UM-Bridge: Bridging the Gap Between UQ and Model Software

Anne Reinarz, Robert Scheichl, Andrew Davis, Matthew Parno, Vivian Cheng-Seelinger, **Linus Seelinger**

Institute for Applied Mathematics, Heidelberg University

Outline

Gap in current research: Advanced UQ methods not often coupled with advanced models.

Even when conceptually easy, software side is challenging...

This talk:

- HPC scale UQ on tsunami model
- MIT UQ Library (MUQ): Modular, general-purpose UQ framework
- UM-Bridge: Abstract interface between UQ and model codes
  $\rightarrow$ UQ benchmarks, UQ in the cloud

Outline

Gap in current research: Advanced UQ methods not often coupled with advanced models.

Even when conceptually easy, software side is challenging...

This talk:

- HPC scale UQ on tsunami model
- MIT UQ Library (MUQ): Modular, general-purpose UQ framework
- UM-Bridge: Abstract interface between UQ and model codes
  $\rightarrow$ UQ benchmarks, UQ in the cloud

People

Anne Reinarz
Durham University

Robert Scheichl
Heidelberg University

Andrew Davis
New York University

Matthew Parno
Dartmouth College

Vivian Cheng-Seelinger
Independent

Linus Seelinger
Heidelberg University

HPC Application: Multilevel MCMC on Tsunami Model

Multilevel MCMC Overview

Telescoping sum of QOI like MLMC:

$$\mathbb{E}_{\nu^L}[Q_L] = \underbrace{\mathbb{E}_{\nu^0}[Q_0]}_{\text{Coarse approx.}} + \sum_{l=1}^{L} \underbrace{\left(\mathbb{E}_{\nu^l}[Q_l] - \mathbb{E}_{\nu^{l-1}}[Q_{l-1}]\right)}_{\text{Corrections}}.$$

How to sample?

High acceptance rates due to samples from coarser levels

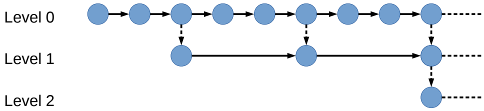
Coarser models cheap, low variance in finer corrections!

Multilevel MCMC Overview

Telescoping sum of QOI like MLMC:

$$\mathbb{E}_{\nu^L}[Q_L] = \underbrace{\mathbb{E}_{\nu^0}[Q_0]}_{\text{Coarse approx.}} + \sum_{l=1}^{L} \underbrace{\left(\mathbb{E}_{\nu^l}[Q_l] - \mathbb{E}_{\nu^{l-1}}[Q_{l-1}]\right)}_{\text{Corrections}}.$$

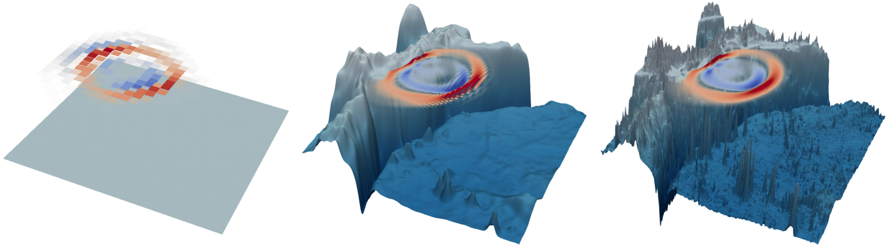
How to sample?



High acceptance rates due to samples from coarser levels

Coarser models cheap, low variance in finer corrections!

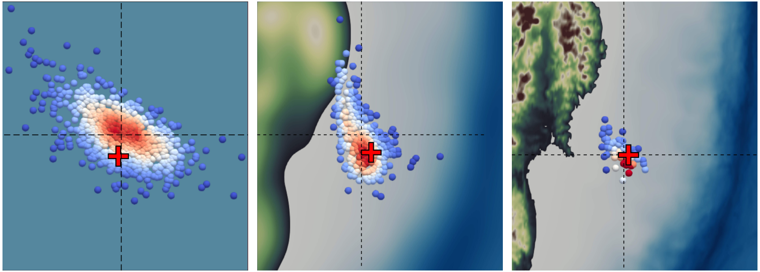
Model hierarchy



Across levels, we adapt

- mesh size
- bathymetry smoothness (specific to hyperbolic solvers!)

Results

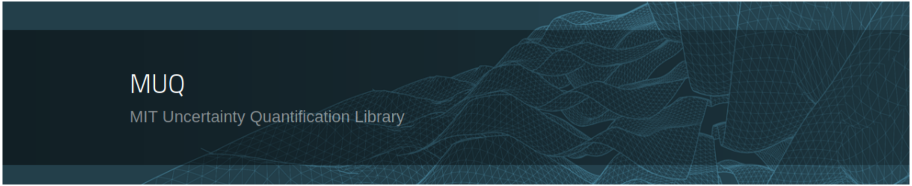


lvl l	t_l[s]	ρ_l	$\mathbb{V}[Q_0]$ or $\mathbb{V}[Q_l - Q_{l-1}]$		$\mathbb{E}[Q_0]+$ $\sum_{k=1}^{l} \mathbb{E}[Q_k - Q_{k-1}]$	
0	7.38	25	1984.09	1337.42	3.61	27.96
1	97.3	5	1592.17	1523.18	-12.29	23.39
2	438.1	0	340.56	938.53	-5.46	0.12

Run on 3456 cores (72 nodes of 48 cores)

Behind the stages: MIT UQ Library (MUQ)

MUQ - MIT Uncertainty Quantification Library



- Modular UQ Library, C++ and Python[1]
- Model-agnostic interfaces
- Numerous UQ algorithms readily available
- www.mituq.bitbucket.io

[1] Matthew Parno, Andrew Davis, and Linus Seelinger. "MUQ: The MIT Uncertainty Quantification Library". In: *Journal of Open Source Software* 6.68 (2021), p. 3076. DOI: 10.21105/joss.03076. URL: https://doi.org/10.21105/joss.03076.

Models in MUQ

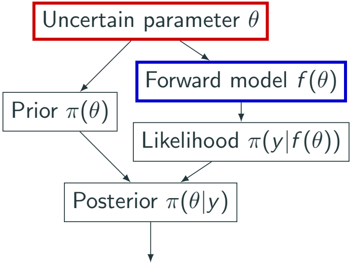


Figure 1: Model graph example for simple Bayesian problem

Well-structured, modular construction of advanced models.

Can couple to external software.

Easy to switch models / methods!

For ML / MI: Define set of models.

Modular MCMC Framework

- Proposals: MH, pCN, MALA, DILI, ...
- Kernels: MH, MC, ML/MI, ...
- Chains: Sequential, parallel, ...

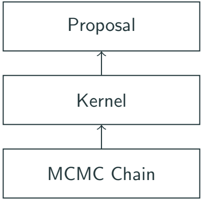


Figure 2: Modular MCMC architecture

Want a different method? Just switch out one component!

Parallel ML / MI MC / MCMC Processor Layout

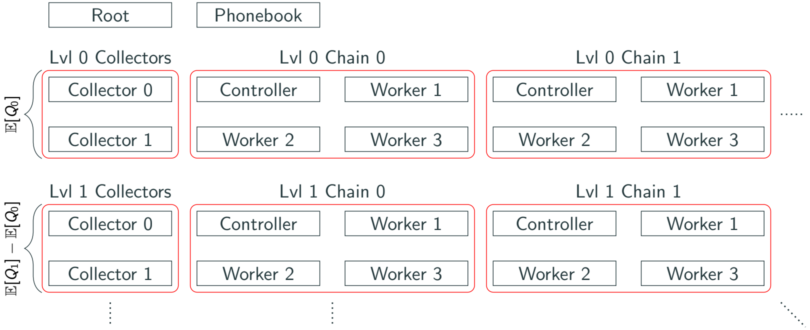


Figure 3: Parallel layout (each box is a processor / MPI rank)

Tested on 2048 parallel chains, more is possible

Too complicated? All this happens behind the scenes!

Weak scaling

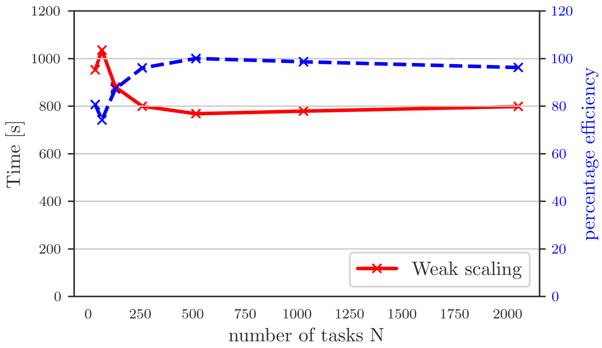


Figure 4: Weak scalability and parallel efficiency of the Poisson model problem. At 64 cores 10^4, 10^3 and 10^2 samples are computed on levels 0, 1 and 2 respectively. The number of samples is modifed linearly with the number of processors.

Reality strikes...

UQ and Model in Math

Model in UQ: (Often) Just a function $F : \mathbb{R}^n \to \mathbb{R}^m$ with

- Model evaluation $F(\theta)$,
- Jacobian action $J(\theta)v$,
- Hessian action $H(\theta)v$.

$\rightarrow$ Simple, model-agnostic interface!

Model **software** and UQ **software**: Not so easy!

Conflicts in buildsystems, dependencies, languages, parallelization; need experts from both sides, ...

UQ and Model in Math

Model in UQ: (Often) Just a function $F : \mathbb{R}^n \to \mathbb{R}^m$ with

- Model evaluation $F(\theta)$,
- Jacobian action $J(\theta)v$,
- Hessian action $H(\theta)v$.

$\to$ Simple, model-agnostic interface!

Model **software** and UQ **software**: Not so easy!

Conflicts in buildsystems, dependencies, languages, parallelization; need experts from both sides, ...

UM-Bridge: Model abstraction in software

UM-Bridge: Model Abstraction in Software

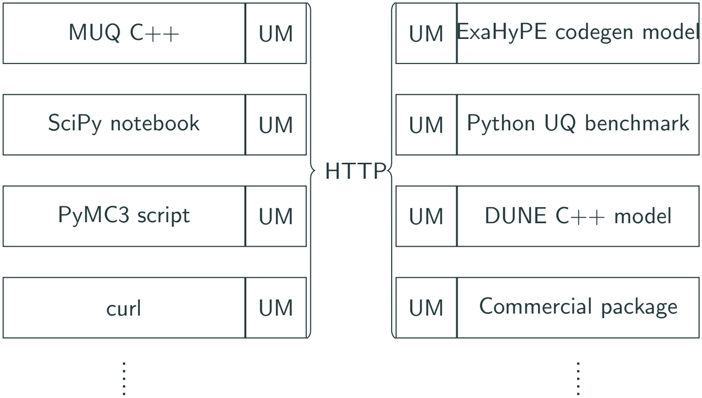
Established approach in large-scale web services

Requires: Minimal extension of software on each side

Achieves:

- Coupling across languages / frameworks
- Separation of concerns between UQ and model experts
- Containers: Portable, reproducible models

UM-Bridge: Bridging Languages and Frameworks



Requires only HTTP and JSON support $\rightarrow$ almost every language

Full integration for C++, Python, MUQ

Python client interface

Connect to model

```
import umbridge

model = umbridge.HTTPModel("http://localhost:4242")
```

Display input / output dimensions

```
print(model.get_input_sizes())
print(model.get_output_sizes())
```

Evaluate model

```
print(model([[0.0, 10.0]]))
```

Optionally, pass configuration options

```
print(model([[0.0, 10.0]], {"level": 0}))
```

Python server interface

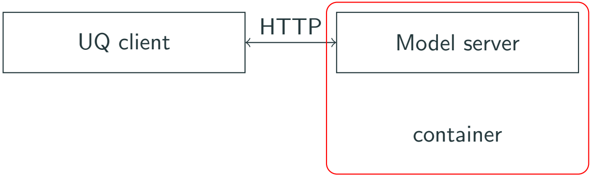
Define model

```python
class TestModel(umbridge.Model):

    def get_input_sizes(self): # Number and dimensions of input vectors
        return [1]

    def get_output_sizes(self): # Number and dimensions of output vectors
        return [1]

    def __call__(self, parameters, config={}):
        output = parameters[0][0] * 2 # Do something with the input
        return [[output]]
```

Serve model via HTTP

```python
testmodel = TestModel()

umbridge.serve_model(testmodel, 4242)
```

UM-Bridge: Containerization - Portable Models



- Run tsunami model as easy as

```
docker run -p 4242:4242 linusseelinger/model-exahype-tsunami
```

- Evaluate model in python:

```
model = umbridge.HTTPModel('localhost:4242')
model([[0.1,0.4]])
```

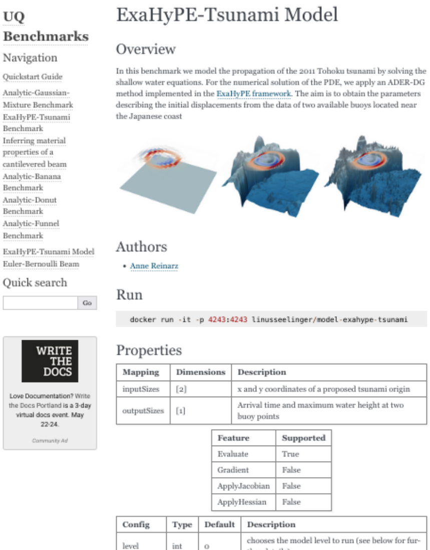
Software Defined UQ Benchmarks

UQ Benchmarks

- Previous ESI workshop '20: Great interest, but technical side lacking
- Now: Easy to use, software defined, reproducible benchmarks possible

$\rightarrow$ Proposal: Community defined benchmark suite

Current state



UQ Benchmarks

Navigation

Quickstart Guide
Analytic-Gaussian-Mixture Benchmark
ExaHyPE-Tsunami Benchmark
Inferring material properties of a cantilevered beam
Analytic-Banana Benchmark
Analytic-Donut Benchmark
Analytic-Funnel Benchmark
ExaHyPE-Tsunami Model
Euler-Bernoulli Beam

Quick search

Go

WRITE THE DOCS

Love Documentation? Write the Docs Portland is a 3-day virtual docs event. May 22-24.

Community Ad

ExaHyPE-Tsunami Model

Overview

In this benchmark we model the propagation of the 2011 Tohoku tsunami by solving the shallow water equations. For the numerical solution of the PDE, we apply an ADER-DG method implemented in the ExaHyPE framework. The aim is to obtain the parameters describing the initial displacements from the data of two available buoys located near the Japanese coast

Authors

- Anne Reinarz

Run

```
docker run -it -p 4243:4243 linusseelinger/model-exahype-tsunami
```

Properties

Mapping	Dimensions	Description
inputSizes	[2]	x and y coordinates of a proposed tsunami origin
outputSizes	[1]	Arrival time and maximum water height at two buoy points

Feature	Supported
Evaluate	True
Gradient	False
ApplyJacobian	False
ApplyHessian	False

Config	Type	Default	Description
level	int	0	chooses the model level to run (see below for fur-

- Several models, Bayesian posteriors, analytic densities
- Automated testing and building
- Partially-automated documentation

UQ in the Cloud

Overview: Cloud Infrastructure

Infrastructure hierarchy

- Application
- Container
- Virtual machine
- Server hardware
- Network

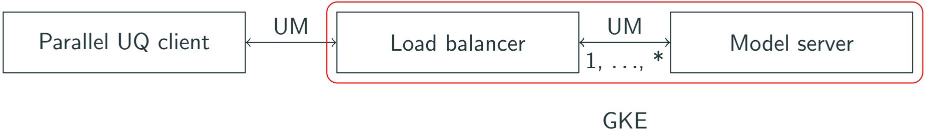
Google Cloud Platform

- Cloud Functions
- Kubernetes Engine
- Compute engine

Servers for rent, (very) different levels of abstraction possible

Kubernetes: "Container orchestration" - fully reproducible HPC setups

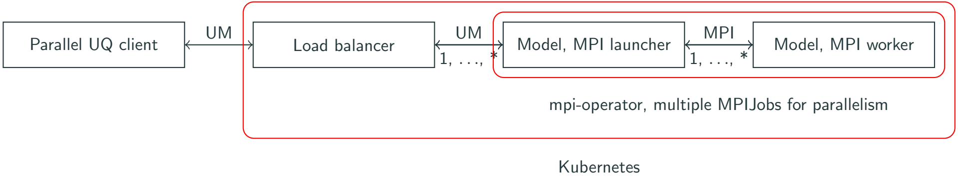
Kubernetes Setup - Sequential Model



Pre-built reference setup, simply plug in your own model container

No modification to model needed!

Kubernetes Setup - MPI Parallel Model



Pre-built reference setup

Shared filesystem between nodes via NFS

Support for OpenMPI, Intel MPI, possibly MPICH in the future

Minor restrictions on model container

Conclusions

Conclusions / Outlook

- UM-Bridge: UQ / model abstraction in software
- Can build UQ benchmark suite now
  $\rightarrow$ Community project!

Resources / Contact:

- MUQ Website: www.mituq.bitbucket.io
- UM-Bridge: www.github.com/UM-Bridge/umbridge
- Initial benchmarks: www.github.com/UM-Bridge/benchmarks

Publications:

- MUQ: The MIT Uncertainty Quantification Library
  M. Parno, A. Davis, L. Seelinger, Journal of Open Source Software, 2021
- High Performance Uncertainty Quantification with Parallelized Multilevel Markov Chain Monte Carlo
  L. Seelinger, A. Reinarz, L. Rannabauer, M. Bader, P. Bastian, R. Scheichl, The International Conference for High Performance Computing, Networking, Storage, and Analysis 2021