Session 1

Practical Bayesian Inference using MCMC

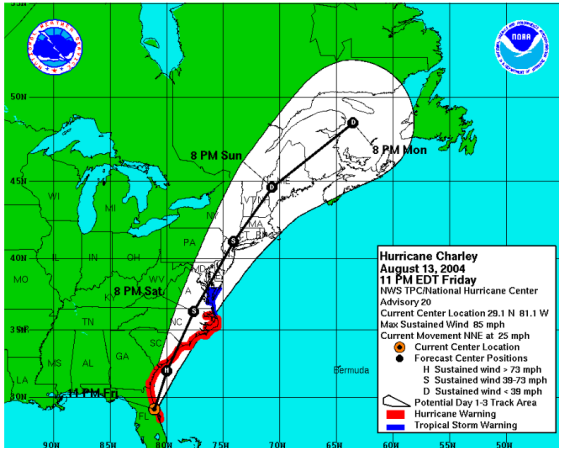
Anne Reinarz

Durham University

Overview

- **Session 1:** Introduction
  - Introduction to UM-Bridge
  - Hands-on exercises
  - Inverse Problems
- Session 2: Markov Chain Monte-Carlo
- Session 3: Scaling up and hierarchical models

Why Uncertainty Quantification (UQ)?



- "Don't focus on the skinny black line"
  - US Hurricane Center
- Uncertain data leads to uncertain prediction / inferences.
  $\Rightarrow$ Quantify this!

What methods are out there?

Many methods:

- Markov Chain Monte Carlo
  - Really an entire zoo of methods. *Session 2*
- Optimization-based MAP point search
- Multilevel / Multiindex *Session 3*
- Particle methods, filtering...
- ...

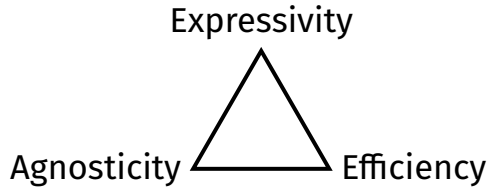


Figure: Main aspects of UQ methods

Practical Uncertainty Quantification requires software

UQ and Model in Math

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

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

$\to$ Simple, model-agnostic interface!

Example 1: Monte Carlo Methods

Estimate statistics of a Quantity of Interest (QoI)

$$Q = F(\theta), \quad \theta \sim \pi(\theta)$$

by independent sampling.

- Non-intrusive
- Embarrassingly parallel
- Often used for validation

R. Y. Rubinstein and D. P. Kroese, *Simulation and the Monte Carlo Method*, Wiley, 2016.

```
def monte_carlo(F, sampler, N):
    samples = []
    for i in range(N):
        theta = sampler()
        qoi = F(theta)
        samples.append(qoi)

    mean = average(samples)
    var  = variance(samples)
    return mean, var
```

Example 2: Linear Regression

Given observations $y \in \mathbb{R}^m$, estimate

$$\theta^* = \arg\min_\theta \frac{1}{2}\|F(\theta) - y\|^2$$

The gradient of the misfit is

$$\nabla_\theta \Phi(\theta) = J(\theta)^T (F(\theta) - y)$$

Only $F(\theta)$ and Jacobian actions are required.

```
def loss(theta):
    r = F(theta) - y
    return 0.5 * dot(r, r)

def grad_loss(theta):
    r = F(theta) - y
    return J(theta).T @ r

theta = theta0
for k in range(max_iter):
    theta -= alpha *
                  grad_loss(theta)
```

Example 3: Using Hessian Actions

Define the MAP estimator

$$\theta^* = \arg\min_\theta \frac{1}{2}\|F(\theta) - y\|^2 + \frac{1}{2}\|\theta - \theta_0\|^2_{\Gamma^{-1}}$$

Newton-type methods require

$$H(\theta)\, p = -\nabla\Phi(\theta)$$

Tan Bui-Thanh and Omar Ghattas, A scalable algorithm for MAP estimators in Bayesian inverse problems with Besov priors, *Inverse Problems and Imaging*, 2015.

```
def grad(theta):
    r = F(theta) - y
    return J(theta).T @ r
     + Gamma_inv @ (theta - theta0)

def Hv(theta, v):
    return J(theta).T @ (J(theta) @ v)
                        + Gamma_inv @ v

p = solve_cg(lambda v: Hv(theta, v),
                  -grad(theta))
theta += p
```

What Does Not Fit This Interface?

Intrusive methods which require rewriting the equation

Examples: Polynomial Chaos, Stochastic Galerkin, some reducted order modeling techniques

- Requires rewriting the forward model
- No black-box map $F(\theta)$
- Cannot be expressed via function or derivative calls

UQ and Model in Math

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

- Model evaluation $F(\theta)$,
- Gradient evaluation
- 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, ...

A (de)motivating example: MLMCMC

- Both the MLMCMC and the numerical solver were written in C++ and both were parallelised in MPI. In principle this should make coupling the codes straight-forward

A (de)motivating example: MLMCMC

- Both the MLMCMC and the numerical solver were written in C++ and both were parallelised in MPI. In principle this should make coupling the codes straight-forward
- *However*, the numerical solver was most efficient when used with a hybrid MPI+TBB parallelisation and this did not interact well with the MCMC implementation
- *and* they had different build systems, which needed to be modified to work together

A (de)motivating example: MLMCMC

- Both the MLMCMC and the numerical solver were written in C++ and both were parallelised in MPI. In principle this should make coupling the codes straight-forward
- *However*, the numerical solver was most efficient when used with a hybrid MPI+TBB parallelisation and this did not interact well with the MCMC implementation
- *and* they had different build systems, which needed to be modified to work together
- It took months of intrusive changes to the solver backend to get the two codes to work together
- The result is an interface that can be used only for this application (or slight variations thereof)

UQ and Model in Math

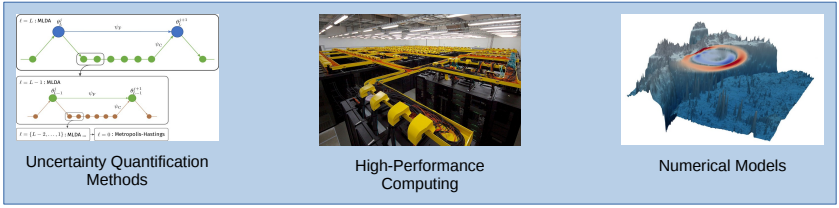


Figure: Monolithic coupling between model and UQ.

Model **software** and UQ **software**: Not so easy!
Conflicts in buildsystems, dependencies, languages, parallelization; need experts from both sides, ...

UQ and Model in Math

Figure: Lightweight coupling between model and UQ.

UM-Bridge: Model Abstraction in Software

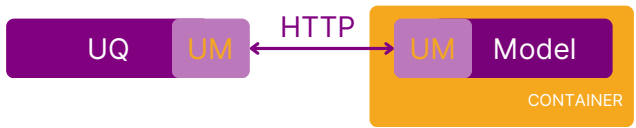


Figure: Network and container based coupling of UQ and model codes that requires minimal extension of software on each side.

UM-Bridge: The Goals

UM-Bridge is a universal interface that makes any numerical models accessible to UQ methods

- seamlessly link numerical models to UQ across languages and frameworks,
- break down complexity of advanced applications into manageable components,
- share your containerized models with collaborators,
- benchmark your algorithms on a community-driven benchmark library, and
- scale your applications from laptop to supercomputer or cloud with minimal effort.

UM-Bridge: Bridging Languages

Language / framework	Client support	Server support
C++	✓	✓
MATLAB	✓	✗
Python	✓	✓
R	✓	planned
julia	✓	✓

UM-Bridge: Bridging Frameworks

Language / framework	Client support	Server support
emcee	✓	✗
MUQ	✓	✓
PyMC	✓	✗
QMCPy	✓	✗
Sparse Grids MATLAB Kit	✓	✗
tinyDA	✓	✗
TT Toolbox	✓	✗

Running a first model

We can connect to dockerhub and pull a pre-configured model (or we can build our own)

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

UM-Bridge: Client

Connect to model

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

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}))
```

UM-Bridge: Server

Define model 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
```

UM-Bridge: Server

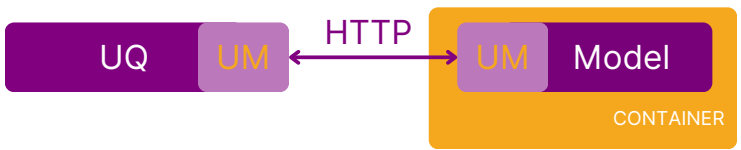
Define model class

```
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
```

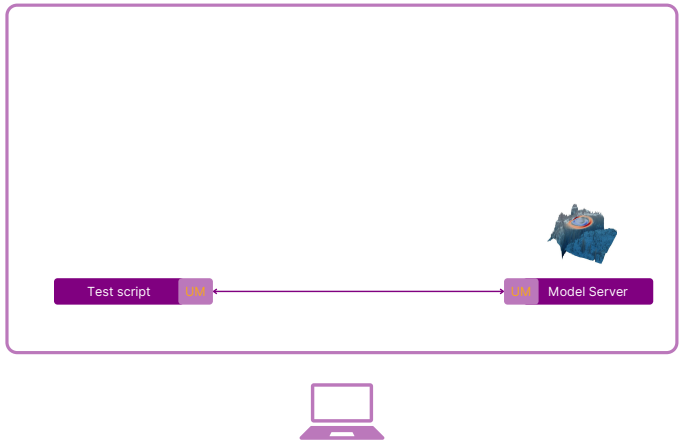
Serve model via HTTP:

```
testmodel = TestModel( )
umbridge.servemodel(testmodel , 4242)
```

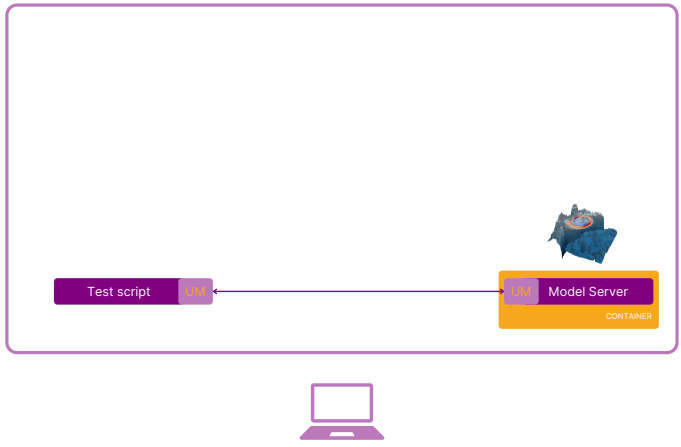
Workflows



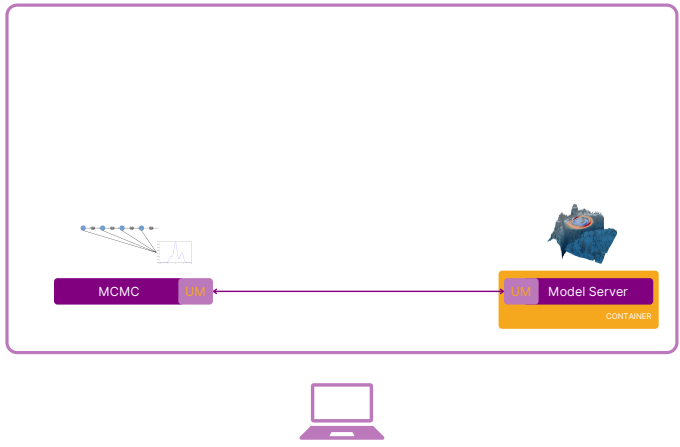
Workflow



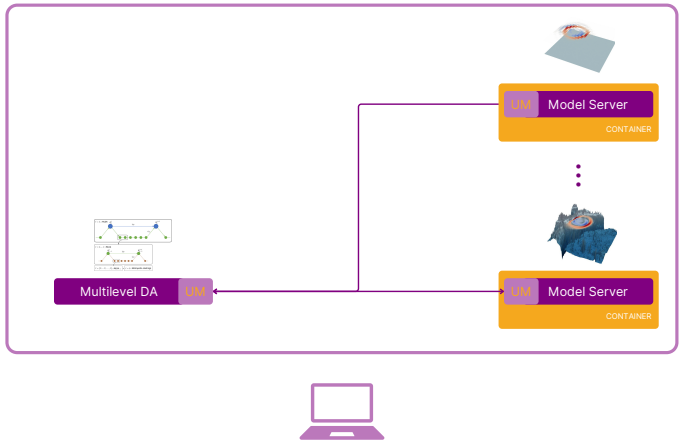
Workflow



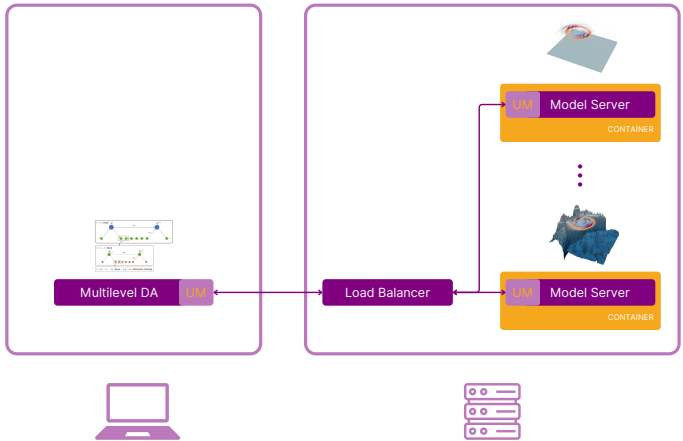
Workflow



Workflow


UM
Model Server
CONTAINER
UM
Model Server
CONTAINER
Multilevel DA
UM

Workflow


UM
Model Server
CONTAINER
Multilevel DA
UM
Load Balancer
UM
Model Server
CONTAINER

Hands-on

Over to you

Tutorial section 1, 2 and 3 up to Uncertainty propagation - Euler-Bernoulli beam

`https://um-bridge-benchmarks.readthedocs.io/en/docs/tutorial.html`

- Talk to each other
- Ask questions

Inverse Problems

Inverse problems

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

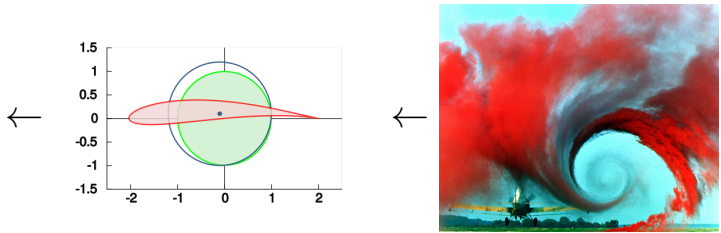
$\leftarrow \underbrace{y}_{\text{Observation distribution}}$

$\leftarrow$ 

Inverse problems

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

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



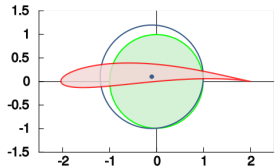
Inverse problems

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

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

?

$\leftarrow$



$\leftarrow$

Inverting for F

Here we assume F was an expensive model, usually the solution to a PDE

$$F : X \to Y$$

We could try to formally solve

$$\theta = F^{-1}(y).$$

Inverting for F

Inverse problems of this form are generally *Hadamard ill-posed*, e.g. at least one of the following does not hold

Inverting for F

Inverse problems of this form are generally *Hadamard ill-posed*, e.g. at least one of the following does not hold

1. *Existence*: for noisy data y there may be no x such that $F(\theta) = y$

Inverting for F

Inverse problems of this form are generally *Hadamard ill-posed*, e.g. at least one of the following does not hold

1. *Existence*: for noisy data y there may be no x such that $F(\theta) = y$
2. *Uniqueness*

Inverting for F

Inverse problems of this form are generally *Hadamard ill-posed*, e.g. at least one of the following does not hold

1. *Existence*: for noisy data y there may be no x such that $F(\theta) = y$
2. *Uniqueness*
3. *Stability*: the inverse map F^{-1} is unlikely to be continuous, so arbitrarily small perturbations in y can lead to arbitrarily large changes in θ.

Inverting for F

Example: If the PDE forward operators is smoothing, high-frequency information is ”destroyed” and F^{-1} is unstable or undefined.

Inverting for F

Example: If the PDE forward operators is smoothing, high-frequency information is ”destroyed” and F^{-1} is unstable or undefined.

$\Rightarrow$ Direct inversion of F is generally neither mathematically nor computationally feasible.

Optimization-based solution

Instead of inverting F, formulate the inverse problem as an optimization problem.

Given noisy observations

$$y = F(\theta) + \eta,$$

find

$$\theta^* = \arg\min_{\theta \in X} \left\{ \frac{1}{2} \|F(\theta) - y\|_Y^2 + \alpha R(\theta) \right\}.$$

$R(\theta)$ encodes prior assumptions on θ and $\alpha > 0$ balances data fit and regularity.

Optimization-based solution

- Solution requires repeated evaluations of $F(\theta)$ and its sensitivities, e.g. using adjoint PDE methods.
- Regularization parameters must be chosen
- Provide a single “best” estimate
- Well-suited when uncertainty quantification is not required, e.g. when noise is small and model confidence is high

The Bayesian inverse problem

Instead of inverting F (often impossible or very expensive), we use Bayes theorem to find

$$P(\theta|y),$$

the inferred parameter distribution given y.

This is the probability of a given parameter θ given our observations.

The Bayesian inverse problem

$$\mathcal{P}(\theta|\mathbf{y}) := \frac{\mathcal{L}(\mathbf{y}|\theta)\pi(\theta)}{\mathcal{P}(\mathbf{y})} \propto \mathcal{L}(\mathbf{y}|\theta)\pi(\theta).$$

The Bayesian inverse problem

$$\mathcal{P}(\theta|y) := \frac{\mathcal{L}(y|\theta)\pi(\theta)}{\mathcal{P}(y)} \propto \mathcal{L}(y|\theta)\pi(\theta).$$

- The likelihood $\mathcal{L}(y|\theta)$ is the probability of observing the given measurements if θ were the true parameter

The Bayesian inverse problem

$$\mathcal{P}(\theta|y) := \frac{\mathcal{L}(y|\theta)\pi(\theta)}{\mathcal{P}(y)} \propto \mathcal{L}(y|\theta)\pi(\theta).$$

- The likelihood $\mathcal{L}(y|\theta)$ is the probability of observing the given measurements if θ were the true parameter
- $\pi(\theta)$ is the prior distribution encoding a-priori knowledge about the inferred parameters

The Bayesian inverse problem

$$\mathcal{P}(\theta|\mathbf{y}) := \frac{\mathcal{L}(\mathbf{y}|\theta)\pi(\theta)}{\mathcal{P}(\mathbf{y})} \propto \mathcal{L}(\mathbf{y}|\theta)\pi(\theta).$$

- The likelihood $\mathcal{L}(\mathbf{y}|\theta)$ is the probability of observing the given measurements if θ were the true parameter
- $\pi(\theta)$ is the prior distribution encoding a-priori knowledge about the inferred parameters
- the distribution of measurements $\mathcal{P}(\mathbf{y})$ is treated as an unobtainable scaling factor

The Bayesian Inverse problem

- The final target is usually the mean of some Quantity of Interest with respect to the posterior distribution.

The Bayesian Inverse problem

- The final target is usually the mean of some Quantity of Interest with respect to the posterior distribution.
- Evaluating the likelihood typically means comparing the model prediction $F(\theta)$ to the measurements y.

The Bayesian Inverse problem

- The final target is usually the mean of some Quantity of Interest with respect to the posterior distribution.
- Evaluating the likelihood typically means comparing the model prediction $F(\theta)$ to the measurements y.
- Assuming Gaussian measurement errors with covariance Σ_F, the likelihood will have the distribution

$$\mathcal{N}(F(\theta), \Sigma_F)$$

The Bayesian Inverse problem

- The final target is usually the mean of some Quantity of Interest with respect to the posterior distribution.
- Evaluating the likelihood typically means comparing the model prediction $F(\theta)$ to the measurements y.
- Assuming Gaussian measurement errors with covariance Σ_F, the likelihood will have the distribution

$$\mathcal{N}(F(\theta), \Sigma_F)$$

- In practice evaluating $F(\theta)$ means solving an expensive PDE $\Leftarrow$ Main cost of solving the inverse problem

The Bayesian Inverse problem

Deterministic	Bayesian analogue
Solution exists	Posterior measure exists
Solution is unique	Posterior measure is unique
Stable solution	Posterior depends continuously on the data

- AM Stuart. *Inverse problems: A Bayesian perspective*. Acta Numerica., 1–101, 2010.

Next session

- Look at Markov Chain Monte Carlo as a tool to solve bayesian inverse problems
- Try it out using UM-Bridge models

References

- L. Seelinger, V. Cheng-Seelinger, A. Davis, M. Parno, A. Reinarz.
  UM-Bridge: Uncertainty quantification and modeling bridge.
  Journal of Open Source Software, 8(83), 2023.
- L. Seelinger, A. Reinarz et al.
  Democratizing uncertainty quantification.
  Journal of Computational Physics, 521:113542, 2024.
- C.M. Loi, A. Reinarz, L. Seelinger, W. Hornsby, J. Buchanan, M. Lykkegaard.
  A Performance Analysis of Task Scheduling for UQ Workflows on HPC Systems.
  ISC High Performance 2025 Research Paper Proceedings.