}^*) - \mathbf{r}^T(\mathbf{x}^*)\tilde{\mathbf{R}}^{-1}\mathbf{r}(\mathbf{x}^*),\end{aligned}$$

with $\mathbf{r}(\mathbf{x}^*) = (K(\mathbf{x}^*, \mathbf{x}_1), ..., K(\mathbf{x}^*, \mathbf{x}_n))^T$. Though the predictive mean $\hat{\mu}(\mathbf{x}^*)$ may be the most accurate for predictions, the calibrated computer model $f^M(\mathbf{x}^*, \hat{\boldsymbol{\theta}})$ is sometimes of interest for predicting the reality, due to its interpretability. Thus, we recorded three slots for predicting the reality: 1) the calibrated computer model $f^M(\mathbf{x}^*, \hat{\boldsymbol{\theta}})$ by `math_model_mean_no_trend`, 2) The calibrated computer model with trend $f^M(\mathbf{x}^*, \hat{\boldsymbol{\theta}}) + \mathbf{h}(\mathbf{x}^*)\hat{\boldsymbol{\theta}}_m$ by `math_model_mean` , and 3) the predictive mean $\hat{\mu}(\mathbf{x}^*)$ by `mean`. Furthermore, users can also output the predictive interval based on (5) by specifying `interval_est` in the `predict.rcalibration` function. For example, setting `interval_est=c(0.025,0.975)` gives the 95% predictive interval of the reality. Furthermore, if `interval_data=T`, the predictive interval of the data will be computed.

Numerically computing the MLE can be unstable as the profile likelihood $\mathcal{L}_K(\boldsymbol{\theta}, \gamma, \eta \mid \hat{\theta}_m, \hat{\sigma}_0^2)$ is typically nonlinear and nonconvex with respect to the parameters $(\boldsymbol{\theta}, \gamma, \eta)$. To obtain the uncertainty assessment of the estimates and to avoid instability in the numerical search in computing the MLE, we implement the MCMC algorithm for Bayesian inference. The posterior distribution is

$$p(\boldsymbol{\theta}_m, \sigma_0^2, \boldsymbol{\theta}, \gamma, \eta \mid \mathbf{y}^F) \propto p_K(\mathbf{y}^F \mid \boldsymbol{\theta}, \gamma, \eta, \boldsymbol{\theta}_m, \sigma_0^2)\pi(\boldsymbol{\theta})\pi(\boldsymbol{\theta}_m, \sigma_0^2, \gamma, \eta), \tag{6}$$

where $p_K(.)$ is a multivariate normal density with covariance function of the discrepancy being $\sigma^2 K(\cdot, \cdot)$,

GaSP calibration	Product kernel ($K(\mathbf{d}) = \prod_{l=1}^{p_x} K_l(d_l)$)
Matérn $\alpha = 5/2$	$K_l(d_l) = \left(1 + \frac{\sqrt{5}d_l}{\gamma_l} + \frac{5d_l^2}{3\gamma_l^2}\right) \exp\left(-\frac{\sqrt{5}d_l}{\gamma_l}\right)$
Matérn $\alpha = 3/2$	$K_l(d_l) = \left(1 + \frac{\sqrt{3}d_l}{\gamma_l}\right) \exp\left(-\frac{\sqrt{3}d_l}{\gamma_l}\right)$
Power exponential	$K_l(d_l) = \exp\left\{-\left(\frac{d_l}{\gamma_l}\right)^{\alpha_l}\right\}, 0 < \alpha_l \leqslant 2$
S-GaSP calibration	Scaled kernel ($K_{Z_d}(\mathbf{x}_a, \mathbf{x}_b) = K(\mathbf{x}_a, \mathbf{x}_b) - \mathbf{r}^T(\mathbf{x}_a)\mathbf{R}_z^{-1}\mathbf{r}(\mathbf{x}_b)$)

Table 1: Kernel functions implemented in **RobustCalibration**. For any $\mathbf{x}_a, \mathbf{x}_b \in \mathcal{X}$, denote $\mathbf{d} = \mathbf{x}_a - \mathbf{x}_d = (d_1, ..., d_{p_x})^T$. For any kernel K, the discretized scaled kernel $K_{Z_d}(\mathbf{x}_a, \mathbf{x}_b)$ with discretization points on observed points $\mathbf{x}_1, ..., \mathbf{x}_n$ is implemented in the S-GaSP calibration, where $\mathbf{R}_z := \mathbf{R} + n\mathbf{I}_n/\lambda_z$, and $\mathbf{r}(\mathbf{x}) = (K(\mathbf{x}, \mathbf{x}_1), ..., K(\mathbf{x}, \mathbf{x}_n))^T$ for any $\mathbf{x}$.

and $\pi(\boldsymbol{\theta})$ is the prior of the calibration parameters, assumed to be uniform over the parameter space. The prior of the mean, variance, kernel and nugget parameters takes the form below

$$\pi(\boldsymbol{\theta}_m, \sigma_0^2, \boldsymbol{\gamma}, \eta) \propto \frac{\pi(\boldsymbol{\gamma}, \eta)}{\sigma_0^2},$$

where the usual reference prior of the trend and variance parameters $\pi(\boldsymbol{\theta}_m, \sigma_0^2) \propto 1/\sigma_0^2$ is assumed. We use jointly robust prior for the kernel and nugget parameters $\pi(\boldsymbol{\gamma}, \eta)$, as it has a similar tail density decay rate as the reference prior, but it is easier and more robust to compute (Gu, 2019). Closed form expressions of the posterior distributions and MCMC algorithm are provided in the Appendix.

In **RobustCalibration**, after burn-in posterior samples $(\boldsymbol{\theta}_m^{(i)}, (\sigma_0^2)^{(i)}, \boldsymbol{\theta}^{(i)}, \boldsymbol{\gamma}^{(i)}, \eta^{(i)})$ are recorded for $i = S_0 + 1, ..., S$. Users can record a subset of posterior samples to reduce storage by thinning the Markov chain. For instance, `thinning=5` in the `rcalibration` function records 1/5 of posterior samples.

We also output three types of predictions. For instance, suppose we obtain S posterior samples with first S_0 samples used as burn-in samples, the predictive mean of the reality can be computed by

$$\hat{\mu}(\mathbf{x}^*) = \sum_{i=S_0+1}^{S} f^M(\mathbf{x}^*, \boldsymbol{\theta}^{(i)}) + \mathbf{h}(\mathbf{x}^*)\boldsymbol{\theta}_m^{(i)} + (\mathbf{r}^{(i)}(\mathbf{x}^*))^T(\tilde{\mathbf{R}}^{(i)})^{-1}(\mathbf{y}^F - \mathbf{f}^M(\mathbf{x}_{1:n}, \boldsymbol{\theta}^{(i)}) - \mathbf{H}\boldsymbol{\theta}_m^{(i)}),$$

where $\mathbf{r}^{(i)}(\mathbf{x}^*)$ and $\tilde{\mathbf{R}}^{(i)}$ are obtained by plugging in the ith posterior samples. The predictive interval can also be computed by specifying the argument `interval_est` in `predict.rcalibration` function.

Code below implements the GaSP calibration of parameter estimation and predictions for the example in (Bayarri et al., 2007b) that was discussed in the previous subsection.

```
R> m_gasp=rcalibration(input, output,math_model = Bayarri_07,theta_range = theta_range,
+                      X =X, have_trend = T,discrepancy_type = 'GaSP')
R> m_gasp_pred=predict(m_gasp,testing_input,math_model=Bayarri_07,
+                      interval_est=c(0.025, 0.975),X_testing=X_testing)
```

Predictions and posterior samples from the GaSP calibration are plotted in Figure 2. The associated predictive error is shown in Table 2. Comparing the first two rows in Table 2, the predictive RMSE in the no-discrepancy calibration is around 0.25, and it decreases to 0.15, after adding the discrepancy function modeled by GaSP. Around 97.5% of the held-out truth is covered by the 95% predictive interval of the reality in the GaSP calibration. In comparison, the 95% predictive interval of the reality in the no-discrepancy calibration only covers around 80% of the held-out reality.

As shown in the right panel in Figure 2, the posterior samples from GaSP spread over a large range, reflecting the large uncertainty in parameter estimation. This is because the discrepancy modeled by GaSP with the frequently used kernel listed in Table 1 is very flexible. As a result, the calibrated computer model by GaSP can be less accurate in predicting the reality. To address this problem, we introduce a new approach that induces a scaled kernel to constrain the discrepancy function.

3.3 Scaled Gaussian stochastic process models of discrepancy functions

Scaling the kernel of GaSP for model calibration was introduced in Gu and Wang (2018), where the random L_2 distance between the discrepancy model was scaled to have more prior probability mass at small values, as small distance indicates the computer model fits the reality well. Note that we

	RMSE (CM+trend)	RMSE (with discrepancy)	$P_{CI}(95\%)$	$L_{CI}(95\%)$
GaSP	0.253	0.151	0.975	0.880
S-GaSP	0.228	0.131	0.955	1.15
No-discrepancy	0.250	/	0.795	0.409

Table 2: Predictive accuracy and uncertainty assessment. The RMSE of the out-of-sample prediction based on the calibrated computer model (CM) and trend are shown in the second column. The predictive RMSE by the summation of the calibrated computer model, trend and discrepancy is given in the third column. The proportion of the held-out reality covered in the 95% predictive interval, and average lengths of predictive intervals are given in the last two columns, respectively.

leave σ^2 as a free parameter to be estimated from data, and thus S-GaSP is still a flexible model of discrepancy.

In **RobustCalibration**, we implemented the discretized S-GaSP to scale the random mean squared error between the reality and computer model:

$$\delta_{z_d}(\mathbf{x}) = \left\{ \delta(\mathbf{x}) \mid \frac{1}{n}\sum_{i=1}^{n} \delta(\mathbf{x}_i)^2 = Z_d \right\}, \tag{7}$$

with the subscript 'd' denoting discretization, and the density of Z_d is defined as

$$p_{Z_d}(z) = \frac{g_{Z_d}(z)\, p_\delta(Z_d = z)}{\int_0^\infty g_{Z_d}(t)\, p_\delta(Z_d = t)\, dt}, \tag{8}$$

where $p_\delta(Z_d = z)$ is the density of Z_d induced by the GaSP with kernel $K(\cdot,\cdot)$, and $g_Z(\cdot)$ is a nondecreasing function that places more probability mass on smaller values of the L_2 loss. For a frequently used kernel function in Table 1, the probability measure of the GaSP places a large probability at large L_2 loss $p_\delta(Z_d = z)$ when the correlation is large, dragging the calibrated computer model away from the reality. In the S-GaSP calibration, the measure for Z_d was scaled to have more probability mass near zero by a scaling function $g_{Z_d}(z)$. The default scaling function is assumed to be an exponential distribution,

$$g_{Z_d}(z) = \frac{\lambda_z}{2\sigma^2} \exp\left(-\frac{\lambda_z z}{2\sigma^2}\right), \tag{9}$$

where λ_z controls how large the scaling factor is. Large λ_z concentrates more prior probability mass at the origin. As shown in Gu et al. (2022), $\lambda_z \propto \sqrt{n}$ guarantees the convergence of the calibration parameters to the ones that minimize the L_2 distance between the reality and computer model. We let the default choice be $\lambda_z = (\lambda||\tilde{\boldsymbol{\gamma}}||)^{-1/2}$, where $\lambda = (\sigma_0^2/\sigma^2 n)$ is the regularization parameter in the kernel ridge regression, and $\tilde{\boldsymbol{\gamma}} = (\gamma_1/L_1, ..., \gamma_{p_x}/L_{p_x})^T$, with L_i being the length of domain of the ith coordinate of the calibration parameter. We allow users to specify λ_z via the argument `lambda_z` in the `rcalibration` function.

The default choice of the scaling function in (9) has computational advantages. After marginalizing out Z_d, it follows from Lemma 2.4 in Gu and Wang (2018) that $\delta_{z_d}(\cdot)$ has the covariance function

$$\sigma^2 K_{z_d}(\mathbf{x}_a, \mathbf{x}_b) = \sigma^2 \left\{ K(\mathbf{x}_a, \mathbf{x}_b) - \mathbf{r}^T(\mathbf{x}_a) \left(\mathbf{R} + \frac{n\mathbf{I}_n}{\lambda_z} \right)^{-1} \mathbf{r}(\mathbf{x}_b) \right\}, \tag{10}$$

for any $\mathbf{x}_a, \mathbf{x}_b$, where $\mathbf{r}(\mathbf{x}) = (K(\mathbf{x}, \mathbf{x}_1), ..., K(\mathbf{x}, \mathbf{x}_n))^T$ for any $\mathbf{x}$. The covariance matrix of the S-GaSP model of $(\delta_z(\mathbf{x}_1), ..., \delta_z(\mathbf{x}_n))^T$ follows

$$\sigma^2 \mathbf{R}_z = \sigma^2 \left\{ \mathbf{R} - \mathbf{R} \left(\mathbf{R} + \frac{n\mathbf{I}_n}{\lambda_z} \right)^{-1} \mathbf{R} \right\}. \tag{11}$$

The closed-form expression in Equation (10) avoids sampling $(\delta_z(\mathbf{x}_1), ..., \delta_z(\mathbf{x}_n))$ to obtain Z_d from the posterior distribution, which greatly improves the computational speed.

For any covariance function $K(\cdot,\cdot)$ in the GaSP calibration, a scaled kernel was induced in S-GaSP calibration, shown in the last row of Table 1. The S-GaSP model of the discrepancy can be specified by the argument `discrepancy_type='S-GaSP'` in the `rcalibration` function. Similar to the model of GaSP-calibration, we also provide MLE and posterior samples for estimating the parameters, with arguments `method='mle'` and `method='post_sample'`, respectively. The MLE of parameters in S-GaSP

are computed by maximizing the profile likelihood

$$(\hat{\boldsymbol{\theta}}, \hat{\gamma}, \hat{\eta}) = \underset{(\boldsymbol{\theta},\gamma,\eta)}{\operatorname{argmax}} \mathcal{L}_{K_{Z_d}}(\boldsymbol{\theta}, \gamma, \eta, \mid \hat{\boldsymbol{\theta}}_m, \hat{\sigma}_0^2), \tag{12}$$

where $(\hat{\boldsymbol{\theta}}_m, \hat{\sigma}_0^2)$ denotes the MLE of trend and noise variance parameters in S-GaSP calibration, and $\mathcal{L}_{K_{Z_d}}$ is the profile likelihood with respect to the scaled kernel K_{Z_d} in (10). Furthermore, point predictions and intervals can be obtained by the `predict.rcalibration` function as well.

The code below gives S-GaSP calibration and prediction of the example in Bayarri et al. (2007b).

```
R> m_sgasp=rcalibration(input, output,math_model = Bayarri_07,theta_range = theta_range,
+                       X =X, have_trend = T,discrepancy_type = 'S-GaSP')
R> m_sgasp_pred=predict(m_sgasp,testing_input,math_model=Bayarri_07,
+                       interval_est=c(0.025, 0.975),X_testing=X_testing)
```

Predictions from S-GaSP are plotted as blue curves in Figure 2 and numerical comparisons are given in Table 2. With the discrepancy modeled by S-GaSP, the prediction of reality is more accurate at $x \in [0.5, 1.5]$, compared to no-discrepancy calibration. Note that predictions from the S-GaSP model also extrapolate the reality at $x \in [3, 5]$ accurately, because the calibrated computer model is closer to the truth. Furthermore, the 95% predictive interval by S-GaSP covers around 95% of the held-out samples.

To compare and illustrate different calibration approaches for differential equations, we discuss another example of the Lorenz-96 system used in modeling atmospheric dynamics (Lorenz, 1996). The mathematical model is written as the following differential equations:

$$\dot{x}_j(t) = (x_{j+1}(t) - x_{j-2}(t))x_{j-1}(t) - x_j(t) + \theta, \tag{13}$$

for $j = 1, ..., k$ states, with $k = 40$ and θ is a scalar value of the force. We further denote that $x_{-1} = x_{k-1}$, $x_0 = x_k$ and $x_{k+1} = x_1$ for any t. The latent variable x_j can model the atmospheric quantities, such as temperature or pressure, measured at k positions along a constant latitude circle. This model is widely used for data assimilation (Maclean and Spiller, 2020; Brajard et al., 2020).

We consider model calibration in two scenarios:

$$\text{Scenario 1:} \qquad y_j(t) = x_j(t) + \epsilon, \tag{14}$$

$$\text{Scenario 2:} \qquad y_j(t) = x_j(t) + 2t \sin\left(\frac{2\pi j}{k}\right) + \epsilon, \tag{15}$$

for $j = 1, ..., k$ and $\epsilon \sim \mathcal{N}(0, 1)$ being an independent Gaussian noise. In the first scenario, the mathematical model is the true model and in the second scenario, a discrepancy term is included for simulating the field data. The discrepancy is treated as unknown. We initialize the states by a multivariate normal distribution with the covariance generated from a Wishart distribution with k degrees of freedom and the scale matrix being a diagonal matrix. We use the Runge Kutta method with order 4 to numerically solve the system. Since the computer model is fast, we do not include an emulator. We simulate the reality at 40 time points, with a time step of 0.05 between the time points.

The unobserved reality simulated by the Lorenz-96 system is plotted in upper panel in Figure 3. Only 5% observations (i.e. 2 observations at each time point, plotted as black circles in upper middle panel) are available for model calibration. The posterior parameters of no-discrepancy calibration, GaSP and S-GaSP calibration are plotted in the upper right panel. Since the computer model (Lorenz-96) is the true model in this scenario, the no-discrepancy calibration is expected to perform well. Even though one allows a flexible discrepancy function, modeled as GaSP or S-GaSP, it seems parameters are estimated reasonably well in both approaches. In all methods, posterior samples are close to the true parameter ($\theta = 8$), and the range of the posterior samples is narrow compared to the parameter range $[-20, 20]^2$. Furthermore, the estimate states by the calibrated computer model and the reality are plotted in the lower panels in Figure 3. All methods have small estimation errors, as the parameters are estimated well.

The reality, observations, posterior samples, and predictions from the different calibration models of a discrepancy-included Lorenz-96 system are graphed in Figure 4. Note that since the discrepancy is included, there is no true calibration parameter. The states estimated by the no-discrepancy calibration model (shown in the lower left panel) have a relatively large error. This is not surprising as the Lorenz-96 system is a misspecified model. The predictive means of GaSP and S-GaSP models which includes both the calibrated computer model and discrepancy function is more accurate, as both models capture the discrepancy between the reality and computer model. The estimation error of all models is larger than the ones when there is no discrepancy. This is because the measurement error is large and we only observe 2 states at each time point. Increasing the number of observations or reducing the variance of measurement error can improve predictive accuracy.

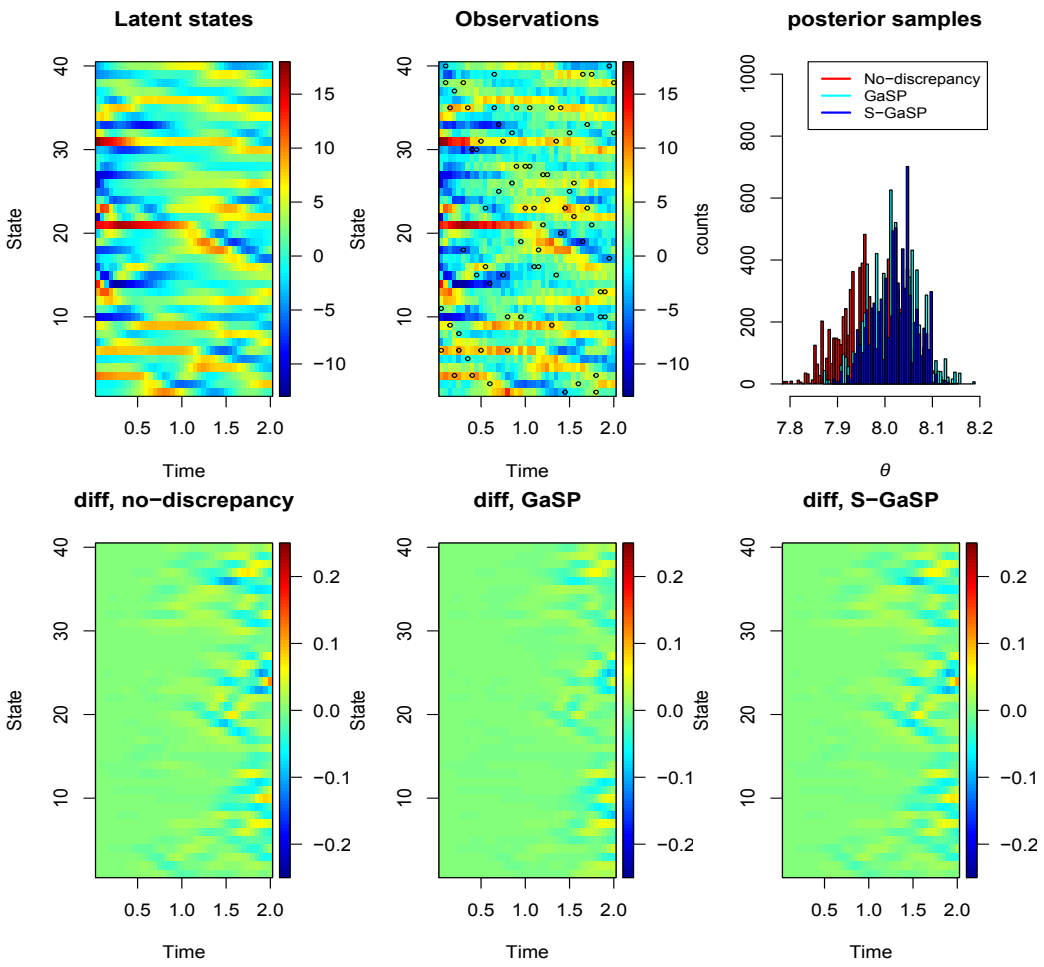

Figure 3: The reality and full observations of Lorenz-96 system in Scenario 1 are plotted in the upper left panel and upper middle panel, respectively, where the black circles are the 5% observations used for model calibration. Posterior samples of calibration parameters of different calibration methods are plotted in the upper right panel. The lower panels give the difference between the reality and calibrated computer model of all latent states.

3.4 Calibration with repeated experiments

Repeated experiments or replicates are commonly used in experiments, as they are helpful for assessing the experimental error. Consider, for example, k_i replicates available for each observable input $\mathbf{x}_i$, denoted as $\mathbf{y}_i^F = (y_1^F(\mathbf{x}_i), ..., y_{k_i}^F(\mathbf{x}_i))^T$. Denote $\bar{\mathbf{y}}^F = (\sum_{j=1}^{k_1} y_j^F(\mathbf{x}_1)/k_1, ..., \sum_{j=1}^{k_n} y_j^F(\mathbf{x}_n)/k_n)^T$ the aggregated data. The probability density of the field observations can be written as:

$$p(\mathbf{y}_1^F, ..., \mathbf{y}_n^F \mid \boldsymbol{\mu}, \boldsymbol{\delta}, \boldsymbol{\theta}, \sigma_0^2) = (2\pi\sigma_0^2)^{-\frac{\sum_{i=1}^n k_i}{2}} \exp\left(-\frac{\sum_{i=1}^n \omega_i \sum_{j=1}^{k_i}(y_j^F(\mathbf{x}_i) - f^M(\mathbf{x}_i, \boldsymbol{\theta}) - \mu_i - \delta(\mathbf{x}_i))^2}{2\sigma_0^2}\right)$$
$$= (2\pi\sigma_0^2)^{-\frac{\sum_{i=1}^n k_i}{2}} \exp\left(-\frac{S_f^2}{2\sigma_0^2} - \frac{\sum_{i=1}^n k_i\omega_i(\bar{y}_i^F - f^M(\mathbf{x}_i, \boldsymbol{\theta}) - \mu_i - \delta(\mathbf{x}_i))^2}{2\sigma_0^2}\right)$$

where $S_f^2 = \sum_{i=1}^n \omega_i \sum_{j=1}^{k_i}(y_j^F(\mathbf{x}_i) - \bar{y}_i^F)^2 = \sum_{i=1}^n \omega_i(\mathbf{y}_i^F - \bar{y}_i^F\mathbf{1}_{k_i})^T(\mathbf{y}_i^F - \bar{y}_i^F\mathbf{1}_{k_i})$ with the subscript 'f' denoting the field observations. After integrating out the discrepancy term, assumed to be modeled as a GaSP as an example, the marginal likelihood of the parameters follows

$$\mathcal{L}_K(\boldsymbol{\theta}, \boldsymbol{\theta}_m, \sigma_0^2, \boldsymbol{\gamma}, \eta) \propto (\sigma_0^2)^{-\frac{\sum_{i=1}^n k_i}{2}} |\tilde{\tilde{\boldsymbol{R}}}|^{-\frac{1}{2}} \exp\left\{-\frac{(\bar{\mathbf{y}}^F - \mathbf{f}_\theta^M - \mu\mathbf{1}_n)^T\tilde{\tilde{\boldsymbol{R}}}^{-1}(\bar{\mathbf{y}}^F - \mathbf{f}_\theta^M - \mu\mathbf{1}_n)}{2\sigma_0^2} - \frac{S_f^2}{2\sigma_0^2}\right\}, \quad (16)$$

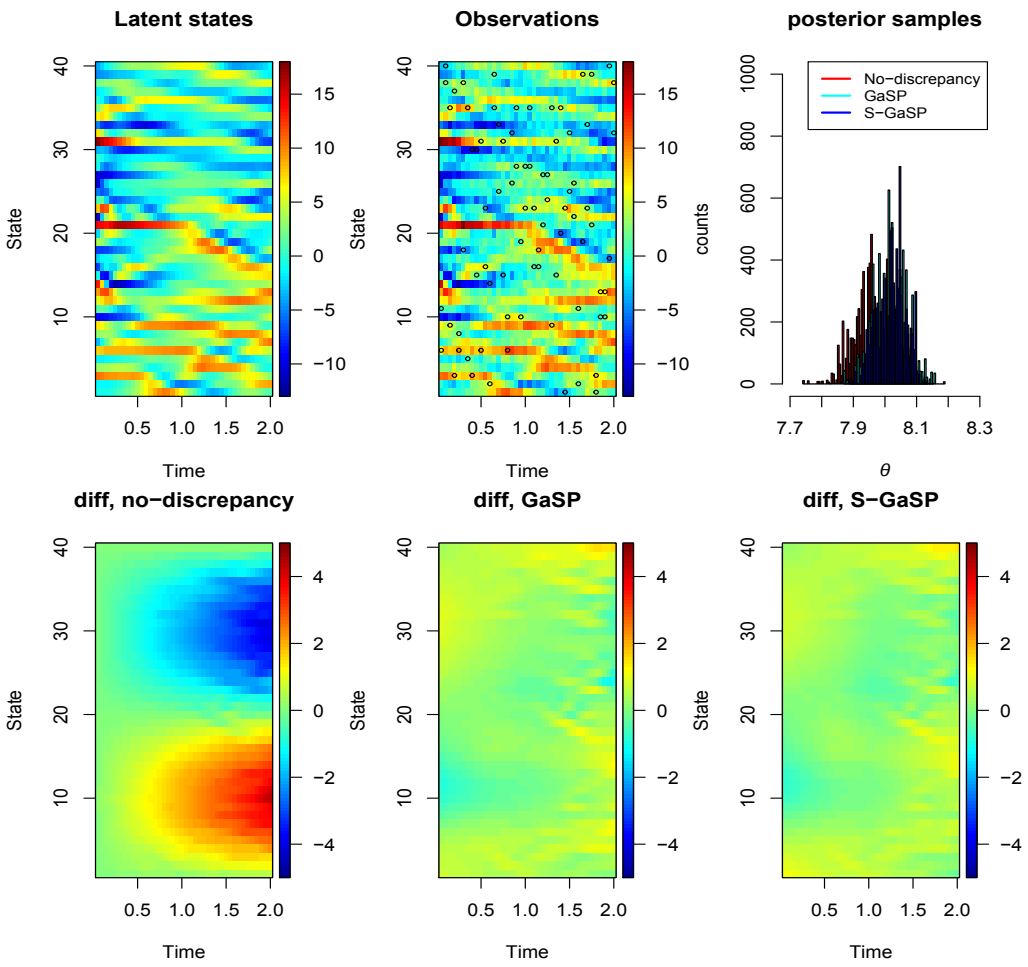

Figure 4: The reality of the discrepancy-included Lorenz-96 system (Scenario 2) is plotted in the upper left panel. The simulated observations are plotted in the upper middle panel, where the 5% of the observations plotted as black circles are used in model calibration. Posterior samples of calibration parameters of different calibration methods are plotted in the upper right panel. The lower panels show the differences between the reality and the predictive mean of all latent states.

where $\tilde{\tilde{\mathbf{R}}} = (\mathbf{R}/\eta + \tilde{\mathbf{\Lambda}})$, with $\tilde{\mathbf{\Lambda}}$ being an diagonal matrix with diagonal entry $\Lambda_{ii} = 1/(\omega_i k_i)$, for $i = 1, 2, ..., n$, and $\eta = \sigma_0^2/\sigma^2$. The S-GaSP calibration with replications can also be defined, by simply replacing $\mathbf{R}$ by $\mathbf{R}_z$ where the (i, j)th term is defined by the scaled kernel in Table 1.

The likelihood in (16) has a clear computational advantage. If one directly computes the covariance and its inverse, the computational complexity is $O((\sum_{i=1}^{n} k_i)^3)$, whereas the computational order by Equation (16) is only $O(n^3) + O(\sum_{i=1}^{n} k_i)$. Suppose $k_i = k$, for $i = 1, ..., n$, using Equation (16) is around k^3 times faster than directly computing the likelihood, with no loss of information.

One can specify replications by inputting a $n \times k$ matrix into the argument `observations` in the `rcalibration` function for scenarios with n observable inputs, each containing k repeated measurements. Or one can give a list of `observations` in the `rcalibration` function with size n, where each element in the list contains k_i values of replicates, for $i = 1, ..., n$. For observations with replicates, MLE and posterior sampling were implemented in the `rcalibration` function, and predictions were implemented in the `predict.rcalibration` function.

3.5 Statistical emulators

Users can specify a mathematical model via the `math_model` argument in `rcalibration` function if the closed-form expression of the mathematical model is available. However, closed-form solutions of physical systems are typically unavailable, and numerical solvers are required. Simulating the outputs from computer models can be slow. For instance, the TITAN2D computer model, which simulates pyroclastic flows for geological hazard quantification, takes roughly 8 minutes per set

of inputs (Simakov et al., 2019). As hundreds of thousands of model runs are sometimes required for model calibration (Anderson et al., 2019), directly computing the computer model by numerical methods is often prohibitive. In this scenario, we construct a statistical emulator as a surrogate model, to approximate the computer simulation, based on a small number of computer model runs.

To start, assume that we have obtained the output of the computer model at D input design points, $(\boldsymbol{\theta}_1, ..., \boldsymbol{\theta}_D)$, often selected from an "space-filling" design, such as the Latin hypercube design (Santner et al., 2003). There are two types of computer model outputs: 1) scalar-valued outputs: $f^M(\boldsymbol{\theta}) \in \mathbb{R}$, and 2) vector-valued outputs at k coordinates $\mathbf{f}^M(\mathbf{x}_{1:k}, \boldsymbol{\theta}) \in \mathbb{R}^k$, where k can be as large as 10^6. Various packages are available for fitting scalar-valued GP emulators, such as **DiceKriging** (Roustant et al. (2012)), **GPfit** (MacDonald et al. (2015)), and **RobustGaSP** (Gu et al. (2019)). Some of these packages were used in Bayesian model calibration packages (Palomo et al., 2015; Carmassi et al., 2018). However, the emulator of computer models with vector-valued outputs, a common scenario in different disciplines (Higdon et al., 2008; Ma et al., 2022; Li et al., 2022; Fang et al., 2022) was rarely implemented in the model calibration packages.

In **Robustcalibration**, we call the `rgasp` function and `ppgasp` function from the **RobustGaSP** package for emulating scalar-valued and vector-valued computer model outputs, respectively. The parallel partial Gaussian process (PP-GaSP) by the `ppgasp` function has two advantages. First, computing the predictive mean by the PP-GaSP only takes $O(kD) + O(D^3)$ operations, which is particularly suitable for computer models with a large number of outputs (k). Second, as the covariance over $\boldsymbol{\theta}$ is shared across all grids, the estimation of PP-GaSP is more stable than building a separate emulator on each output coordinate. Furthermore, the marginal posterior mode with robust parameterization typically avoids the degenerated estimation of the covariance matrix (Gu (2019)).

To call the emulator, users can select the argument `simulator=0`, and then specify the simulation runs in arguments `input_simul`, and `output_simul`. One can input a vector into `output_simul` to emulate scalar-valued computer models, or a matrix into `output_simul` to call the PP-GaSP emulator for vector-valued outputs. For both scenarios, we use a modular approach to fit the PP-GaSP emulator (Bayarri et al., 2007b; Liu et al., 2009), where the predictive distribution of the emulator depends on simulator runs, but not on field observations. After fitting an emulator, the predictive mean of the PP-GaSP emulator will be used to approximate the computer model at any unobserved $\boldsymbol{\theta}^*$.

We illustrate the efficiency and accuracy of the PP-GaSP emulator for approximating differential equations from Box and Coutie (1956), where the interaction between two chemical substances y_1 and y_2 are modeled as

$$\dot{y}_1(t) = 10^{\theta_1 - 3} y_1(t),$$
$$\dot{y}_2(t) = 10^{\theta_1 - 3} y_1(t) - 10^{\theta_2 - 3} y_2(t).$$

In each of the 6 time points $t = 10, 20, 40, 80, 160$, and 320, 2 replicates of the second chemical substance are available in Box and Coutie (1956). Initial conditions are $y_1(t = 0) = 100$ and $y_2(t = 0) = 0$. The computer model contains two parameters $\theta_1 \in [0.5, 1.5]$ and $\theta_2 \in [0.5, 1.5]$.

To start, we first use the default method in the `ode` function from the **deSolve** package (Soetaert et al., 2010) to numerically solve the ODEs at a given initial condition and parameter set:

```
library(deSolve)
R> Box_model <- function(time, state, parameters) {
+   par <- as.list(c(state, parameters))
+   with(par, {
+     dM1=-10^{parameters[1]-3}*M1
+     dM2=10^{parameters[1]-3}*M1-10^{parameters[2]-3}*M2
+     list(c(dM1, dM2))
+   })
+ }
R> Box_model_solved<-function(input, theta){
+   init <- c(M1 = 100 , M2 = 0)
+   out <- ode(y = init, times = c(0,input), func = Box_model, parms = theta)
+   return(out[-1,3])
+ }
```

We specify the observations and range of calibration parameters from Box and Coutie (1956) below.

```
R> n=6
R> output=t(matrix(c(19.2,14,14.4,24,42.3,30.8,42.1,40.5,40.7,46.4,27.1,22.3),2,n))
R> input=c(10,20,40,80,160,320)
R> theta_range=matrix(c(0.5,0.5,1.5,1.5),2,2)
###the testing input for emulator should contain the observed inputs
```

```
R> testing_input=as.matrix(seq(1,350,1))
#if observed inputs are not included, then add it
R> set_diff_obs=setdiff(input,testing_input)
R> testing_input=sort(c(testing_input,set_diff_obs))
```

We compare model calibration and predictions based on the numerical solver by the **deSolve** package, and by the GaSP emulator approach below. We first implement the direct approach, where the numerical solver is called to generate posterior samples.

```
R> m_sgasp_time=system.time({
+ m_sgasp=rcalibration(input,output,math_model=Box_model_solved,
+                     sd_proposal=c(0.25,0.25,1,1),
+                     theta_range=theta_range)
+ m_sgasp_pred=predict(m_sgasp,testing_input,math_model=Box_model_solved,
+                           interval_est=c(0.025,0.975))})
```

We then generate 50 design points from maximin Latin hypercube design by the **lhs** package, and run a numerical solver at these design points. As the simulator outputs a vector $\mathbf{f}^M(t_{1:k}, \boldsymbol{\theta}) \in \mathbb{R}^k$ for calibration parameter, we call the PP-GaSP emulator to predict the output at all time points. The simulator runs are then specified as `input_simul` and `output_simul` in the `rcalibration` function. We let `simul_type=0` to call the emulator instead of the numerical solver to generate posterior samples. The `loc_index_emulator` below gives a subset of the output coordinates for the the PP-GaSP emulator to be predicted and by default, the function will predict all output coordinates.

```
R> m_sgasp_emulator_time=system.time({
+   ##constructing simulation data
+   D=50
+   p=2
+   lhs_sample=maximinLHS(n=D,k=p)
+   input_simul=matrix(NA,D,p)
+   input_simul[,1]=lhs_sample[,1]+0.5
+   input_simul[,2]=lhs_sample[,2]+0.5
+   k_simul=length(testing_input)
+
+   output_simul=matrix(NA,D,k_simul)
+   for(i_D in 1:D){
+    output_simul[i_D,]=Box_model_solved(c(testing_input), input_simul[i_D,])
+   }
+   ##create loc_index_emulator as a subset of the simulation output
+   loc_index_emulator=rep(NA,n)
+   for(i in 1:n){
+     loc_index_emulator[i]=which(testing_input==input[i])
+   }
+
+   ##emulator
+   m_sgasp_with_emulator=rcalibration(input,output,simul_type=0,
+                        input_simul=input_simul, output_simul=output_simul,simul_nug=T,
+                        loc_index_emulator=loc_index_emulator,
+                        sd_proposal=c(0.25,0.25,1,1),
+                        theta_range=theta_range)
+   m_sgasp_with_emulator_pred=predict(m_sgasp_with_emulator,testing_input)
+   })
R> m_sgasp_time[3]/m_sgasp_emulator_time[3]
elapsed
6.252316
```

The approach with an emulator only costs around 1/6 of the computational time in this example, even though the numerical solver of this simple ODE is fast. For computer models that take a few or hours to run, the emulator approach can substantially reduce the computational cost.

Figure 5 gives predictions and posterior samples based on a numerical solver and the PP-GaSP emulator. First, predictions from the calibrated computer model and the discrepancy function, and by the calibrated computer model itself are close to each other, implying that the discrepancy modeled by a S-GaSP is close to be zero. Second, predictions and posterior distributions from the numerical solver and the emulator are close to each other, meaning that the emulator approximates the solver well. Calibration with an emulator is preferred as it is faster than numerically solving ODEs each time when generating the posterior samples.

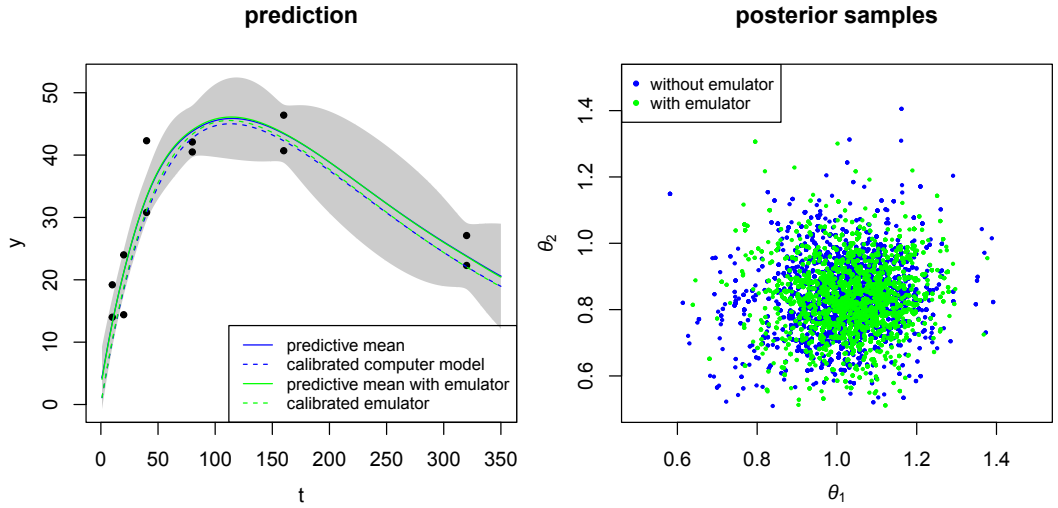

Figure 5: Comparison between the Bayesian model calibration based on the numerical solver and the PP-GaSP emulator of the computer model. In the left panel, the green solid curve is the predictive mean from the calibrated computer model and the discrepancy function, and the blue dashed curves are predictions from the calibrated computer model alone, both of which call the numerical solver for each posterior sample. The same result based on the emulator is plotted as the green curves. The black dots are field observations and the shaded area is the 95% predictive interval of the reality by the approach that uses the PP-GaSP emulator. In the right panel, posterior samples based on the numerical solver and the PP-GaSP emulator are denoted by the blue dots and green dots, respectively.

3.6 Calibration with multiple sources of observations

In reality, one may have observations of different types and they may come from multiple sources. In Anderson et al. (2019), for instance, multiple satellite interferogram and GPS observations measuring the ground deformation during the Kīlauea volcano eruption in Summer 2018 are used for calibrating mechanical models that relate the observations to calibration parameters, such as depth and shape of the magma chamber, magma density, pressure change rate, and so on. In these applications, the measurement bias, such as the atmospheric error, can substantially affect the satellite measurements of the ground deformation (Zebker et al., 1997; Agram and Simons, 2015).

Model calibration using multiple sources of observations was studied in Gu et al. (2023). For each source l, $l = 1, 2, ..., k$, let the field observations of the l th source at observable input $\mathbf{x}$ be modeled as

$$\mathbf{y}_l^F(\mathbf{x}) = f_l^M(\mathbf{x}, \boldsymbol{\theta}) + \delta(\mathbf{x}) + \delta_l(\mathbf{x}) + \mu_l + \epsilon_l, \tag{17}$$

where $f_l^M(\mathbf{x}, \boldsymbol{\theta})$ is the computer model for source l. The discrepancy function is denoted by $\delta(\mathbf{x})$ and the source-specific measurement bias is denoted by $\delta_l(\mathbf{x})$. Measurement bias often appears in the satellite radar interferogram, as atmospheric error could affect the quality of the image. The mean of each data source is denoted by μ_l, and the independent measurement noise is denoted by ϵ_l.

In **RobustCalibration**, we allow users to integrate observations from multiple sources to calibrate computer models by the function `rcalibration_MS` and to make predictions using the function `predict_MS.rcalibration_MS`. The posterior sampling method is implemented for parameter estimation for multiple sources of data. Besides, both GaSP and S-GaSP models can be chosen to model the measurement bias δ_l, for $l = 1, ..., k$. Similar to calibration model with a single source of data, we allow users to choose a model with no-discrepancy, and with GaSP or S-GaSP model of the discrepancy function δ. Users can choose to have measurement bias or not. We also allow users to integrate different types of measurements for model calibration by using different designs of observable inputs.

To illustrate, we study a synthetic example for calibrating computer models using multiple sources of observations. We assume the lth source of data, $l = 1, ..., k$, is simulated below

$$y_l^F(x) = sin(\pi x) + \delta(x) + \delta_l(x) + \epsilon_l(x), \tag{18}$$

where $\delta(.)$ and $\delta(.)$ are independently simulated from Gaussian processes with covariance $\sigma^2 K(.,.)$ and $\sigma_l^2 K_l(\cdot,\cdot)$, and the independent noise follows $\epsilon_l(x) \overset{i.i.d}{\sim} N(0, \sigma_0^2)$. We let $\sigma_0 = 0.05$, $\sigma = 0.2$ and $\sigma_l^2 = 0.5 \times l/(l-1)$, for $l = 1, ..., k$. The $K(.,.)$ and $K_l(\cdot,\cdot)$ are assumed to follow Matérn kernel with

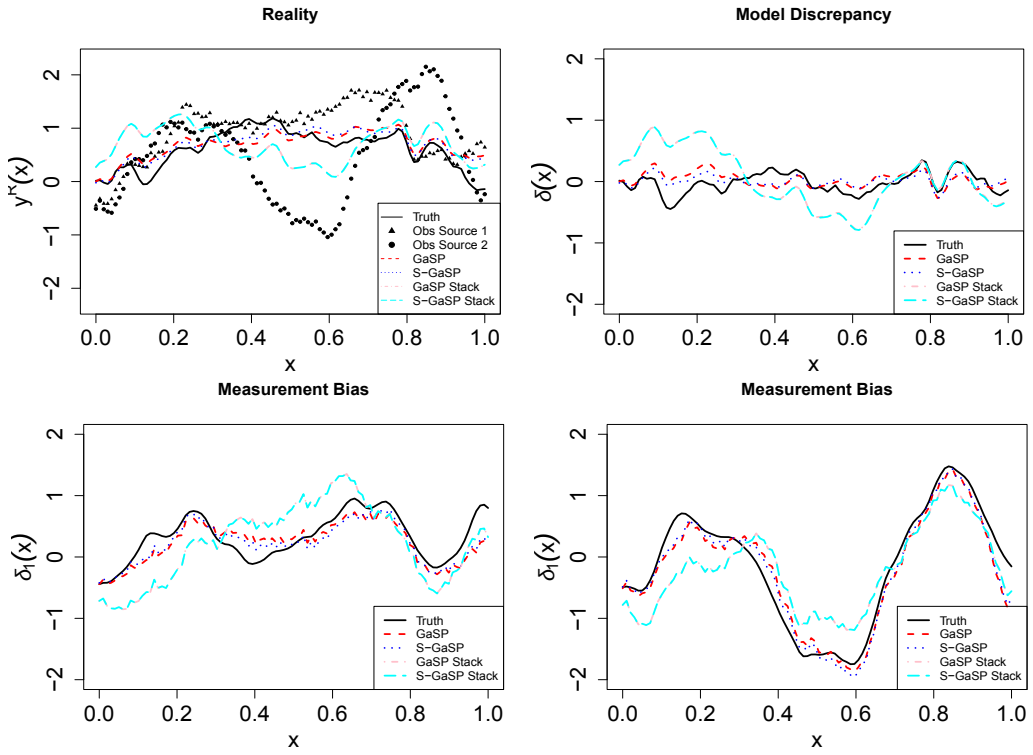

Figure 6: Comparison between modeling individual data and aggregated data for simulation from Equation (18). The upper left and right panels give the reality and model discrepancy, whereas the lower two panels give the measurement bias for the first two sources. The estimation results from GaSP, S-GaSP, GaSP Stack and S-GaSP calibration are plotted as red, blue, pink and cyan curves. The observations from the first two sources are graphed as the triangles and dots in the upper left panel.

roughness parameter being $\alpha = 2.5$, and the range parameter being $\gamma = 1/30$ and $\gamma = 1/10$, respectively. We collect $n = 100$ observations for each source l with x equally spaced from $[0, 1]$. We assume data from $k = 5$ sources are available. Here the mathematical model follows $f_l^M(x, \theta) = sin(\theta x)$, for $l = 1, ..., k$, and the reality is $y^R(x) = f_l^M(x, \theta) + \delta(x)$. The goal is to estimate the calibration parameter (θ), the reality $(y^R(x) = f_l^M(x, \theta) + \delta(x))$, model discrepancy $(\delta(x))$, and measurement bias $(\delta_l(x))$.

We compare four methods. The first two methods are GaSP and S-GaSP calibration methods to model individual sources of data, with the discrepancy model specified as a GaSP and S-GaSP respectively. The measurement bias terms are modeled by GaSPs. For instance, the code below implements S-GaSP model of discrepancy and GaSP model of the measurement bias:

```
> model_sgasp=rcalibration_MS(design=input_measurement,observations=output,p_theta=1,
              math_model=math_model, simul_type=rep(1,length(input_measurement)),
              S=5000,S_0=2000,thinning=1,measurement_bias=T,shared_design=input_model,
              have_measurement_bias_recorded=T,
              discrepancy_type=c(rep('GaSP',length(input_measurement)),'S-GaSP'),
              theta_range=matrix(c(-2*pi,2*pi),1,2),sd_proposal_theta=rep(0.02,2));
```

where `math_model`, `input_measurement` and `output` are lists where each contains the model, input and output of the field observations for each source. The `input_model` is a matrix for the input of the discrepancy function.

As modeling each source of the data can be time-consuming for some applications, a common solution is to use the aggregated or stacked data $\bar{y}^R(x) = \sum_{l=1}^k y_l^R(x)/k$ in calibration, as modeling the aggregated data is around k times faster than modeling the individual sources of data. Thus, we also include two methods of modeling the aggregated data for comparison, namely the GaSP Stack and S-GaSP Stack, representing GaSP or S-GaSP discrepancy model, respectively.

In Figure 6, we plot the truth, observations of the first two sources of observations, and the estimates of reality, discrepancy, and measurement bias in two sources from four different methods. Here the GaSP and S-GaSP perform better than the GaSP Stack and S-GaSP Stack calibration methods, as aggregating data leads to loss of information when the data contain measurement bias. The RMSE of reality, discrepancy and measurement bias estimation of different methods are given in the Table 3,

RMSE	reality	discrepancy	measurement bias
GaSP	0.238	0.194	0.247
S-GaSP	0.204	0.159	0.214
GaSP Stack	0.501	0.483	0.504
S-GaSP Stack	0.502	0.484	0.504

Table 3: The RMSE of the reality, discrepancy and measurement bias, where the smaller number indicates a smaller error. For measurement bias, we average the RMSE for each source of data.

which shows the S-GaSP calibration of individual sources of data achieves the highest calibration and predictive accuracy.

Finally, the mean of posterior samples θ from the GaSP, S-GaSP, GaSP Stack and S-GaSP Stack calibration for calibration parameter θ are 2.46, 2.73, 2.15, and 2.17 respectively, whereas the truth $\theta = \pi \approx 3.14$. This is a challenging scenario, as the large measurement bias makes estimation hard. We found model calibration by modeling individual sources of data is more accurate than modeling the aggregated data, whereas modeling the aggregated data is more computationally scalable.

4 Concluding remarks

We have introduced the **RobustCalibration** package for Bayesian model calibration and data inversion. This package has implemented a range of estimation methods and models, such as posterior sampling and MLE, for no-discrepancy calibration, GaSP and S-GaSP models of the discrepancy function. We implement statistical emulators for approximating computationally expensive computer models with both scalar-valued output or vector-valued output. The package is applicable to observations from a single source, with or without replications, and to observations from multiple sources or having different types. We illustrate our approaches using mathematical models with closed-form expressions or written as differential equations.

Even though we tried our best to consider different possible scenarios, a comprehensive statistical package for model calibration and data inversion is an ambitious goal. The **RobustCalibration** package provides tools for researchers to perform Bayesian model inversion without the need to write emulators or MCMC samplers themselves. We plan to improve the **RobustCalibration** package with several specific directions in mind. First, we will make the package more computationally scalable for structured data, such as imaging and time series observations. Furthermore, we plan allow users to specify other prior distributions of the parameters and proposal distributions to sample from in future versions of the package. Third, for problems like the Lorzen-96 system, it may be important to emulate the time evolution operator, if forecasting future states is the goal.

5 Appendix

5.1 Auxiliary facts

1. Let $\tilde{\mathbf{R}}_{\gamma,\eta} = \mathbf{R}_\gamma/\eta + \mathbf{\Lambda}$ be an $n \times n$ positive definite matrix as a function of parameters γ and η. Then for $l = 1, ..., p_x$:

$$\frac{\partial \log|\tilde{\mathbf{R}}_{\gamma,\eta}|}{\partial \gamma_l} = \operatorname{tr}\left(\frac{\tilde{\mathbf{R}}_{\gamma,\eta}^{-1}}{\eta}\frac{\partial \mathbf{R}_\gamma}{\partial \gamma_l}\right) \quad \text{and} \quad \frac{\partial \log|\tilde{\mathbf{R}}_{\gamma,\eta}|}{\partial \eta} = -\operatorname{tr}\left(\frac{\tilde{\mathbf{R}}_{\gamma,\eta}^{-1}\mathbf{R}_\gamma}{\eta^2}\right).$$

2. Let $\mathbf{H}$ be an $n \times q$ full rank matrix with $q < n$, $\mathbf{Q} = \left(\tilde{\mathbf{R}}_{\gamma,\eta}^{-1} - \tilde{\mathbf{R}}_{\gamma,\eta}^{-1}\mathbf{H}^T(\mathbf{H}^T\tilde{\mathbf{R}}_{\gamma,\eta}^{-1}\mathbf{H})^{-1}\mathbf{H}^T\tilde{\mathbf{R}}_{\gamma,\eta}^{-1}\right)$ with $\tilde{\mathbf{R}}_{\gamma,\eta} = \mathbf{R}_\gamma/\eta + \mathbf{\Lambda}$ and $\mathbf{y}$ is an $n \times 1$ vector. Then for $l = 1, ..., p_x$:

$$\frac{\partial \log(\mathbf{y}^T\mathbf{Q}_{\gamma,\eta}\mathbf{y})}{\partial \gamma_l} = -\frac{\mathbf{y}^T\mathbf{Q}_{\gamma,\eta}\frac{\partial \mathbf{R}_\gamma}{\partial \gamma_l}\mathbf{Q}_{\gamma,\eta}\mathbf{y}}{\eta\mathbf{y}^T\mathbf{Q}_{\gamma,\eta}\mathbf{y}} \quad \text{and} \quad \frac{\partial \log(\mathbf{y}^T\mathbf{Q}_{\gamma,\eta}\mathbf{y})}{\partial \eta} = \frac{\mathbf{y}^T\mathbf{Q}_{\gamma,\eta}\mathbf{R}_\gamma\mathbf{Q}_{\gamma,\eta}\mathbf{y}}{\eta^2\mathbf{y}^T\mathbf{Q}_{\gamma,\eta}\mathbf{y}}.$$

3. Suppose $v_{\boldsymbol{\theta}}$ is an $n \times 1$ vector as a function of $\boldsymbol{\theta}$ and $\tilde{\mathbf{R}}$ is an $n \times n$ positive definite matrix. Then for $i = 1, ..., p_\theta$:

$$\frac{\partial \log(\mathbf{v}_{\boldsymbol{\theta}}^T\mathbf{R}^{-1}\mathbf{v}_{\boldsymbol{\theta}})}{\partial \theta_l} = \frac{2\mathbf{v}_\theta^T\tilde{\mathbf{R}}^{-1}\frac{\partial \mathbf{v}_\theta}{\partial \theta_l}}{\mathbf{v}_\theta^T\tilde{\mathbf{R}}^{-1}\mathbf{v}_\theta}.$$

5.2 Likelihood functions and posterior distributions

Posterior distributions of the no-discrepancy calibration. Using the objective prior for the mean and variance parameters, $\pi(\boldsymbol{\theta}_m, \sigma_0^2) \propto 1/\sigma_0^2$, the full conditional posterior distributions of the trend and variance parameters follow:

$$\sigma_0^{-2} \mid \boldsymbol{\theta}_m, \boldsymbol{\theta} \sim \text{Gamma}\left(\frac{n}{2}, \frac{(\mathbf{y}^F - \mathbf{f}_\theta^M - \mathbf{H}\boldsymbol{\theta}_m)^T \boldsymbol{\Lambda}^{-1} (\mathbf{y}^F - \mathbf{f}_\theta^M - \mathbf{H}\boldsymbol{\theta}_m)}{2}\right),$$
$$\boldsymbol{\theta}_m \mid \sigma_0^2, \boldsymbol{\theta} \sim \mathcal{N}\left(\hat{\boldsymbol{\theta}}_m^{LS}, \sigma_0^{-2}(\mathbf{H}^T \boldsymbol{\Lambda}^{-1} \mathbf{H})^{-1}\right).$$

The posterior distribution of the calibration parameters follows:

$$p(\boldsymbol{\theta} \mid \boldsymbol{\theta}_m, \sigma_0^2, \boldsymbol{\beta}, \eta) \propto \exp\left(-\frac{(\mathbf{y}^F - \mathbf{f}_\theta^M - \mathbf{H}\boldsymbol{\theta}_m)^T \boldsymbol{\Lambda}^{-1} (\mathbf{y}^F - \mathbf{f}_\theta^M - \mathbf{H}\boldsymbol{\theta}_m)}{2\sigma_0^2}\right) \pi(\boldsymbol{\theta}),$$

where $\pi(\boldsymbol{\theta})$ is the prior of calibration parameters, where the default prior distribution is the uniform distribution $\pi(\boldsymbol{\theta}) \propto 1$. We use the Metropolis algorithm to sample the calibration parameters as a block, as we only need to compute the likelihood once in each iteration. The proposal distribution of each coordinate of the calibration parameter vector is chosen as a normal distribution with a pre-specified standard deviation, proportional to the range of the parameters. The default proportion is 0.05, and users can adjust the proportion by changing the first p_θ value in the argument `sd_proposal` in `rcalibration` function.

After obtaining S posterior samples with the first S_0 burn-in samples, the predictive mean of reality based on calibrated computer model and trend by the no-discrepancy calibration can be computed by:

$$\hat{\mu}(\mathbf{x}^*) = \frac{1}{S - S_0} \sum_{i=S_0+1}^{S} f^M(\mathbf{x}^*, \boldsymbol{\theta}^{(i)}) + \mathbf{h}(\mathbf{x}^*)\boldsymbol{\theta}_m^{(i)}.$$

The posterior credible interval of the reality $y^R(\mathbf{x}^*)$ in the no-discrepancy calibration can be obtained by the empirical quantile of posterior samples $f^M(\mathbf{x}^*, \boldsymbol{\theta}^{(i)}) + \mathbf{h}(\mathbf{x}^*)\boldsymbol{\theta}_m^{(i)}$. For example, the posterior credible interval of $1 - \alpha$ percentile of the data in no-discrepancy calibration can be computed by based on $\frac{1}{S-S_0}\sum_{i=S_0+1}^{S}\left(f^M(\mathbf{x}^*, \boldsymbol{\theta}^{(i)}) + \mathbf{h}(\mathbf{x}^*)\boldsymbol{\theta}_m^{(i)} + \sqrt{\frac{\sigma_0^{(i)}}{w^*}} Z_\alpha\right)$, where Z_α is an upper α quantile of the standard normal distribution and w^* is the relative test output weight, assumed to be 1 by default.

Likelihood functions and derivatives in GaSP calibration. The profile likelihood function in GaSP calibration follows

$$\mathcal{L}_K(\boldsymbol{\theta}, \gamma, \eta \mid \hat{\boldsymbol{\theta}}_m, \hat{\sigma}_0^2) \propto |\tilde{\mathbf{R}}|^{-\frac{1}{2}} (S_K^2)^{-\frac{n}{2}}, \tag{19}$$

where $S_K^2 = (\mathbf{y} - \mathbf{f}_\theta^M - \mathbf{H}^T\hat{\boldsymbol{\theta}}_m)^T \tilde{\mathbf{R}}^{-1} (\mathbf{y} - \mathbf{f}_\theta^M - \mathbf{H}^T\hat{\boldsymbol{\theta}}_m)$.

Denote $\mathbf{v}_{\boldsymbol{\theta}_m} = (\mathbf{y} - \mathbf{f}_\theta^M - \mathbf{H}^T\hat{\boldsymbol{\theta}}_m)$. Similar to the no-discrepancy calibration, we call the `lbfgs` function in the **nloptr** package (Ypma (2014)) for optimization. By facts 1 and 3, directly differentiating the profile likelihood function with respect to kernel and nugget parameters in (19) gives:

$$\frac{\partial \log(\mathcal{L}_K(\boldsymbol{\theta}, \gamma, \eta \mid \hat{\theta}_m, \hat{\sigma}^2))}{\partial \gamma_l} = -\frac{1}{2}\text{tr}\left(\frac{\tilde{\mathbf{R}}_{\gamma,\eta}^{-1}}{\eta}\frac{\partial \mathbf{R}_\gamma}{\partial \gamma_l}\right) + \frac{n}{2} \times \frac{\mathbf{v}_{\boldsymbol{\theta}_m}^T \mathbf{Q}_{\gamma,\eta} \frac{\partial \mathbf{R}_\gamma}{\partial \gamma_l} \mathbf{Q}_{\gamma,\eta} \mathbf{v}_{\boldsymbol{\theta}_m}}{\eta \mathbf{v}_{\boldsymbol{\theta}_m}^T \mathbf{Q}_{\gamma,\eta} \mathbf{v}_{\boldsymbol{\theta}_m}}$$
$$\frac{\partial \log(\mathcal{L}_K(\boldsymbol{\theta}, \gamma, \eta \mid \hat{\theta}_m, \hat{\sigma}^2))}{\partial \eta} = \frac{1}{2}\text{tr}\left(\frac{\tilde{\mathbf{R}}_{\gamma,\eta}^{-1}\mathbf{R}_\gamma}{\eta^2}\right) + \frac{n}{2} \times \frac{\mathbf{v}_{\boldsymbol{\theta}_m}^T \mathbf{Q}_{\gamma,\eta} \mathbf{R}_\gamma \mathbf{Q}_{\gamma,\eta} \mathbf{v}_{\boldsymbol{\theta}_m}}{\eta^2 \mathbf{v}_{\boldsymbol{\theta}_m}^T \mathbf{Q}_{\gamma,\eta} \mathbf{v}_{\boldsymbol{\theta}_m}}$$
$$\frac{\partial \log(\mathcal{L}_K(\boldsymbol{\theta}, \gamma, \eta \mid \hat{\theta}_m, \hat{\sigma}^2))}{\partial \theta_l} = \frac{n}{2} \times \frac{\mathbf{v}_{\boldsymbol{\theta}_m}^T \mathbf{Q}_{\gamma,\eta} \frac{\partial \mathbf{f}_\theta^M}{\partial \theta_l}}{\mathbf{v}_{\boldsymbol{\theta}_m}^T \mathbf{Q}_{\gamma,\eta} \mathbf{v}_{\boldsymbol{\theta}_m}}$$

for $l = 1, ..., p_x$ and $i = 1, ..., p_\theta$. When the mean function is zero, one can simply replace $\mathbf{Q}_{\gamma,\eta}$ by $\tilde{\boldsymbol{R}}_{\gamma,\eta}$ in the formula above to obtain the derivatives. Besides, when $\frac{\partial \mathbf{f}_\theta^M}{\partial \theta_l}$ is not available, we use the numerical derivatives to approximate.

Posterior distributions in GaSP calibration. After marginalizing out the discrepancy function, the marginal distribution follows $\mathbf{y}^F \sim \mathcal{MN}(\mathbf{f}_\theta^M + \mathbf{H}\boldsymbol{\theta}_m, \sigma_0^2\tilde{\mathbf{R}})$, where $\tilde{\mathbf{R}} = \mathbf{R}/\eta + \boldsymbol{\Lambda}$. We assume an objective prior for the mean and noise variance parameters $\pi(\boldsymbol{\theta}_m, \sigma_0^2) \propto 1/\sigma_0^2$. The full conditional

posterior distributions of the trend and variance parameters follow

$$\sigma_0^{-2} \mid \boldsymbol{\theta}_m, \boldsymbol{\theta} \sim \text{Gamma}\left(\frac{n}{2}, \frac{(\mathbf{y}^F - \mathbf{f}_\theta^M - \mathbf{H}\boldsymbol{\theta}_m)^T \tilde{\mathbf{R}}^{-1}(\mathbf{y}^F - \mathbf{f}_\theta^M - \mathbf{H}\boldsymbol{\theta}_m)}{2}\right),$$
$$\boldsymbol{\theta}_m \mid \sigma_0^2, \boldsymbol{\theta} \sim \mathcal{MN}\left(\hat{\boldsymbol{\theta}}_m, \sigma_0^{-2}(\mathbf{H}^T\tilde{\mathbf{R}}^{-1}\mathbf{H})^{-1}\right),$$

where $\hat{\boldsymbol{\theta}}_m = \left(\mathbf{H}^T\tilde{\mathbf{R}}^{-1}\mathbf{H}\right)^{-1}\mathbf{H}^T\tilde{\mathbf{R}}^{-1}(\mathbf{y} - \mathbf{f}_\theta^M)$. A Gibbs sampler is used to sample σ_0^{-2} and $\boldsymbol{\theta}_m$ from the full conditional distribution.

Given the trend and variance parameters, the posterior distribution of the calibration parameters follows

$$p(\boldsymbol{\theta} \mid \boldsymbol{\theta}_m, \sigma^2) \propto \exp\left(-\frac{(\mathbf{y}^F - \mathbf{f}_\theta^M - \mathbf{H}\boldsymbol{\theta}_m)^T\tilde{\mathbf{R}}^{-1}(\mathbf{y}^F - \mathbf{f}_\theta^M - \mathbf{H}\boldsymbol{\theta}_m)}{2\sigma_0^2}\right)\pi(\boldsymbol{\theta}).$$

We implement the Metropolis algorithm where the proposal distribution is a normal distribution with the mean centered around the current value of the posterior sample. Users can adjust the standard deviation of the proposal distribution by argument `sd_proposal` in the `rcalibration` function.

Denote the inverse range parameter $\beta_l = 1/\gamma_l$, $l = 1, ..., p_x$. The conditional distribution of the inverse range and nugget parameters follow

$$p(\boldsymbol{\beta}, \eta \mid \boldsymbol{\theta}, \boldsymbol{\theta}_m, \sigma_0^2) \propto |\tilde{\mathbf{R}}|^{-\frac{n}{2}} \exp\left(-\frac{(\mathbf{y}^F - \mathbf{f}_\theta^M - \mathbf{H}\boldsymbol{\theta}_m)^T\tilde{\mathbf{R}}^{-1}(\mathbf{y}^F - \mathbf{f}_\theta^M - \mathbf{H}\boldsymbol{\theta}_m)}{2\sigma_0^2}\right)\pi(\boldsymbol{\gamma}, \eta).$$

We define the inverse range parameter $\beta_l = 1/\gamma_l$, and assume that the prior for the inverse range and nugget parameters follows the jointly robust (JR) prior (Gu, 2019)

$$\pi(\boldsymbol{\beta}, \eta) = C\left(\sum_{l=1}^{p_x} C_l\beta_l + \eta\right)^a \exp\left\{-b\left(\sum_{l}^{p_x} C_l\beta_l + \eta\right)\right\},$$

where C is a normalizing constant. By default, we use the following prior parameters $a = 1/2 - p_x$, $b = 1$, and $C_l = n^{-1/p_x}|x_l^{max} - x_l^{min}|$, with x_l^{max} and x_l^{min} being the maximum and minimum values of the observable input at the lth coordinate, respectively. The default choices of prior parameters induces a small penalty on very large correlation, preventing the identifiability issues of the calibration parameters due to large correlation (Gu, 2019). Users can adjust a and b in the `rcalibration` function.

We use the Metropolis algorithm to sample the logarithm of the inverse range parameter $\log(\beta_l)$ and nugget parameter $\log(\eta)$. The proposal distribution of each parameter is a normal distribution centered around the current value with the standard deviation chosen to be 0.25 by default. User can change the $p_\theta + 1$ to $p_x + p_\theta + 1$ coordinates of the vector in the argument `sd_proposal` in the `rcalibration` function to specify the standard deviation in the proposal distribution for these parameters.

Likelihood function and posterior distributions S-GaSP calibration. The likelihood function and posterior distribution of S-GaSP follow from the those in GaSP calibration, by simply replacing the kernel K to K_{Z_d} in Table 1.

Acknowledgements

This research was supported by the National Science Foundation under Award Number DMS-2053423. We thank Xubo Liu for implementing the numerical method for simulating the Lorenz 96 model.

References

P. Agram and M. Simons. A noise model for InSAR time series. *Journal of Geophysical Research: Solid Earth*, 120(4):2752–2771, 2015. [p99]

K. R. Anderson, I. A. Johanson, M. R. Patrick, M. Gu, P. Segall, M. P. Poland, E. K. Montgomery-Brown, and A. Miklius. Magma reservoir failure and the onset of caldera collapse at Kīlauea volcano in 2018. *Science*, 366(6470), 2019. [p85, 86, 97, 99]

P. D. Arendt, D. W. Apley, and W. Chen. Quantification of model uncertainty: calibration, model discrepancy, and identifiability. *Journal of Mechanical Design*, 134(10):100908, 2012. [p84]

M. Bayarri, J. Berger, J. Cafeo, G. Garcia-Donato, F. Liu, J. Palomo, R. Parthasarathy, R. Paulo, J. Sacks, and D. Walsh. Computer model validation with functional output. *The Annals of Statistics*, 35(5): 1874–1906, 2007a. [p84]

M. J. Bayarri, J. O. Berger, R. Paulo, J. Sacks, J. A. Cafeo, J. Cavendish, C.-H. Lin, and J. Tu. A framework for validation of computer models. *Technometrics*, 49(2):138–154, 2007b. [p89, 91, 92, 94, 97]

G. Box and G. Coutie. Application of digital computers in the exploration of functional relationships. *Proceedings of the IEE-Part B: Radio and Electronic Engineering*, 103(1S):100–107, 1956. [p97]

J. Brajard, A. Carrassi, M. Bocquet, and L. Bertino. Combining data assimilation and machine learning to emulate a dynamical model from sparse and noisy observations: A case study with the Lorenz 96 model. *Journal of Computational Science*, 44:101171, 2020. [p94]

M. Carmassi, P. Barbillon, M. Chiodetti, M. Keller, and E. Parent. CaliCo: a R package for Bayesian calibration. *arXiv preprint arXiv:1808.01932*, 2018. [p85, 97]

W. Chang, M. Haran, P. Applegate, and D. Pollard. Calibrating an ice sheet model using high-dimensional binary spatial data. *Journal of the American Statistical Association*, 111(513):57–72, 2016. [p84]

W. Chang, B. A. Konomi, G. Karagiannis, Y. Guan, and M. Haran. Ice model calibration using semicontinuous spatial data. *The Annals of Applied Statistics*, 16(3):1937–1961, 2022. [p84]

X. Fang, M. Gu, and J. Wu. Reliable emulation of complex functionals by active learning with error control. *The Journal of Chemical Physics*, 157(21):214109, 2022. [p97]

M. Gu. Jointly robust prior for Gaussian stochastic process in emulation, calibration and variable selection. *Bayesian Analysis*, 14(1), 2019. [p92, 97, 103]

M. Gu and L. Wang. Scaled Gaussian stochastic process for computer model calibration and prediction. *SIAM/ASA Journal on Uncertainty Quantification*, 6(4):1555–1583, 2018. [p84, 92, 93]

M. Gu, J. Palomo, and J. O. Berger. RobustGaSP: Robust Gaussian Stochastic Process Emulation in R. *The R Journal*, 11(1):112–136, 2019. doi: 10.32614/RJ-2019-011. [p85, 97]

M. Gu, F. Xie, and L. Wang. A theoretical framework of the scaled Gaussian stochastic process in prediction and calibration. *SIAM/ASA Journal on Uncertainty Quantification*, 10(4):1435–1460, 2022. [p93]

M. Gu, K. Anderson, and E. McPhillips. Calibration of imperfect geophysical models by multiple satellite interferograms with measurement bias. *Technometrics, In Press*, 2023. doi: 10.1080/00401706.2023.2182365. [p99]

R. K. Hankin. Introducing BACCO, an R bundle for Bayesian analysis of computer code output. *Journal of Statistical Software*, 14:1–21, 2005. [p85]

D. Higdon, J. Gattiker, B. Williams, and M. Rightley. Computer model calibration using high-dimensional output. *Journal of the American Statistical Association*, 103(482):570–583, 2008. [p84, 97]

M. C. Kennedy and A. O'Hagan. Bayesian calibration of computer models. *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, 63(3):425–464, 2001. [p84, 85]

H. Li, M. Zhou, J. Sebastian, J. Wu, and M. Gu. Efficient force field and energy emulation through partition of permutationally equivalent atoms. *The Journal of Chemical Physics*, 156(18):184304, 2022. [p97]

D. C. Liu and J. Nocedal. On the limited memory BFGS method for large scale optimization. *Mathematical programming*, 45(1-3):503–528, 1989. [p88]

F. Liu, M. Bayarri, and J. Berger. Modularization in Bayesian analysis, with emphasis on analysis of computer models. *Bayesian Analysis*, 4(1):119–150, 2009. [p97]

E. N. Lorenz. Predictability: A problem partly solved. In *Proc. Seminar on predictability*, volume 1, 1996. [p94]

P. Ma, G. Karagiannis, B. A. Konomi, T. G. Asher, G. R. Toro, and A. T. Cox. Multifidelity computer model emulation with high-dimensional output: An application to storm surge. *Journal of the Royal Statistical Society Series C: Applied Statistics*, 71(4):861–883, 2022. [p85, 97]

B. MacDonald, P. Ranjan, H. Chipman, et al. Gpfit: An R package for fitting a Gaussian process model to deterministic simulator outputs. *Journal of Statistical Software*, 64(i12), 2015. [p85, 97]

J. Maclean and E. T. Spiller. A surrogate-based approach to nonlinear, non-Gaussian joint state-parameter data assimilation. *arXiv preprint arXiv:2012.04793*, 2020. [p94]

J. Palomo, R. Paulo, G. García-Donato, et al. Save: an R package for the statistical analysis of computer models. *Journal of Statistical Software*, 64(13):1–23, 2015. [p85, 97]

R. Paulo, G. García-Donato, and J. Palomo. Calibration of computer models with multivariate output. *Computational Statistics and Data Analysis*, 56(12):3959–3974, 2012. [p84]

M. Plumlee. Bayesian calibration of inexact computer models. *Journal of the American Statistical Association*, 112(519):1274–1285, 2017. [p85]

O. Roustant, D. Ginsbourger, and Y. Deville. Dicekriging, Diceoptim: Two R packages for the analysis of computer experiments by kriging-based metamodeling and optimization. *Journal of statistical software*, 51:1–55, 2012. [p85, 97]

J. Sacks, W. J. Welch, T. J. Mitchell, H. P. Wynn, et al. Design and analysis of computer experiments. *Statistical science*, 4(4):409–423, 1989. [p84]

T. J. Santner, B. J. Williams, and W. I. Notz. *The design and analysis of computer experiments*. Springer Science & Business Media, 2003. [p97]

N. A. Simakov, R. L. Jones-Ivey, A. Akhavan-Safaei, H. Aghakhani, M. D. Jones, and A. K. Patra. Modernizing Titan2D, a parallel AMR geophysical flow code to support multiple rheologies and extendability. In *International Conference on High Performance Computing*, pages 101–112. Springer, 2019. [p97]

K. Soetaert, T. Petzoldt, and R. W. Setzer. Solving differential equations in R: package deSolve. *Journal of statistical software*, 33:1–25, 2010. [p97]

R. Tuo and C. J. Wu. Efficient calibration for imperfect computer models. *The Annals of Statistics*, 43(6): 2331–2352, 2015. [p85]

H. Wickham. ggplot2. *Wiley Interdisciplinary Reviews: Computational Statistics*, 3(2):180–185, 2011. [p85]

D. Williamson, M. Goldstein, L. Allison, A. Blaker, P. Challenor, L. Jackson, and K. Yamazaki. History matching for exploring and reducing climate model parameter space using observations and a large perturbed physics ensemble. *Climate dynamics*, 41(7-8):1703–1729, 2013. [p85]

R. K. Wong, C. B. Storlie, and T. Lee. A frequentist approach to computer model calibration. *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, 79:635–648, 2017. [p85]

J. Ypma. *nloptr: R interface to NLopt*, 2014. URL https://CRAN.R-project.org/package=nloptr. R package version 1.0.4. [p88, 102]

H. A. Zebker, P. A. Rosen, and S. Hensley. Atmospheric effects in interferometric synthetic aperture radar surface deformation and topographic maps. *Journal of Geophysical Research: Solid Earth*, 102 (B4):7547–7563, apr 1997. ISSN 01480227. doi: 10.1029/96JB03804. URL http://doi.wiley.com/10.1029/96JB03804. [p99]

Mengyang Gu
University of California, Santa Barbara
Department of Statistics and Applied Probability
Santa Barbara, California, USA
mengyang@pstat.ucsb.edu