parafields: A generator for distributed, stationary Gaussian processes

Dominic Kempf [1,2*¶], **Ole Klein** [4*], **Robert Kutri** [2,3], **Robert Scheichl** [2,3], **and Peter Bastian** [2]

1 Scientific Software Center, Heidelberg University **2** Interdisciplinary Center for Scientific Computing, Heidelberg University **3** Institute for Mathematics, Heidelberg University **4** Independent Researcher ¶ Corresponding author * These authors contributed equally.

DOI: 10.21105/joss.05735

Software

- Review
- Repository
- Archive

Editor: Patrick Diehl

Reviewers:

- @shahmoradi
- @gchure

Submitted: 03 July 2023
Published: 18 December 2023

Summary

Parafields is a Python package for the generation of stationary Gaussian random fields with well-defined, known statistical properties. The use of such fields is a key ingredient of simulation workflows that involve uncertain, spatially heterogeneous parameters. As such, Gaussian Random Fields play a dominant role in geostatistics, e.g. in the modelling of particulate matter concentration, temperature distributions and subsurface flow (Cameletti et al., 2013) (Sain et al., 2011) (Dodwell et al., 2015). Outside these traditional applications, Gaussian random fields are also used in biomedical imaging (Penny et al., 2005), material sciences (Torquato & Haslach Jr, 2002) or within Markov-Chain Monte-Carlo methods in Bayesian estimation (Scheichl et al., 2017).

Parafields is also able to run in parallel using the Message Passing Interface (MPI) standard through mpi4py (Dalcin & Fang, 2021). In this case, the computational domain is split and only the part of the random field relevant to a certain process is generated on that process. The generation process is implemented in a performance-oriented C++ backend library and exposed in Python though an intuitive Python interface.

Statement of need

The simulation of large-scale Gaussian random fields is a computationally challenging task, particularly if the considered field has a short correlation length when compared to its computational domain.

However, when the random field in question is stationary, that is, its covariance function is translation invariant, fast and exact methods of simulation based on the Fast Fourier Transform have been proposed in (Dietrich & Newsam, 1997) and (Wood & Chan, 1994). These can outperform more traditional, factorization-based methods both in terms of scaling as well as absolute performance.

Through the combination of an efficient C++ backend with an easy-to-use Python interface this package aims to make these methods accessible for integration into existing workflows. This separation also allows the package to support both large-scale, peformance-oriented applications, as well as providing a means to quickly generate working prototypes using just a few lines.

Other packages for the generation of stationary Gaussian processes exist, like e.g. the R package lgcp (Davies & Bryant, 2013), the Julia package GaussianRandomFields.jl (Robbe, 2023) or the Python package GSTools (Müller et al., 2022). In comparison with these alternative packages, parafields is specifically designed and adapted to the sampling of very large Gaussian random fields within a HPC workflow. This was a major concern in the development of the backend and is among other things, reflected in the possibility to create Gaussian processes in a an MPI-distributed fashion.

Implementation

Parafields looks back at over ten years of development history: It was first implemented as an extension to the Dune framework (Bastian et al., 2021) for the numerical solution of partial differential equations. This restricted the potential user base to users of that software framework, although there was quite some interest in the software from outside this community. In 2022, we started a huge refactoring: The previous C++ code base (Klein, 2017) got rewritten to have a weaker dependency on Dune, which e.g. included a rewrite of the CMake build system (Klein & Kempf, 2022). In order to open up to a wider user base, a Python interface written in pybind11 (Jakob et al., 2017) was added.

When engineering the Python package, we put special emphasis on the following usability aspects: Installability, customizability and embedding into existing user workflows.

The recommended installation procedure for parafields is perfectly aligned with the state-of-the-art of the Python language: It is installable through `pip` and automatically compiles using the CMake build system of the project through scikit-build (Fillion-Robin et al., 2018). Required dependencies of the C++ library are automatically fetched and built in the required configuration. For sequential usage we also provide pre-compiled Python wheels. They are built against the sequential MPI stub library FakeMPI (Kempf & PetSc Developers, 2022), which allows us to build the sequential and the parallel version from the same code base. Users that want to leverage MPI through mpi4py will instead build the package from source against their system MPI library.

It was a goal of the design of the Python API to expose as much of the flexibility of the underlying C++ framework as possible. In order to do so, we use pybind11's capabilities to pass Python callables to the C++ backend. This allows users to e.g. implement custom covariance functions or use different random number generators. Furthermore, we acknowledge the fact that many Python users write scientific applications within Jupyter: Our fields render nicely as images in Jupyter and field generation can optionally be configured through an interactive widget frontend within Jupyter.

Acknowledgments

The authors thank all contributors of the dune-randomfield project for their valuable contributions that are now part of the parafields-core library. Dominic Kempf is employed by the Scientific Software Center of Heidelberg University which is funded as part of the Excellence Strategy of the German Federal and State Governments. Ole Klein's work is supported by the federal ministry of education and research of Germany (Bundesministerium für Bildung und Forschung) and the ministry of science, research and arts of the federal state of Baden-Württemberg (Ministerium für Wissenschaft, Forschung und Kunst Baden-Württemberg).

Bastian, P., Blatt, M., Dedner, A., Dreier, N.-A., Engwer, C., Fritze, R., Gräser, C., Grüninger, C., Kempf, D., Klöfkorn, R., Ohlberger, M., & Sander, O. (2021). The dune framework: Basic concepts and recent developments. *Computers & Mathematics with Applications*, *81*, 75–112. https://doi.org/10.1016/j.camwa.2020.06.007

Cameletti, M., Lindgren, F., Simpson, D., & Rue, H. (2013). Spatio-temporal modeling of particulate matter concentration through the SPDE approach. *AStA Advances in Statistical Analysis*, *97*, 109–131. https://doi.org/10.1007/s10182-012-0196-3

Dalcin, L., & Fang, Y.-L. L. (2021). mpi4py: Status update after 12 years of development. *Computing in Science & Engineering*, *23*(4), 47–54. https://doi.org/10.1109/MCSE.2021.3083216

Davies, T. M., & Bryant, D. (2013). On circulant embedding for gaussian random fields in r. *Journal of Statistical Software*, *55*(9), 1–21. https://doi.org/10.18637/jss.v055.i09

Dietrich, C. R., & Newsam, G. N. (1997). Fast and exact simulation of stationary gaussian processes through circulant embedding of the covariance matrix. *SIAM Journal on Scientific Computing*, *18*(4), 1088–1107. https://doi.org/10.1137/s1064827592240555

Dodwell, T. J., Ketelsen, C., Scheichl, R., & Teckentrup, A. L. (2015). A hierarchical multilevel markov chain monte carlo algorithm with applications to uncertainty quantification in subsurface flow. *SIAM/ASA Journal on Uncertainty Quantification*, *3*(1), 1075–1108. https://doi.org/10.1137/130915005

Fillion-Robin, J.-C., McCormick, M., Padron, O., Smolens, M., Grauer, M., & Sarahan, M. (2018). *jcfr/scipy_2018_scikit-build_talk: SciPy 2018 Talk | scikit-build: A Build System Generator for CPython C/C++/Fortran/Cython Extensions* (Version v1.0). Zenodo. https://doi.org/10.5281/zenodo.2565368

Jakob, W., Rhinelander, J., & Moldovan, D. (2017). *pybind11 – seamless operability between c++11 and python*.

Kempf, D., & PetSc Developers, the. (2022). *FakeMPI - a sequential MPI stub*.

Klein, O. (2017). *Dune-randomfield - generation of gaussian random fields in arbitrary dimensions, based on circulant embedding*.

Klein, O., & Kempf, D. (2022). In *GitHub repository*. GitHub. https://github.com/parafields/parafields-core

Müller, S., Schüler, L., Zech, A., & Heße, F. (2022). `GSTools` v1.3: A toolbox for geostatistical modelling in python. *Geoscientific Model Development*, *15*(7), 3161–3182. https://doi.org/10.5194/gmd-15-3161-2022

Penny, W. D., Trujillo-Barreto, N. J., & Friston, K. J. (2005). Bayesian fMRI time series analysis with spatial priors. *NeuroImage*, *24*(2), 350–362. https://doi.org/10.1016/j.neuroimage.2004.08.034

Robbe, P. (2023). GaussianRandomFields.jl: A julia package to generate and sample from gaussian random fields. *Journal of Open Source Software*, *8*(89), 5595. https://doi.org/10.21105/joss.05595

Sain, S. R., Furrer, R., & Cressie, N. (2011). A spatial analysis of multivariate output from regional climate models. *The Annals of Applied Statistics*, 150–175. https://doi.org/10.1214/10-AOAS369

Scheichl, R., Stuart, A. M., & Teckentrup, A. L. (2017). Quasi-monte carlo and multilevel monte carlo methods for computing posterior expectations in elliptic inverse problems. *SIAM/ASA Journal on Uncertainty Quantification*, *5*(1), 493–518. https://doi.org/10.1137/16m1061692

Torquato, S., & Haslach Jr, H. (2002). Random heterogeneous materials: Microstructure and macroscopic properties. *Appl. Mech. Rev.*, *55*(4), B62–B63. https://doi.org/10.1115/1.1483342

Wood, A. T., & Chan, G. (1994). Simulation of stationary gaussian processes in [0, 1] d. *Journal of Computational and Graphical Statistics*, *3*(4), 409–432. https://doi.org/10.2307/1390903