The Journal of Open Source Software

# UM-Bridge: Uncertainty quantification and modeling bridge

**Linus Seelinger** [1], **Vivian Cheng-Seelinger** [5], **Andrew Davis** [2], **Matthew Parno** [4], **and Anne Reinarz** [3]

**1** Institute for Applied Mathematics, Heidelberg University, Heidelberg, Germany **2** Courant Institute of Mathematical Sciences, New York Univeristy, New York, NY, USA **3** Department of Computer Science, Durham University, Durham, United Kingdom **4** Department of Mathematics, Dartmouth College, Hannover, NH, USA **5** Independent researcher

**DOI:** 10.xxxxxx/draft

**Software**

- Review
- Repository
- Archive

**Editor:** Pierre de Buyl

**Reviewers:**

- @georgiastuart
- @Himscipy

**Submitted:** 22 July 2022
**Published:** unpublished



## Summary

UM-Bridge (the uncertainty quantification (UQ) and modeling bridge) provides a unified interface for numerical models that is accessible from any programming language or framework. It is primarily intended for coupling advanced models (e.g. simulations of complex physical processes) to advanced statistical or optimization methods for UQ without requiring the model and UQ algorithms to share a common computational environment.

By allowing containerization, numerical models become portable and reproducible. This improves separation of concerns between the fields, and lays the groundwork for unified UQ benchmark problems. Further, high peformance computing is simplified through a pre-defined configuration for cloud environments, allowing to run many parallel instances of any UM-Bridge model container.

## Statement of need

Many uncertainty quantification and optimization methods treat a model as an abstract function $f : \mathbb{R}^n \mapsto \mathbb{R}^m$ and only interact with the model through some of the following operations: (i) model evaluation $y = f(x)$, (ii) gradient evaluation, (iii) Jacobian action, and/or (iv) Hessian action. Many UQ algorithms do not require knowledge of $f$ other than these abstract operations. Examples include Markov chain Monte Carlo methods (Metropolis et al., 1953; Seelinger et al., 2021), polynomial chaos (Najm, 2009), stochastic collocation (Marzouk & Xiu, 2009), optimal transport (Marzouk et al., 2016), and maximum likelihood estimation.

In theory, this abstraction allows the same UQ algorithm to be immediately applied on a wide range of problems. In practice however, UQ algorithms and models are often developed separately. Each implementation is typically done by experts in different fields,. Implementing interfaces between these (often incompatible) code bases tends to add considerable complexity, is time consuming, and sometimes requires completely re-implementing either UQ algorithm or model. This issue is exacerbated by the distributed or heterogeneous computing environments required by many advanced simulation codes.

The Journal of Open Source Software

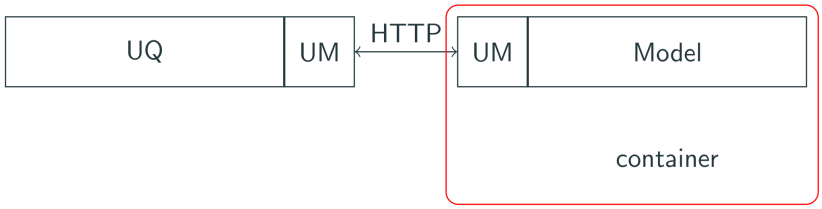


**Figure 1:** UM-Bridge architecture.

UM-Bridge addresses these issues by providing a universal, microservice-inspired software interface between models and UQ algorithms. At its core, UM-Bridge consists of an HTTP based protocol mirroring the mathematical interface above. Integrations for (currently) C++, R, Python, MUQ (Parno et al., 2021), QMCPy (Choi et al., 2020+) and PyMC (Salvatier et al., 2016) are provided for convenience. This approach has a number of benefits:

- Codes can be coupled across programming languages,
- Separation of concerns between developers is achieved since proficiency in only one side is needed to implement the interface,
- UM-Bridge is easy to integrate into many existing codes since they often (implicitly or explicitly) already implement a similar interface internally.

Further, due to being based on HTTP, containerization (Kurtzer et al., 2017; Merkel, 2014) of models becomes trivial, leading to:

- Portability across operating systems and vastly reduced setup cost when sharing models with collaborators,
- Fully reproducible models and benchmarks,
- Access to container based compute resources in the cloud (e.g. GCP, AWS).

UM-Bridge is the, to our knowledge, first universal model interface geared towards uncertainty quantification. Frameworks with somewhat similar architectures exist in related fields, for example preCICE (Chourdakis et al., 2022). However, their particular focus (coupling meshes across different numerical simulation codes, in case of preCICE) makes them less suitable for UQ.

## Current applications and future work

A library of UQ benchmarks and models based on UM-Bridge is currently being built here. To the best of our knowledge, this is the first UQ benchmark library available.

Further, support for running UM-Bridge models in cloud environments at large scale is being developed.

## Acknowledgements

We would like to acknowledge support from Robert Scheichl and Cristian Mezzanotte. Parno's effort was supported by Office of Naval Research MURI grant N00014-20-1-2595.

## References

Choi, S.-C. T., Hickernell, F. J., McCourt, M., & Sorokin, A. (2020+). *QMCPy: A quasi-Monte Carlo Python library*. https://github.com/QMCSoftware/QMCSoftware

Chourdakis, G., Davis, K., Rodenberg, B., Schulte, M., Simonis, F., Uekermann, B., Abrams, G., Bungartz, H., Cheung Yau, L., Desai, I., Eder, K., Hertrich, R., Lindner, F., Rusch, A.,

Sashko, D., Schneider, D., Totounferoush, A., Volland, D., Vollmer, P., & Koseomur, O. (2022). preCICE v2: A sustainable and user-friendly coupling library [version 2; peer review: 2 approved]. *Open Research Europe*, *2*(51). https://doi.org/10.12688/openreseurope.14445.2

Kurtzer, G. M., Sochat, V., & Bauer, M. W. (2017). Singularity: Scientific containers for mobility of compute. *PloS One*, *12*(5), e0177459–e0177459.

Marzouk, Y., Moselhy, T., Parno, M., & Spantini, A. (2016). *Sampling via measure transport: An introduction* (pp. 1–41). https://doi.org/10.1007/978-3-319-11259-6_23-1

Marzouk, Y., & Xiu, D. (2009). A stochastic collocation approach to bayesian inference in inverse problems. *PRISM: NNSA Center for Prediction of Reliability, Integrity and Survivability of Microsystems*, *6*. https://doi.org/10.4208/cicp.2009.v6.p826

Merkel, D. (2014). Docker: Lightweight linux containers for consistent development and deployment. *Linux Journal*, *2014*(239), 2.

Metropolis, N., Rosenbluth, A. W., Rosenbluth, M. N., Teller, A. H., & Teller, E. (1953). Equation of state calculations by fast computing machines. *The Journal of Chemical Physics*, *21*(6), 1087–1092. https://doi.org/10.1063/1.1699114

Najm, H. N. (2009). Uncertainty quantification and polynomial chaos techniques in computational fluid dynamics. *Annual Review of Fluid Mechanics*, *41*, 35–52. https://doi.org/10.1146/annurev.fluid.010908.165248

Parno, M., Davis, A., & Seelinger, L. (2021). MUQ: The MIT uncertainty quantification library. *Journal of Open Source Software*, *6*(68), 3076. https://doi.org/10.21105/joss.03076

Salvatier, J., Wiecki, T., & Fonnesbeck, C. (2016). *Probabilistic programming in python using PyMC3*. https://doi.org/10.7287/PEERJ.PREPRINTS.1686V1

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