UM-Bridge Workshop 2022

The interface between UQ and models

Anne Reinarz (Durham University, UK)
Linus Seelinger (Heidelberg University, Germany)

Goals

Teach how to use UM-Bridge

Initiate new collaborations between UQ and model experts

Advance the UQ software ecosystem through common interfaces, unified benchmarks and access to cloud-based HPC

Content

- Talks presenting aspects of UM-Bridge
- Project sessions, (optional) practical exercises
- Invited talks about current projects
- Discussions

Locations

- Lecture hall: Talks
- Breakout room: Practical sessions, discussions
- Lobby: Breaks, socializing

Private areas and spotlight areas control visibility

Platform sponsor: digiLab Solutions

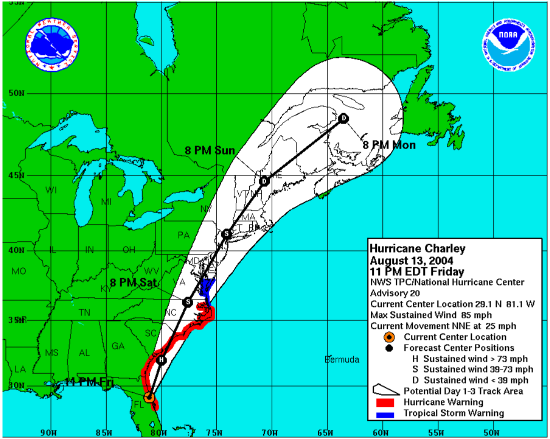
digiLab

- First of a kind solutions to solve the biggest challenges of today
- Closing the skills gap by training people to deploy "AI in the wild"
- 4-day week company based in the southwest of the UK

→ `https://www.digilab.co.uk/`

Intro

Why Uncertainty Quantification (UQ)?



- "Don't focus on the skinny black line"
  - US Hurricane Center
- Uncertain data $\Longrightarrow$ uncertain prediction / inferences.

  UQ: Quantify this!

Inverse Problems

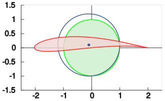
Given observations, how likely is a parameter? And what quantity of interest is therefore likely?

$$\underbrace{y}_{\text{Observation}}$$

Inverse Problems

Given observations, how likely is a parameter? And what quantity of interest is therefore likely?

$$\underbrace{F^{-1}(y)}_{\text{Inverse model}} \leftarrow \underbrace{y}_{\text{Observation}}$$



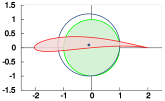
$\leftarrow$

Inverse Problems

Given observations, how likely is a parameter? And what quantity of interest is therefore likely?

$$\underbrace{Q(F^{-1}(y))}_{\text{Quantity of interest}} \leftarrow \underbrace{F^{-1}(y)}_{\text{Inverse model}} \leftarrow \underbrace{y}_{\text{Observation}}$$

? $\leftarrow$



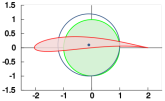
$\leftarrow$

Inverse Problems

Given observations, how likely is a parameter? And what quantity of interest is therefore likely?

$$\underbrace{Q(F^{-1}(y))}_{\text{Quantity of interest}} \leftarrow \underbrace{F^{-1}(y)}_{\text{Inverse model}} \leftarrow \underbrace{y}_{\text{Observation}}$$

? $\leftarrow$



$\leftarrow$

Cannot invert F, therefore Bayesian problem!

Bayesian Inverse Problems

By Bayes' rule, probability of parameter θ given data y is

$$\pi(\theta) := \pi_{post}(\theta|y) \propto \pi_{prior}(\theta)\pi_{likelihood}(y|\theta),$$

where likelihood encodes trust in data. For normal dist. observation error with covariance matrix Σ_{obs}:

$$\pi_{likelihood}(y|\theta) \propto \exp(-\frac{1}{2}(F(\theta) - y)^\top \Sigma_{obs}^{-1}(F(\theta) - y)).$$

Forward model F enters here!

Algorithm 1 Metropolis-Hastings Markov Chain Monte Carlo

Input: Starting point $\theta_0 \in \mathbb{R}^n$, density π, parameter Σ, sample num. M.
Output: Sequence of samples $\{\theta_k\}_{k=0}^{N}$ approximating distribution of π.
for $m \leftarrow 1$ to M **do**
Draw $\tilde{\theta}$ from $q(\cdot|\theta^{m-1}) \sim \mathcal{N}(\theta^{m-1}, \Sigma)$
With probability $\alpha = \min\left\{1, \frac{\pi(\tilde{\theta})q(\theta^{m-1}|\tilde{\theta})}{\pi(\theta^{m-1})q(\tilde{\theta}|\theta^{m-1})}\right\}$, do $\theta^m \leftarrow \tilde{\theta}$;
else $\theta^m \leftarrow \theta^{m-1}$
end for

Algorithm 2 Hamiltonian Monte Carlo

Input: Starting point $\theta_0 \in \mathbb{R}^n$, density $\mathcal{L}(\theta) := \log(\pi(\theta))$, parameters L and ϵ, M.
Output: Sequence of samples $\{\theta_k\}_{k=0}^N$ approximating distribution of π.

```
for m ← 1 to M do
    Sample r^0 ~ N(0, I)
    θ^m ← θ^{m−1}, θ̃ ← θ^{m−1}, r̃ ← r^0
    for i ← 1 to L do
        θ̃, r̃ ← Leapfrog(θ̃, r̃, ε)
    end for
    With probability α = min{1, exp(L(θ̃) − ½ r̃·r̃) / exp(L(θ^{m−1}) − ½ r^0·r^0)}, do θ^m ← θ̃, r^m ← −r̃
end for

function LEAPFROG(θ, r, ε)
    r̃ ← r + (ε/2)∇_θ L(θ)
    θ̃ ← θ + ε r̃
    r̃ ← r̃ + (ε/2)∇_θ L(θ)
    return θ̃, r̃
end function
```

With probability $\alpha = \min\left\{1, \frac{\exp(\mathcal{L}(\tilde{\theta}) - \frac{1}{2}\tilde{r}\cdot\tilde{r})}{\exp(\mathcal{L}(\theta^{m-1}) - \frac{1}{2}r^0\cdot r^0)}\right\}$, do $\theta^m \leftarrow \tilde{\theta}$, $r^m \leftarrow -\tilde{r}$

Newton's Method

Algorithm 3 Newton's method for optimization

Input: Starting point $x_0 \in \mathbb{R}^n$, density π.

Output: Approx. local minimum x_N.

for $k \leftarrow 0$ to $N - 1$ **do**

$x_{k+1} \leftarrow x_k - H_\pi^{-1}(x_k)\nabla\pi(x_k)$

end for

UQ and Model in Math

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

- Model evaluation $F(\theta)$,
- Gradient $v^\top J(\theta)$,
- 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, ...

UQ and Model in Math

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

- Model evaluation $F(\theta)$,
- Gradient $v^\top J(\theta)$,
- 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: Abstract interface in software

UM-Bridge: Model Abstraction in Software

Interface mimics math

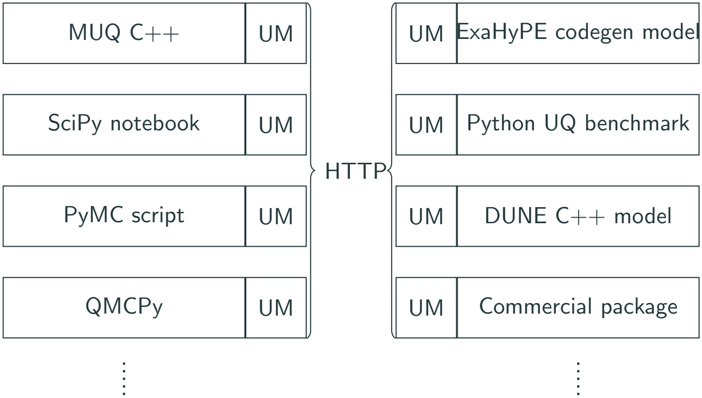
Established approach in large-scale web services ($\rightarrow$ microservices)

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 → almost every language

Full integrations for various languages and frameworks

Supported languages

Language / framework	Client support	Server support
C++	√	√
MATLAB	planned	✗
Python	√	√
R	√	✗
MUQ	√	√
PyMC (4.x)	√	✗
QMCPy	√	✗
Sparse Grids MATLAB Kit	planned	✗
tinyDA	√	✗

Python client interface

Connect to model

```python
import umbridge

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

Display input / output dimensions

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

Evaluate model

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

Optionally, pass configuration options

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

C++ client interface

Connect to model

```cpp
umbridge::HTTPModel model("http://localhost:4242", "forward");
```

Display input / output dimensions

```cpp
std::cout << model.GetInputSizes() << std::endl;
std::cout << model.GetOutputSizes() << std::endl;
```

Evaluate model

```cpp
  std::vector<std::vector<double>> outputs
    = model.Evaluate({{100.0, 18.0}});
```

Optionally, pass configuration options

```cpp
json config;
config["level"] = 0;
model.Evaluate(input, config);
```

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]]

    def supports_evaluate(self):
        return True
```

Serve model via HTTP

```python
testmodel = TestModel()

umbridge.serve_model(testmodel, 4242)
```

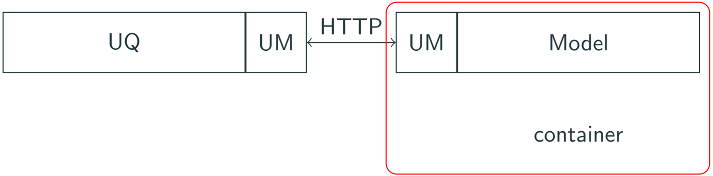
Demo

Demo

UM-Bridge Demo

Upcoming topics

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', 'forward')
model([[0.1,0.4]])
```

UM-Bridge: UQ Benchmarks

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

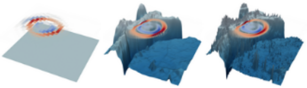
Quick search

Go

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-

- Portable, reproducible models and UQ problems
- Several models, Bayesian posteriors, analytic densities
- Automated testing and building
- Partially-automated documentation

Overview: Cloud Infrastructure

Infrastructure hierarchy	Google Cloud Platform
Application	Cloud Functions
Container	Kubernetes Engine
Virtual machine	Compute engine
Server hardware	
Network	

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

Kubernetes: "Container orchestration" - fully reproducible HPC setups

Conclusions

Conclusions

- Universal UQ / model interface, following maths
- Easy to use in various languages and frameworks
- Opens up new possibilities for UQ:
    - Portable models, separation of concerns via containers
    - Library of reproducible UQ benchmark problems
    - Easy scaling to HPC in the cloud