EnsembleKalmanProcesses.jl: Derivative-free ensemble-based model calibration

Oliver R. A. Dunbar[1*¶], **Ignacio Lopez-Gomez**[1*], **Alfredo Garbuno-Iñigo**[2], **Daniel Zhengyu Huang**[1], **Eviatar Bach**[1], **and Jin-long Wu**[3]

1 Division of Geological and Planetary Sciences, California Institute of Technology **2** Department of Statistics, Mexico Autonomous Institute of Technology **3** Department of Mechanical Engineering, University of Wisconsin-Madison ¶ Corresponding author * These authors contributed equally.

DOI: 10.xxxxxx/draft

Software

- Review
- Repository
- Archive

Editor: Open Journals

Reviewers:

- @openjournals

Submitted: 01 January 1970
Published: unpublished

Summary

`EnsembleKalmanProcesses.jl` is a Julia-based toolbox that can be used for a broad class of black-box gradient-free optimization problems. Specifically, the tools enable the optimization, or calibration, of parameters within a computer model in order to best match user-defined outputs of the model with available observed data (Kennedy & O'Hagan, 2001). Some of the tools can also approximately quantify parametric uncertainty (Huang, Huang, et al., 2022). Though the package is written in Julia (Bezanson et al., 2017), a read–write TOML-file interface is provided so that the tools can be applied to computer models implemented in any language. Furthermore, the calibration tools are non-intrusive, relying only on the ability of users to compute an output of their model given a parameter value.

As the package name suggests, the tools are inspired by the well-established class of ensemble Kalman methods. Ensemble Kalman filters are currently one of the only practical ways to assimilate large volumes of observational data into models for operational weather forecasting (Evensen, 1994; Houtekamer & Mitchell, 1998, 2001). In the data assimilation setting, a computational weather model is integrated for a short time over a collection, or ensemble, of initial conditions, and the ensemble is updated frequently by a variety of atmospheric observations, allowing the forecasts to keep track of the real system.

The workflow is similar for ensemble Kalman processes. Here, a computer code is run (in parallel) for an ensemble of different values of the parameters that require calibration, producing an ensemble of outputs. This ensemble of outputs is then compared to observed data, and the parameters are updated to a new set of values which reduce the output–data misfit. The process is iterated until a user-defined criterion of convergence is met. Optimality of the update is guaranteed for linear models and Gaussian uncertainties, but good performance is observed outside of these settings, see Schillings & Stuart (2017). Optimal values are selected from statistics of the final ensemble.

Statement of need

The task of estimating parameters of a computer model or simulator such that its outputs fit with data is ubiquitous in science and engineering, coming under many names such as calibration, inverse problems, and parameter estimation. In statistics and machine learning, when closed-form estimators of parameters of a model are unavailable, similar approaches may need to be employed to fit the model to data. There is a wide variety of algorithms to suit these applications; however, there are many bottlenecks in the practical application of such methods to computer codes:

- Legacy codes: Often code is old, and written in different languages than the packages implementing the calibration algorithms, requiring elaborate interfaces.
- Complex codes: Often large complex codes are difficult to change, so application of intrusive calibration tools to models can be challenging.
- Derivatives: When derivatives of a model output can be taken with respect to parameters, they can often improve the rate of convergence. But in many practical cases, these parameter-to-output maps are not differentiable; they may be chaotic or stochastic. Here one should not – or cannot – apply gradient-based methods.
- Lack of parallelism: There is now widespread access to high-performance computing clusters, cloud computing, and local multi-threading, and such facilities should be exploited where possible.

`EnsembleKalmanProcesses.jl` aims to provide a flexible and comprehensive solution to address these challenges:

1. It is embarrassingly parallel with respect to the ensemble; therefore, all computer model evaluations within an ensemble can happen simultaneously within an iteration.
2. It is derivative-free, and so is appropriate for computer codes for which derivatives are not available. The optimal updates are robust to noise.
3. It is non-intrusive and so can be applied to black-box computer codes written in any language or style, or to computer models for which the source code is not available to the user.
4. With scalability enhancements, such as the ones provided by the `Localizer` structure, it can be applied to high-dimensional problems.

State of the field

Many gradient-based optimizers have been implemented in Julia, collected in `Optim.jl` (Mogensen & Riseth, 2018) and `JuliaSmoothOptimizers.jl`, for example. Some gradient-free optimization tools, better suited for non-deterministic or noisy optimization, are collected within packages such as `BlackBoxOptim.jl` and `Metaheuristics.jl` (Mejía-de-Dios & Mezura-Montes, 2022). Although these packages feature a number of ensemble-based approaches, none utilize Kalman-based statistical updates, and instead rely on heuristic algorithms inspired from biological processes such as natural selection (Genetic Algorithms) or swarming (Particle Swarm Optimization). A related class of methods to calibrate black-box computer codes are based on Bayesian inference, such as (Markov Chain) Monte Carlo, implemented in `Turing.jl` (Ge et al., 2018), for example. Such methods provide the posterior distribution of parameters, from which the optimum is taken as the summary statistic. However, they are far more computationally expensive.

`EnsembleKalmanProcesses.jl` fills the need for computationally inexpensive, gradient-free, mathematically-grounded, ensemble-based calibration algorithms. Ensemble Kalman processes are provably optimal in simple settings, and have a large literature of extensions to complex problems. Although implementations of Kalman filters exist in Julia (`EnKF.jl`; `Kalman.jl`; `GaussianFilters.jl`), `EnsembleKalmanProcesses.jl` is the only package to implement ensemble-based updates for parameter estimation; other packages focus on state estimation from sequential data.

Features

There are different ensemble Kalman algorithms in the literature, which differ in the way that the ensemble update is performed. The following ensemble Kalman processes are implemented tools in our package, and we provide published references for detailed descriptions and evidence of their efficacy:

- Ensemble Kalman Inversion (EKI, Iglesias et al. (2013)),
- Ensemble Kalman Sampler (EKS, Garbuno-Inigo, Hoffmann, et al. (2020); Garbuno-Inigo, Nüsken, et al. (2020)),
- Unscented Kalman Inversion (UKI, Huang, Schneider, et al. (2022)),
- Sparse Ensemble Kalman Inversion (SEKI, Schneider, Stuart, et al. (2022)).

The package also implements some features to improve robustness and flexibility of the ensemble algorithms:

- The `ParameterDistribution` structure allows users to perform calibrations for parameters with known constraints. It does so by defining transformation maps under-the-hood from the constrained space to an unconstrained space where the optimization problem can be suitably defined. Constrained optimization using this framework has been successfully demonstrated in a variety of settings (Dunbar et al., 2022; Lopez-Gomez et al., 2022; Schneider, Dunbar, et al., 2022).
- The `FailureHandler` structure allows calibrations to continue when several ensemble members fail. Common reasons for failure could be, for instance, simulation blow-up for certain parameter configurations, user termination of slow computations, data corruption, or bad nodes in a high-performance computing facility. This methodology is demonstrated in Lopez-Gomez et al. (2022).
- The `Localizer` structure allows users to overcome the restriction of the solution of the calibration to the linear span of the initial ensemble, and to reduce sampling errors due to the finite size of the ensemble. Various such localization and sampling error correction methods are implemented in `EnsembleKalmanProcesses.jl` (Lee, 2021; Tong & Morzfeld, 2022).
- The TOML-file interface defined in the `TOMLInterface` module allows non-intrusive use of `EnsembleKalmanProcesses.jl` through TOML files, which are widely used for configuration files and easily read in any programming language. Given the computer model to calibrate and prior distributions on the parameters, `EnsembleKalmanProcesses.jl` reads these distributions from a file and, after an iteration of the ensemble Kalman algorithm, writes each member of the updated ensemble to a parameter file. Each of these parameter files can be then read individually to initiate the ensemble of the computer model for the next iteration.

Pedagogical example

In this example, the computer code simulates a sine curve

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

with a random phase shift φ applied to every evaluation. We define the observable map

$$G(A, v) = [\max f(A, v) - \min f(A, v), \mathrm{mean} f(A, v)].$$

We treat φ as a "nuisance parameter" that we are not interested in estimating, thus the observable map $G(A, v)$ is chosen independent of φ. We are given one sample measurement of G, polluted by Gaussian noise $\mathcal{N}(0, \Gamma)$, and call this y. Our task is to deduce the most likely amplitude A and vertical shift v of the curve that produced the sample y.

We encode information into prior distributions over the parameters:

```julia
# A is positive, has likely value 2 with standard deviation 1
# v has likely value 0 with standard deviation 5
prior_A = constrained_gaussian("amplitude", 2, 1, 0, Inf)
prior_v = constrained_gaussian("vert_shift", 0, 5, -Inf, Inf)
prior = combine_distributions([prior_A, prior_v])
```

To use a basic ensemble method we need to specify the size of the ensemble, which we take to be `N_ensemble = 5`. We now initialize the problem, by drawing the initial ensemble from the prior and selecting the `Inversion()` tool to perform ensemble Kalman inversion:

```julia
initial_ensemble = construct_initial_ensemble(prior, N_ensemble)
ensemble_kalman_inversion =
    EnsembleKalmanProcess(initial_ensemble, y, Γ, Inversion())
```

Then we run the algorithm iteratively. In this case, we choose to perform 5 iterations:

```julia
N_iterations = 5
for i in 1:N_iterations`
    # get the latest parameter ensemble
    params_i = get_phi_final(prior, ensemble_kalman_process)
    # run a simulation for each parameter in the ensemble
    G_ens = hcat([G(params_i[:, i]) for i in 1:N_ensemble]...)
    # perform the Kalman update, producing a new ensemble
    update_ensemble!(ensemble_kalman_process, G_ens)
end
```

The initial and final ensembles are shown in Figure 1, by evaluating f at these parameters. We observe that the final sinusoid ensemble has greatly reduced the error in amplitude and vertical shift, despite the presence of the random phase shifts.

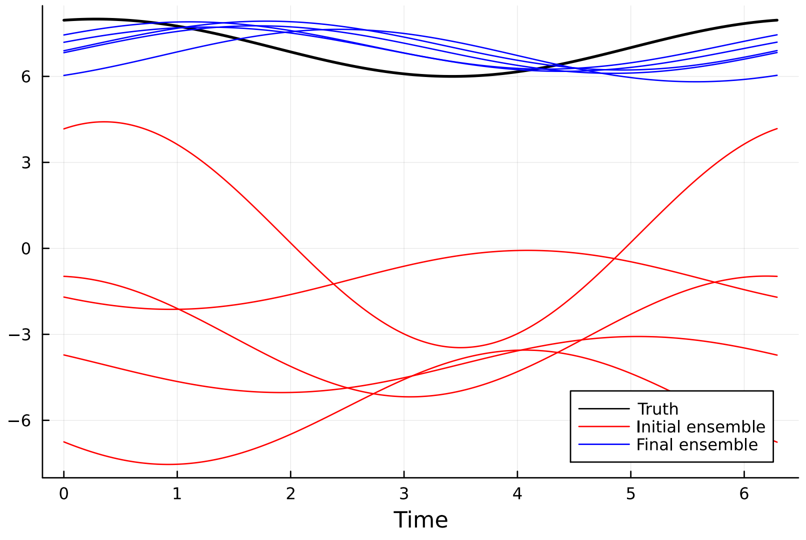

Figure 1: Sinusoids produced from the initial and final ensembles, and the sine curve that generated the data (Truth).

This final ensemble determines the problem solution; for ensemble Kalman inversion, a best estimate of the parameters is taken as the mean of the final ensemble:

```julia
best_parameter_estimate = get_phi_mean_final(prior, ensemble_kalman_process)
```

The Julia code and further explanation of this example is provided in the documentation.

Research projects using the package

- `EnsembleKalmanProcesses.jl` has been used to train physics-based and machine-learning models of atmospheric turbulence and convection, implemented using `Flux.jl` and `TurbulenceConvection.jl` (Lopez-Gomez et al., 2022). In this application, the available model outputs are not differentiable with respect to the learnable parameters, so gradient-based optimization was not an option. In addition, the unscented Kalman inversion algorithm was used to approximately quantify parameter uncertainty.
- `EnsembleKalmanProcesses.jl` features within Calibrate-Emulate-Sample (CES, Cleary et al. (2021)), a pipeline used to accelerate parameter uncertainty quantification (by a factor of 10^3 - 10^4 with respect to Monte Carlo methods) by using statistical emulators. `EnsembleKalmanProcesses.jl` is used to choose training points for these emulators. The training points are naturally concentrated by the ensemble Kalman processes into areas of high posterior probability mass. Within CES, the trained emulators are used to sample this probability distribution, and by design are most accurate where they need to be. CES has been successfully used to quantify parameter uncertainty within the moist convection scheme of a simplified climate model (Dunbar et al., 2021, 2022; Howland et al., 2022), within a droplet collision-coalescence scheme for cloud microphyiscs (Bieli et al., 2022), and within boundary layer turbulence schemes for ocean modeling (Hillier, 2022).
- `EnsembleKalmanProcesses.jl` has been used to learn hyperparameters within a machine learning tool known as Random Features within the Julia package `RandomFeatures.jl`. Here, the hyperparameters characterize an infinite family of functions, from which a finite sample is drawn to use as a basis in regression problems. The objective for learning the parameters is noisy and non-differentiable due to the random sampling, so ensemble Kalman processes naturally perform well in this setting.

Acknowledgements

We acknowledge contributions from several others who played a role in the evolution of this package. These include Jake Bolewski, Navid Constantinou, Gregory L. Wagner, Thomas Jackson, Michael Howland, Melanie Bieli, and Adeline Hillier. The development of this package was supported by the generosity of Eric and Wendy Schmidt by recommendation of the Schmidt Futures program, and by the Defense Advanced Research Projects Agency (Agreement No. HR00112290030).

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/10.1016/j.jcp.2020.109716

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/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/10.1029/2022MS002997

Evensen, G. (1994). Sequential data assimilation with a nonlinear quasi-geostrophic model using Monte Carlo methods to forecast error statistics. *Journal of Geophysical Research: Oceans*, *99*, 10143–10162. https://doi.org/10.1029/94JC00572

Garbuno-Inigo, A., Hoffmann, F., Li, W., & Stuart, A. M. (2020). Interacting Langevin diffusions: Gradient structure and ensemble Kalman sampler. *SIAM Journal on Applied Dynamical Systems*, *19*(1), 412–441. https://doi.org/10.1137/19M1251655

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

Ge, H., Xu, K., & Ghahramani, Z. (2018). Turing: A language for flexible probabilistic inference. *International Conference on Artificial Intelligence and Statistics, AISTATS 2018, 9-11 April 2018, Playa Blanca, Lanzarote, Canary Islands, Spain*, 1682–1690. http://proceedings.mlr.press/v84/ge18b.html

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

Houtekamer, P. L., & Mitchell, H. L. (1998). Data assimilation using an ensemble Kalman filter technique. *Monthly Weather Review*, *126*, 796–811. https://doi.org/10.1175/1520-0493(1998)126%3C0796:DAUAEK%3E2.0.CO;2

Houtekamer, P. L., & Mitchell, H. L. (2001). A sequential ensemble Kalman filter for atmospheric data assimilation. *Monthly Weather Review*, *129*, 123–137. https://doi.org/10.1175/1520-0493(2001)129%3C0123:ASEKFF%3E2.0.CO;2

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/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

Huang, D. Z., Schneider, T., & Stuart, A. M. (2022). Iterated Kalman methodology for inverse problems. *Journal of Computational Physics*, *463*, 111262. https://doi.org/10.1016/j.jcp.2022.111262

Iglesias, M. A., Law, K. J., & Stuart, A. M. (2013). Ensemble Kalman methods for inverse problems. *Inverse Problems*, *29*(4), 045001. https://doi.org/10.1088/0266-5611/29/4/045001

Kennedy, M., & O'Hagan, A. (2001). Bayesian calibration of computer models. *Journal of the Royal Statistical Society Series B*, *63*, 425–464. https://doi.org/10.1111/1467-9868.00294

Lee, Y. (2021). *Sampling error correction in ensemble Kalman inversion*. https://doi.org/10.48550/arxiv.2105.11341

Lopez-Gomez, I., Christopoulos, C., Langeland Ervik, H. L., Dunbar, O. R. A., Cohen, Y., & Schneider, T. (2022). Training physics-based machine-learning parameterizations with gradient-free ensemble Kalman methods. *Journal of Advances in Modeling Earth Systems*, *14*(8), e2022MS003105. https://doi.org/10.1029/2022MS003105

Mejía-de-Dios, J.-A., & Mezura-Montes, E. (2022). Metaheuristics: A julia package for single- and multi-objective optimization. *Journal of Open Source Software*, *7*(78), 4723. https://doi.org/10.21105/joss.04723

Mogensen, P. K., & Riseth, A. N. (2018). Optim: A mathematical optimization package for julia. *Journal of Open Source Software*, *3*(24), 615. https://doi.org/10.21105/joss.00615

Schillings, C., & Stuart, A. M. (2017). Analysis of the ensemble Kalman filter for inverse problems. *SIAM Journal on Numerical Analysis*, *55*(3), 1264–1290. https://doi.org/10.1137/16M105959X

Schneider, T., Dunbar, O. R. A., Wu, J., Böttcher, L., Burov, D., Garbuno-Inigo, A., Wagner, G. L., Pei, S., Daraio, C., Ferrari, R., & Shaman, J. (2022). Epidemic management and control through risk-dependent individual contact interventions. *PLOS Computational Biology*, *18*(6), e1010171. https://doi.org/10.1371/journal.pcbi.1010171

Schneider, T., Stuart, A. M., & Wu, J.-L. (2022). Ensemble Kalman inversion for sparse learning of dynamical systems from time-averaged data. *Journal of Computational Physics*, 111559. https://doi.org/10.1016/j.jcp.2022.111559

Tong, X. T., & Morzfeld, M. (2022). *Localization in ensemble Kalman inversion*. https://doi.org/10.48550/arXiv.2201.10821