The Journal of Open Source Software

MUQ: The MIT Uncertainty Quantification Library

Matthew Parno[*1], **Andrew Davis**[2], **and Linus Seelinger**[3]

1 Department of Mathematics, Dartmouth College, Hanover, NH USA **2** Courant Institute of Mathematical Sciences, New York University, New York, NY USA **3** Institute for Scientific Computing, Heidelberg University, Heidelberg, Germany

DOI: 10.21105/joss.03076

Software

- Review
- Repository
- Archive

Editor: Pierre de Buyl

Reviewers:

- @martinmodrak
- @georgiastuart

Submitted: 26 February 2021
Published: 09 December 2021

Summary

Scientists and engineers frequently rely on mathematical and numerical models to interpret observational data, forecast system behavior, and make decisions. However, unknown and neglected physics, limited and noisy data, and numerical error result in uncertain model predictions. The MIT Uncertainty Quantification library (MUQ) is a modular software framework for defining and solving uncertainty quantification problems involving complex models. MUQ is written in C++ but uses pybind11 (Jakob et al., 2017) to provide a nearly comprehensive Python interface. Users can access nearly all of MUQ's capabilities from either language.

MUQ provides users many commonly used UQ tools and its modular design allows developers to easily modify, extend, and advance existing algorithms. For example, MUQ allows exact sampling of non-Gaussian distributions (e.g., Markov chain Monte Carlo and importance sampling), approximating computationally intensive forward models (e.g., polynomial chaos expansions and Gaussian process regression), working with integral covariance operators (e.g., Gaussian processes and Karhunen-Loève decompositions), and characterizing predictive uncertainties. The software is designed to support algorithm developers who want to easily construct new algorithms by exploiting a wide variety of existing algorithmic building blocks. Many UQ algorithms are model agnostic: Different physics-based or statistical models can be substituted into the algorithm based on the application. Therefore, MUQ enables users to quickly implement new models and exploit state-of-the art UQ algorithms.

A suite of documented examples, including Gaussian process regression of Mauna Loa C02 observations, global sensitivity analysis of an Euler-Bernoulli beam, and a hierarchical Bayesian model of groundwater pump-test data, are provided to guide users through the process of implementing their own models and leveraging MUQ's UQ algorithms on quasi-realistic applications.

Statement of need

Scientists and engineers are increasingly using physical and statistical models to inform policy, system design, and experiments. Although useful tools, models are inherently error prone and assessing predictive capabilities and robustness requires rigorous uncertainty quantification (UQ). The last decade has seen an explosion in the number and complexity of algorithms for characterizing various sources of uncertainty (e.g, Kennedy & O'Hagan (2001), Conrad & Marzouk (2013), Giles (2008), Sargsyan et al. (2019), Cotter et al. (2013), Cui et al. (2016), Han & Liu (2018), Detommaso et al. (2018)). The complexity of many recent advancements makes it difficult to rigorously compare new algorithms against the current state-of-the-art and for users to leverage these new tools on practical applications. Likewise, many interesting

*Corresponding Author.

models are developed, but due to a lack of a common interface they are often not widely used. MUQ aims to reduce the gap between algorithmic research and application by providing a software pipeline between the algorithmic development community and UQ practitioners. The goal is to reduce the costly and error prone practice of reimplementing state-of-the-art techniques, lower the barriers preventing widespread use of cutting-edge techniques, and provide an algorithm-agnostic model interface. MUQ leverages well-known packages such as Eigen3 (Guennebaud et al., 2010), Stan Math (Carpenter et al., 2015), NLOpt (Johnson, 2007), Sundials (Hindmarsh et al., 2005), Nanoflann (Blanco & Rai, 2014), and boost (Boost, 2015), to help make this possible.

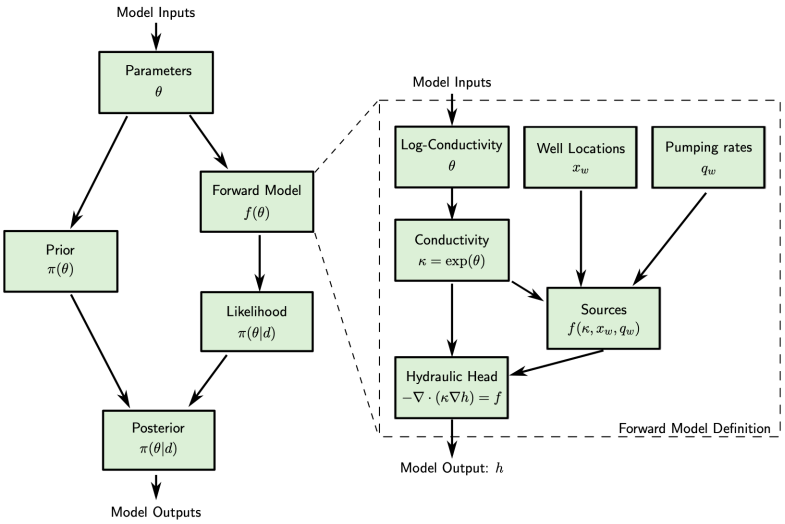

Figure 1: MUQ allows for complicated models to be constructed by connecting model components on a graph. Here is a possible graph for a Bayesian inverse problem built on a model for groundwater flow. MUQ treats each box as a black-box, but if all components can provide derivative information individually, e.g., through adjoint methods, then MUQ can compute gradients, Jacobians, and Hessian actions through the entire graph.

While MUQ is capable of solving both forward and inverse UQ problems, its primary focus is solving of Bayesian inverse problems with computationally expensive models, as demonstrated in Davis et al. (2021), and potentially high dimensional parameter spaces. In comparison, other MCMC and UQ packages, such as Stan (Carpenter et al., 2017), BUGS (Lunn et al., 2009), or JAGS (Plummer & others, 2003), are rooted in the statistics community. MUQ is particularly useful when the target distribution depends on complicated physical models that are difficult to implement and computationally expensive. MUQ employs a semi-intrusive "gray-box" approach (see Figure Figure 1) that enables efficient gradient calculations, through techniques like the adjoint methods used in PDE constrained optimization, but does not rely on automatic differentiation and does not place any restrictions on how the model is implemented (language, solver, etc...). Other sampling packages, such as PyMC3 (Salvatier et al., 2016), Stan (Carpenter et al., 2017), and tensorflow-probability (Lao et al., 2020), have adopted various probabilistic programming approaches that are more intrusive than MUQ's gray-box approach. Although such tools are useful, they make it more difficult for users to expose efficient gradient evaluation techniques for problems that rely on PDE solvers or other complex forward models. MUQ also has a variety of algorithms (e.g., Cotter et al. (2013); Cui et al. (2016)) for tackling discretizations of infinite-dimensional Bayesian inverse problems and several other novel MCMC implementations, including multi-level Dodwell et al. (2019) and multi-index MCMC, local approximation algorithms Davis et al. (2021), and adaptive transport map MCMC (Parno & Marzouk, 2018).

Acknowledgements

We are grateful for the many users and developers in the MUQ community. In particular, we would like to acknowledge software contributions from Alexandra Datz, Arnold Song, Brendan West, Devin O'Connor, Taylor Hodgdon, Patrick Conrad, Josephine Westermann, Ki-Tae Kim, Max Liu, and Cassie Lumbrazo. We would also like to acknowledge financial and technical support from Youssef Marzouk, Matthew Farthing, Peter Bastian, and Robert Scheichl.

This material is based upon work supported by the National Science Foundation under Grant No. ACI-1550487.

This material is based upon work supported by the US Department of Energy, Office of Advanced Scientific Computing Research, SciDAC (Scientific Discovery through Advanced Computing) program under awards DE-SC0007099 and DE-SC0021226, for the QUEST and FASTMath SciDAC Institutes.

References

Blanco, J. L., & Rai, P. K. (2014). *Nanoflann: A C++ header-only fork of FLANN, a library for nearest neighbor (NN) with KD-trees*. https://github.com/jlblancoc/nanoflann.

Boost. (2015). *Boost C++ Libraries*. http://www.boost.org/.

Carpenter, B., Gelman, A., Hoffman, M. D., Lee, D., Goodrich, B., Betancourt, M., Brubaker, M., Guo, J., Li, P., & Riddell, A. (2017). Stan: A probabilistic programming language. *Journal of Statistical Software*, *76*(1), 1–32. https://doi.org/10.18637/jss.v076.i01

Carpenter, B., Hoffman, M. D., Brubaker, M., Lee, D., Li, P., & Betancourt, M. (2015). *The Stan Math library: Reverse-mode automatic differentiation in C++*. https://arxiv.org/abs/1509.07164

Conrad, P. R., Davis, A. D., Marzouk, Y. M., Pillai, N. S., & Smith, A. (2018). Parallel local approximation MCMC for expensive models. *SIAM/ASA Journal on Uncertainty Quantification*, *6*(1), 339–373. https://doi.org/10.1137/16M1084080

Conrad, P. R., & Marzouk, Y. M. (2013). Adaptive Smolyak pseudospectral approximations. *SIAM Journal on Scientific Computing*, *35*(6), A2643–A2670. https://doi.org/10.1137/120890715

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*, 424–446. https://doi.org/10.1214/13-STS421

Cui, T., Law, K. J., & Marzouk, Y. M. (2016). Dimension-independent likelihood-informed MCMC. *Journal of Computational Physics*, *304*, 109–137. https://doi.org/10.1016/j.jcp.2015.10.008

Davis, A. D., Marzouk, Y., Smith, A., & Pillai, N. (2021). *Rate-optimal refinement strategies for local approximation MCMC*. https://arxiv.org/abs/2006.00032

Detommaso, G., Cui, T., Spantini, A., Marzouk, Y., & Scheichl, R. (2018). *A Stein variational Newton method*. https://arxiv.org/abs/1806.03085

Dodwell, T., Ketelsen, C., Scheichl, R., & Teckentrup, A. (2019). Multilevel Markov chain Monte Carlo. *SIAM Review*, *61*, 509–545. https://doi.org/10.1137/19M126966X

Dodwell, T., Ketelsen, C., Scheichl, R., & Teckentrup, A. (2015). A hierarchical multilevel Markov chain Monte Carlo algorithm with applications to uncertainty quantification in

subsurface flow. *SIAM ASA Journal on Uncertainty Quantification*, *3*, 34 S. https://doi.org/10.1137/130915005

Giles, M. B. (2008). Multilevel Monte Carlo path simulation. *Operations Research*, *56*(3), 607–617. https://doi.org/10.1287/opre.1070.0496

Guennebaud, G., Jacob, B., & others. (2010). *Eigen v3*. http://eigen.tuxfamily.org.

Han, J., & Liu, Q. (2018). Stein variational gradient descent without gradient. In J. Dy & A. Krause (Eds.), *Proceedings of the 35th international conference on machine learning* (Vol. 80, pp. 1900–1908). PMLR. https://proceedings.mlr.press/v80/han18b.html

Hindmarsh, A. C., Brown, P. N., Grant, K. E., Lee, S. L., Serban, R., Shumaker, D. E., & Woodward, C. S. (2005). SUNDIALS: Suite of nonlinear and differential/algebraic equation solvers. *ACM Transactions on Mathematical Software (TOMS)*, *31*(3), 363–396. https://doi.org/10.1145/1089014.1089020

Jakob, W., Rhinelander, J., & Moldovan, D. (2017). *pybind11 – seamless operability between C++11 and Python*. https://github.com/pybind/pybind11.

Johnson, S. (2007). *The NLopt nonlinear-optimization package*. http://github.com/stevengj/nlopt.

Kennedy, M. C., & O'Hagan, A. (2001). Bayesian calibration of computer models. *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, *63*(3), 425–464. https://doi.org/10.1111/1467-9868.00294

Lao, J., Suter, C., Langmore, I., Chimisov, C., Saxena, A., Sountsov, P., Moore, D., Saurous, R. A., Hoffman, M. D., & Dillon, J. V. (2020). *Tfp.mcmc: Modern Markov chain Monte Carlo tools built for modern hardware*. https://arxiv.org/abs/2002.01184

Lunn, D., Spiegelhalter, D., Thomas, A., & Best, N. (2009). The BUGS project: Evolution, critique and future directions. *Statistics in Medicine*, *28*(25), 3049–3067. https://doi.org/10.1002/sim.3680

Parno, M. D., & Marzouk, Y. M. (2018). Transport map accelerated Markov chain Monte Carlo. *SIAM/ASA Journal on Uncertainty Quantification*, *6*(2), 645–682. https://doi.org/10.1137/17M1134640

Plummer, M., & others. (2003). JAGS: A program for analysis of Bayesian graphical models using Gibbs sampling. *Proceedings of the 3rd International Workshop on Distributed Statistical Computing*, *124*, 1–10. https://www.r-project.org/conferences/DSC-2003/Proceedings/

Salvatier, J., Wiecki, T. V., & Fonnesbeck, C. (2016). Probabilistic programming in Python using PyMC3. *PeerJ Computer Science*, *2*, e55. https://doi.org/10.7717/peerj-cs.55

Sargsyan, K., Huan, X., & Najm, H. N. (2019). Embedded model error representation for Bayesian model calibration. *International Journal for Uncertainty Quantification*, *9*(4). https://doi.org/10.1615/Int.J.UncertaintyQuantification.2019027384

Seelinger, L., Reinarz, A., Rannabauer, L., Bader, M., Bastian, P., & Scheichl, R. (2021). High performance uncertainty quantification with parallelized multilevel Markov chain Monte Carlo. *Proceedings of the International Conference for High Performance Computing, Networking, Storage and Analysis*. https://doi.org/10.1145/3458817.3476150