JOSS
The Journal of Open Source Software

CalibrateEmulateSample.jl: Accelerated Parametric Uncertainty Quantification

Oliver R. A. Dunbar [1¶], **Melanie Bieli**[2], **Alfredo Garbuno-Iñigo** [3], **Michael Howland** [4], **Andre De Souza**[5], **Laura Anne Mansfield** [6], **and Gregory L. Wagner** [5]

1 Geological and Planetary Sciences, California Institute of Technology **2** Swiss Re Ltd. **3** Department of Statistics, Mexico Autonomous Institute of Technology **4** Civil and Environmental Engineering, Massachusetts Institute of Technology **5** Earth, Atmospheric, and Planetary Sciences, Massachusetts Institute of Technology **6** Earth System Science, Doerr School of Sustainability, Stanford University ¶ Corresponding author

DOI: 10.xxxxxx/draft

Software

- Review
- Repository
- Archive

Editor: Open Journals

Reviewers:

- @openjournals

Submitted: 01 January 1970
Published: unpublished

Summary

A julia-language (Bezanson et al., 2017) package providing practical and modular implementation of "Calibrate, Emulate, Sample" (Cleary et al., 2021), hereafter CES, an accelerated workflow for obtaining model parametric uncertainty is presented. This is also known as Bayesian inversion or uncertainty quantification. To apply CES one requires a computer model (written in any programming language) dependent on free parameters, and some data with which to constrain the free parameter distribution. The pipeline has three stages, most easily explained in reverse: the last stage is to draw samples (Sample) from the Bayesian posterior distribution, i.e. the constrained joint parameter distribution consistent with observed data; to accelerate and smooth this process we train statistical machine-learning emulators to represent the user-provided parameter-to-data map (Emulate); the training points for these emulators are generated by the computer model, and selected adaptively around regions of high posterior mass (Calibrate). We describe CES as an accelerated workflow, as it uses dramatically fewer evaluations of the computer model when compared with traditional algorithms to draw samples from the joint parameter distribution.

- Calibration tools: We recommend choosing adaptive training points with Ensemble Kalman methods such as EKI (Iglesias et al., 2013) and its variants (Huang et al., 2022); and CES provides explicit utilities from the codebase EnsembleKalmanProcesses.jl (Dunbar, Lopez-Gomez, et al., 2022).
- Emulation tools: CES integrates any statistical emulator, currently implemented are Gaussian Processes (Williams & Rasmussen, 2006), explicitly provided through packages SciKitLearn.jl (Pedregosa et al., 2011) and GaussianProcesses.jl (Fairbrother et al., 2022), and Random Features (Liu et al., 2022; Rahimi et al., 2007; Rahimi & Recht, 2008), explicitly provided through RandomFeatures.jl that can provide additional flexibility and scalability, particularly in higher dimensions.
- Sampling tools: The smoothed accelerated sampling problem is solved with Markov Chain Monte Carlo, and CES provides the variants of Random Walk Metropolis (Sherlock et al., 2010), and preconditioned Crank-Nicholson (Cotter et al., 2013), using APIs from Turing.jl.

To highlight code accessibility, we also provide a suite of detailed scientifically-inspired examples, with documentation that walks users through some use cases. Such use cases not only demonstrate the capability of the CES pipeline, but also teach users about typical interface and workflow experience.

Statement of need

Computationally expensive computer codes for predictive modelling are ubiquitous across science and engineering disciplines. Free parameter values that exist within these modelling frameworks are typically constrained by observations to produce accurate and robust predictions about the system they are approximating numerically. In a Bayesian setting, this is viewed as evolving an initial parameter distribution (based on prior information) with the input of observed data, to a more informative data-consistent distribution (posterior). Unfortunately, this task is intensely computationally expensive, commonly requiring over 10^5 evaluations of the expensive computer code, with accelerations relying on intrusive model information, such as a derivative of the parameter-to-data map. CES is able to approximate and accelerate this process in a non-intrusive fashion and requiring only on the order of 10^2 evaluations of the code. This opens the doors for quantifying parametric uncertainty for a class of numerically intensive computer codes that classically this has been unavailable.

State of the field

In Julia there are a few tools for performing non-accelerated uncertainty quantification, from classical sensitivity analysis approaches, e.g., UncertaintyQuantification.jl, GlobalSensitivity.jl (Dixit & Rackauckas, 2022), and Bayesian Markov Chain Monte Carlo, e.g., Mamba.jl or Turing.jl. For computational efficiency, ensemble Methods also provide approximate sampling (e.g., the Ensemble Kalman Sampler (Dunbar, Lopez-Gomez, et al., 2022; Garbuno-Inigo et al., 2020)) though these only provide Gaussian approximations of the posterior.

Accelerated uncertainty quantification tools also exist for the related approach of Approximate Bayesian Computation (ABC), e.g., GpABC (Tankhilevich et al., 2020) or ApproxBayes.jl; these tools both approximately sample from the posterior distribution. In ABC, this approximation comes from bypassing the likelihood that is usually required in sampling methods, such as MCMC. Instead, the goal ABC is to replace the likelihood with a scalar-valued sampling objective that compares model and data. In CES, the approximation comes from learning the parameter-to-data map, then following this it calculates an explicit likelihood and uses exact sampling via MCMC. Some ABC algorithms also make use of statistical emulators to further accelerate sampling (gpABC). ABC can be used in more general contexts than CES, but suffers greater approximation error and more stringent assumptions, especially in multi-dimensional problems.

A simple example from the code documentation

We sketch an end-to-end example of the pipeline, with fully-detailed walkthrough given in the online documentation.

We have a model of a sinusoidal signal that is a function of parameters $\theta = (A, v)$, where A is the amplitude of the signal and v is vertical shift of the signal

$$f(A, v) = A\sin(\phi + t) + v, \forall t \in [0, 2\pi].$$

Here, ϕ is the random phase of each signal. The goal is to estimate not just point estimates of the parameters $\theta = (A, v)$, but entire probability distributions of them, given some noisy observations. We will use the range and mean of a signal as our observable:

$$G(\theta) = \big[\mathrm{range}(f(\theta)), \mathrm{mean}(f(\theta))\big]$$

Then, our noisy observations, y_{obs}, can be written as:

$$y_{obs} = G(\theta^{\dagger}) + \mathcal{N}(0, \Gamma)$$

where Γ is the observational covariance matrix. We will assume the noise to be independent for each observable, giving us a diagonal covariance matrix.

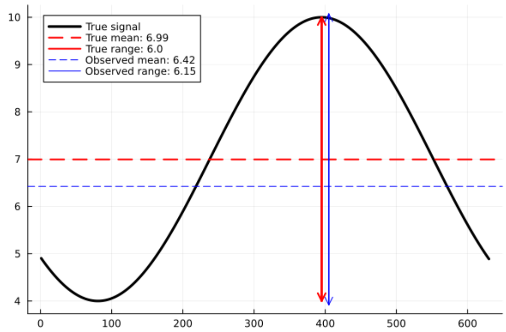

Figure 1: The true and observed range and mean.

For this experiment $\theta^\dagger = (A^\dagger, v^\dagger) = (3.0, 7.0)$, and the noisy observations are displayed in blue in Figure 1.

We define prior distributions on the two parameters. For the amplitude, we define a prior with mean 2 and standard deviation 1. It is additionally constrained to be nonnegative. For the vertical shift we define a prior with mean 0 and standard deviation 5.

```
const PD = CalibrateEmulateSample.ParameterDistributions
prior_u1 = PD.constrained_gaussian("amplitude", 2, 1, 0, Inf)
prior_u2 = PD.constrained_gaussian("vert_shift", 0, 5, -Inf, Inf)
prior = PD.combine_distributions([prior_u1, prior_u2])
```

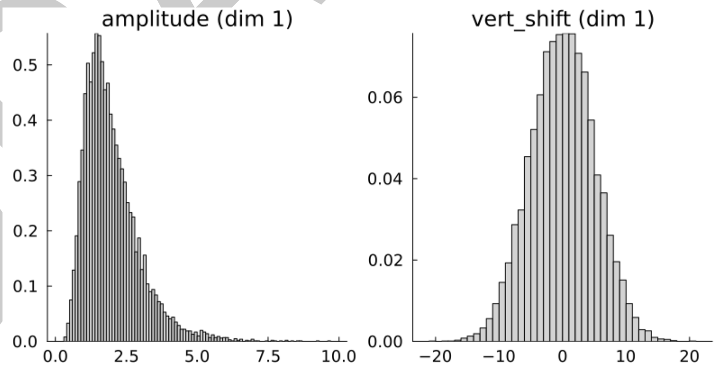

Figure 2: Marginal distributions of the prior

The prior is displayed in Figure 2.

We now adaptively find input-output pairs from our map G in a region of interest using an inversion method (an ensemble Kalman process). This is the Calibrate stage, and iteratively generates parameter combinations, that refine around a region of high posterior mass.

```
const EKP = CalibrateEmulateSample.EnsembleKalmanProcesses
N_ensemble = 10
N_iterations = 5
initial_ensemble = EKP.construct_initial_ensemble(prior, N_ensemble)
ensemble_kalman_process = EKP.EnsembleKalmanProcess(
    initial_ensemble, y_obs, Γ, EKP.Inversion();
)
```

```julia
for i in 1:N_iterations
    params_i = EKP.get_phi_final(prior, ensemble_kalman_process)
    G_ens = hcat([G(params_i[:, i]) for i in 1:N_ensemble]...)
    EKP.update_ensemble!(ensemble_kalman_process, G_ens)
end
```

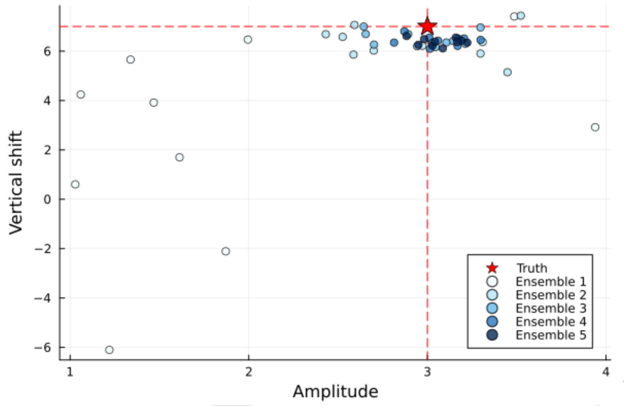

Figure 3: The resulting ensemble from a calibration.

The adaptively refined training points from EKP are displayed in Figure 3. We now build an basic Gaussian process emulator from the GaussianProcesses.jl package to emulate the map G using these points.

```julia
const UT = CalibrateEmulateSample.Utilities
const EM = CalibrateEmulateSample.Emulators

input_output_pairs = UT.get_training_points(
    ensemble_kalman_process, N_iterations,
)
gppackage = EM.GPJL()
gauss_proc = EM.GaussianProcess(gppackage, noise_learn = false)
emulator = EM.Emulator(
    gauss_proc, input_output_pairs, normalize_inputs = true,  obs_noise_cov = Γ,
)
EM.optimize_hyperparameters!(emulator) # train the emulator
```

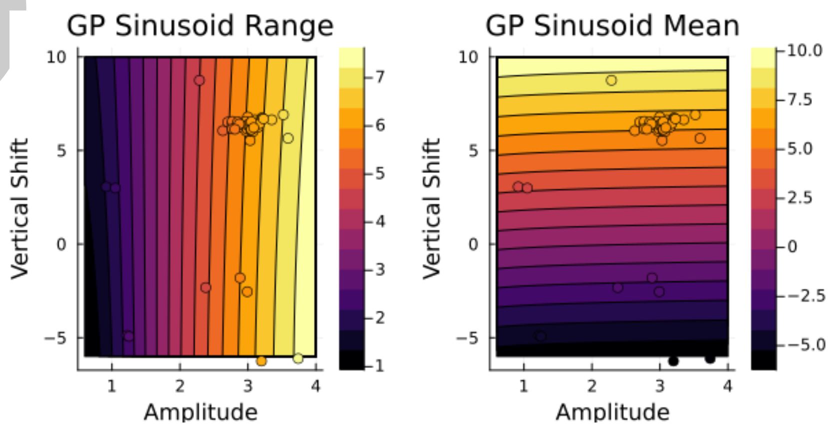

Figure 4: The Gaussian process emulator of the range and mean maps, trained on the re-used calibration pairs

We evaluate the mean of this emulator on a grid, and also show the value of the true G at training point locations in Figure 4.

We can then sample with this emulator using an MCMC scheme. We first choose a good step size (an algorithm parameter) by running some short sampling runs (of length 2,000 steps). Then we run the 100,000 step sampling run to generate samples of the joint posterior distribution.

```
const MC = CalibrateEmulateSample.MarkovChainMonteCarlo
mcmc = MC.MCMCWrapper(
    MC.RWMHSampling(), y_obs, prior, emulator,
)
# choose a step size
new_step = MC.optimize_stepsize(
    mcmc; init_stepsize = 0.1, N = 2000,
)
# Now begin the actual MCMC
chain = MC.sample(
    mcmc, 100_000; stepsize = new_step, discard_initial = 2_000,
)
```

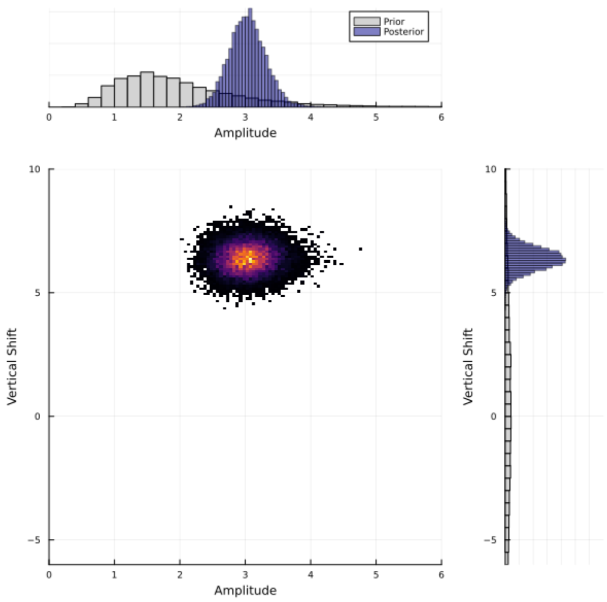

Figure 5: The joint posterior distribution histogram

A histogram of the samples from is displayed in Figure 5. We see that the posterior distribution contains the true value $(3.0, 7.0)$ with high probability.

Research projects using the package

Some research projects that use this codebase, or modifications of it, are (Bieli et al., 2022; Dunbar et al., 2021; Dunbar, Howland, et al., 2022; Hillier, 2022; Howland et al., 2022; King et al., 2023; Mansfield & Sheshadri, 2022).

Acknowledgements

We acknowledge contributions from several others who played a role in the evolution of this package. These include Adeline Hillier, Ignacio Lopez Gomez and Thomas Jackson. The development of this package was supported by the generosity of Eric and Wendy Schmidt by recommendation of the Schmidt Futures program, National Science Foundation Grant AGS-1835860, the Defense Advanced Research Projects Agency (Agreement No. HR00112290030), the Heising-Simons Foundation, Audi Environmental Foundation, and the Cisco Foundation.

References

Bezanson, J., Edelman, A., Karpinski, S., & Shah, V. B. (2017). Julia: A fresh approach to numerical computing. *SIAM Review*, *59*(1), 65–98. https://doi.org/10.1137/141000671

Bieli, M., Dunbar, O. R. A., Jong, E. K. de, Jaruga, A., Schneider, T., & Bischoff, T. (2022). An efficient Bayesian approach to learning droplet collision kernels: Proof of concept using "Cloudy," a new n-moment bulk microphysics scheme. *Journal of Advances in Modeling Earth Systems*, *14*(8), e2022MS002994. https://doi.org/10.1029/2022MS002994

Cleary, E., Garbuno-Inigo, A., Lan, S., Schneider, T., & Stuart, A. M. (2021). Calibrate, emulate, sample. *Journal of Computational Physics*, *424*, 109716. https://doi.org/https://doi.org/10.1016/j.jcp.2020.109716

Cotter, S. L., Roberts, G. O., Stuart, A. M., & White, D. (2013). MCMC Methods for Functions: Modifying Old Algorithms to Make Them Faster. *Statistical Science*, *28*(3), 424–446. https://doi.org/10.1214/13-STS421

Dixit, V. K., & Rackauckas, C. (2022). GlobalSensitivity.jl: Performant and parallel global sensitivity analysis with julia. *Journal of Open Source Software*, *7*(76), 4561. https://doi.org/10.21105/joss.04561

Dunbar, O. R. A., Garbuno-Inigo, A., Schneider, T., & Stuart, A. M. (2021). Calibration and uncertainty quantification of convective parameters in an idealized GCM. *Journal of Advances in Modeling Earth Systems*, *13*(9), e2020MS002454. https://doi.org/https://doi.org/10.1029/2020MS002454

Dunbar, O. R. A., Howland, M. F., Schneider, T., & Stuart, A. M. (2022). Ensemble-based experimental design for targeting data acquisition to inform climate models. *Journal of Advances in Modeling Earth Systems*, *14*(9), e2022MS002997. https://doi.org/https://doi.org/10.1029/2022MS002997

Dunbar, O. R. A., Lopez-Gomez, I., Garbuno-Iñigo, A. G.-I., Huang, D. Z., Bach, E., & Wu, J. (2022). EnsembleKalmanProcesses.jl: Derivative-free ensemble-based model calibration. *Journal of Open Source Software*, *7*(80), 4869. https://doi.org/10.21105/joss.04869

Fairbrother, J., Nemeth, C., Rischard, M., Brea, J., & Pinder, T. (2022). GaussianProcesses.Jl: A nonparametric bayes package for the julia language. *Journal of Statistical Software*, *102*, 1–36.

Garbuno-Inigo, A., Nüsken, N., & Reich, S. (2020). Affine invariant interacting Langevin dynamics for Bayesian inference. *SIAM Journal on Applied Dynamical Systems*, *19*(3), 1633–1658. https://doi.org/10.1137/19M1304891

Hillier, A. (2022). *Supervised calibration and uncertainty quantification of subgrid closure parameters using ensemble Kalman inversion* [Master's thesis, Massachusetts Institute of Technology. Department of Electrical Engineering; Computer Science]. https://hdl.handle.net/1721.1/145140

The Journal of Open Source Software

Howland, M. F., Dunbar, O. R. A., & Schneider, T. (2022). Parameter uncertainty quantification in an idealized GCM with a seasonal cycle. *Journal of Advances in Modeling Earth Systems*, *14*(3), e2021MS002735. https://doi.org/https://doi.org/10.1029/2021MS002735

Huang, D. Z., Huang, J., Reich, S., & Stuart, A. M. (2022). Efficient derivative-free bayesian inference for large-scale inverse problems. *Inverse Problems*, *38*(12), 125006. https://doi.org/10.1088/1361-6420/ac99fa

Iglesias, M. A., Law, K. J., & Stuart, A. M. (2013). Ensemble kalman methods for inverse problems. *Inverse Problems*, *29*(4), 045001.

King, R. C., Mansfield, L. A., & Sheshadri, A. (2023). *Bayesian history matching applied to the calibration of a gravity wave parameterization* [Preprint]. https://doi.org/10.22541/essoar.170365299.96491153/v1

Liu, F., Huang, X., Chen, Y., & Suykens, J. A. K. (2022). Random features for kernel approximation: A survey on algorithms, theory, and beyond. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, *44*(10), 7128–7148. https://doi.org/10.1109/TPAMI.2021.3097011

Mansfield, L. A., & Sheshadri, A. (2022). Calibration and uncertainty quantification of a gravity wave parameterization: A case study of the Quasi-Biennial Oscillation in an intermediate complexity climate model. *Journal of Advances in Modeling Earth Systems*, *14*(11). https://doi.org/10.1029/2022MS003245

Pedregosa, F., Varoquaux, G., Gramfort, A., Michel, V., Thirion, B., Grisel, O., Blondel, M., Prettenhofer, P., Weiss, R., Dubourg, V., Vanderplas, J., Passos, A., Cournapeau, D., Brucher, M., Perrot, M., & Duchesnay, E. (2011). Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, *12*, 2825–2830.

Rahimi, A., & Recht, B. (2008). Uniform approximation of functions with random bases. *2008 46th Annual Allerton Conference on Communication, Control, and Computing*, 555–561.

Rahimi, A., Recht, B., & others. (2007). Random features for large-scale kernel machines. *NIPS*, *3*, 5.

Sherlock, C., Fearnhead, P., & Roberts, G. O. (2010). The random walk metropolis: Linking theory and practice through a case study. *Statistical Science*, *25*(2), 172–190. http://www.jstor.org/stable/41058939

Tankhilevich, E., Ish-Horowicz, J., Hameed, T., Roesch, E., Kleijn, I., Stumpf, M. P. H., & He, F. (2020). GpABC: a Julia package for approximate Bayesian computation with Gaussian process emulation. *Bioinformatics*. https://doi.org/10.1093/bioinformatics/btaa078

Williams, C. K., & Rasmussen, C. E. (2006). *Gaussian processes for machine learning* (Vol. 2). MIT press Cambridge, MA.