Introduction to Uncertainty Quantification

Natalie M. Isenberg
Computational Science Initiative
Applied Mathematics
Brookhaven National Laboratory

November 1st, 2022

Why Uncertainty Quantification?

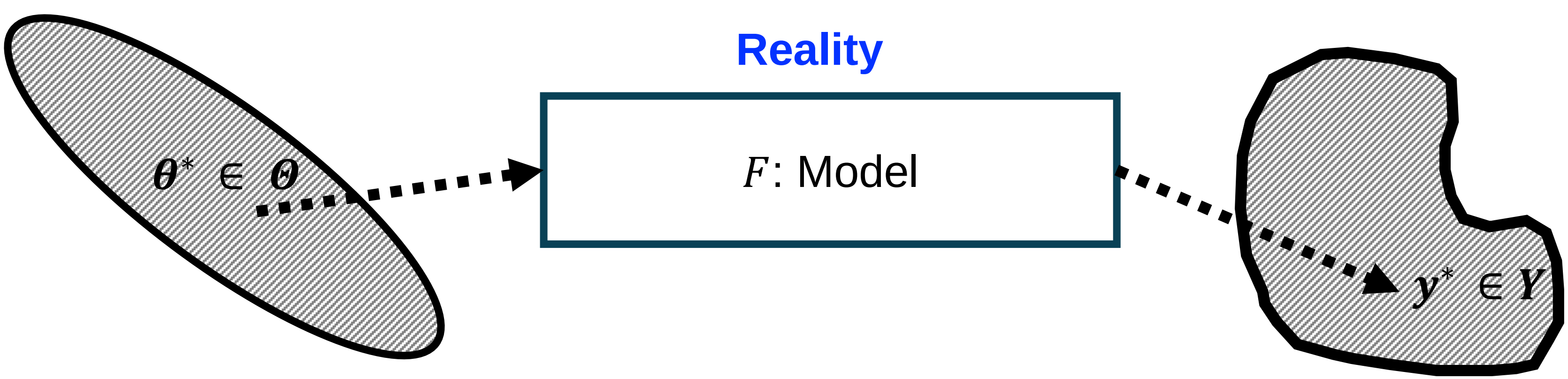
Computational science relies on the use of **models** to make predictions

We often work in a **deterministic** framework

Nominal Conditions

θ: inputs to model (parameters) → F: Model → y: quantities of interest

Reality



Why Uncertainty Quantification?

UQ is mostly just another term for **statistics**

Some communities have specific meanings, like "statistics of physical computer models"
(as opposed to linear regression)

Types of UQ

- Spatiotemporal statistics
- Sensitivity analysis
- Uncertainty Propagation
- Parameter estimation
- Data assimilation
- Other inverse problems

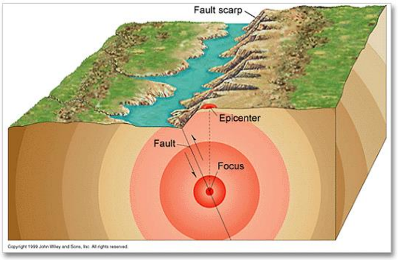
Applications of UQ

- Optimal Experimental Design
- Forecasting
- Robust Control
- Active Learning
- Image Processing
- Bioinformatics

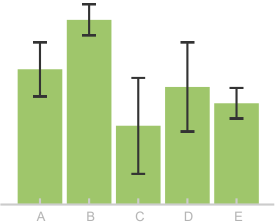
Brookhaven National Laboratory

Sources of Uncertainty

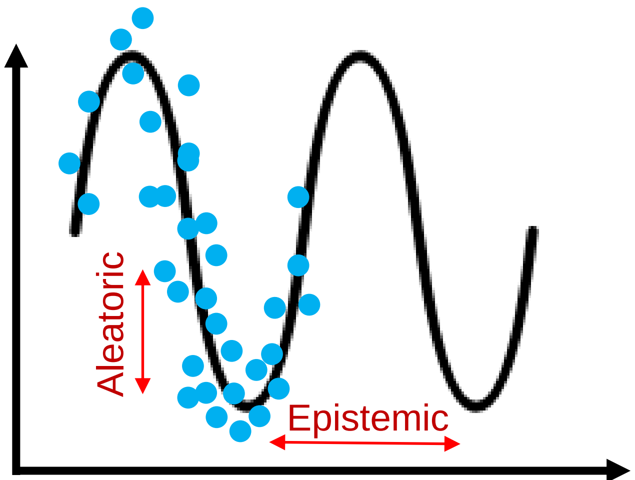
- **Aleatoric** (i.e. random)
  - Inherently stochastic aspects of a system



- **Epistemic** (i.e. non-random)
  - Incomplete or flawed information

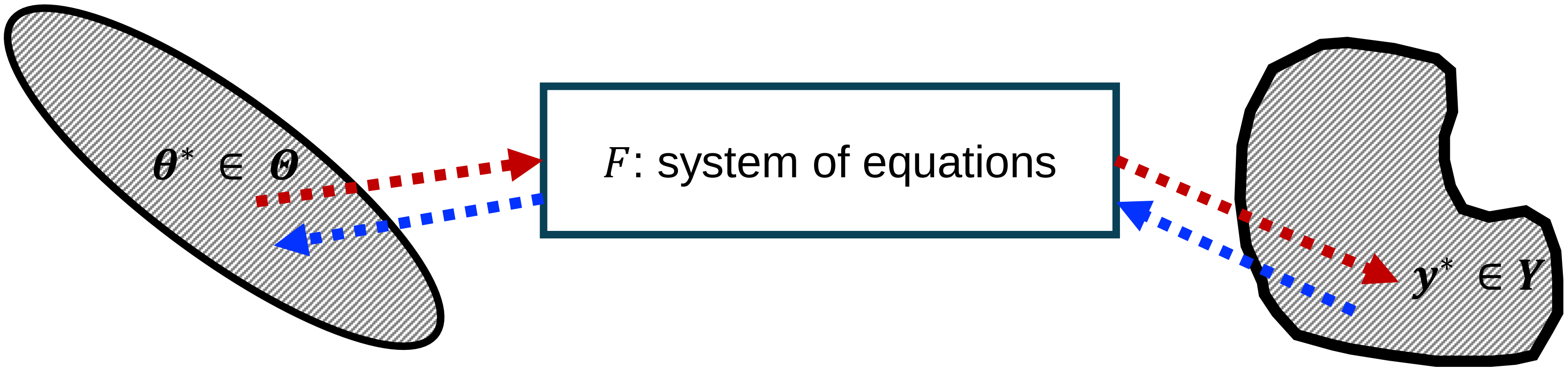


- Relevance to **Machine Learning**:
  - Epistemic: Quantity of training data
    - This can be reduced with more data, property of the model
  - Aleatoric: Noise in measured data
    - Irreducible property of the data, not the model



Abdar, Moloud, et al.

Bayesian UQ for Computer Models



- Forward UQ $\boldsymbol{F}(\boldsymbol{\Theta}) = \boldsymbol{Y}$
  - Propagate input uncertainty through model to quantify output uncertainty
- Inverse UQ $\boldsymbol{F}^{-1}(\boldsymbol{Y}) = \boldsymbol{\Theta}$
  - Use available output data to improve characterization of input uncertainties

Bayesian Uncertainty Propagation (Forward UQ)

UQ: **Identify**, **Characterize** and **Propagate** uncertainty through a computational model

- **Identify**
  - What are the sources of uncertainty in the inputs?
- **Characterize**
  - Identify probability distributions which describe the uncertainty in the inputs
- **Propagate**
  - Sample uncertain model inputs, solve model at those inputs, obtain distribution of model outputs

Bayesian Parameter Inference (Inverse UQ) workflow

(1) Preparation Work

Determine **data** constraints (what is measured) → y

Elicit **prior** parameter distributions → $p(\theta)$

(2) Model Specification

Note: likelihood is *data-generating process*
Samples should match data!

Formulate a **likelihood** function (i.e., the **distribution of the data** that would arise if we knew all the parameters)

Use **residual analysis** from a "best fit" to guide model building

How is **data-model misfit distributed**? (Marginal distribution? Bias? Correlation?)

For **regression**: data = model + error (e.g., measurement error, unmodeled variability, model bias or other "discrepancy")

↓ $p(y|\theta)$

(3) Prior Predictive Check

Sample from prior, then conditionally from likelihood (predictive distribution) compute $y^{sim} \sim p(y|\theta^{sim})$ →

Do samples resemble our expectations of data?

Yes → **You're ready to conduct Bayesian inference!**

No → **Revisit your model specification, may be an issue with the likelihood or priors!**

Bayesian Parameter Inference (Inverse UQ) workflow

(4) Inference

Usually **MCMC**

Look for good **effective sample size**

Use **convergence consistency diagnostics** (Rhat) over multiple chains

$p(\theta|y)$ →

(5) Posterior Checks

Sample from prior, then posterior…Do samples look like data?

Compute $p(y'|y)$ where y' is unseen data

$p(\theta|y)$ →

(6) Iterative Model Building

If your **predictions** (based on the posterior estimates) **do not match** the data, you may need to **revisit aspects of the statistical model** (likelihood or prior, or confirm MCMC convergence)

If this seems overwhelming, fear not! We will review (and you have at your disposal) a **Jupyter Notebook** that will walk through the **entirety of this workflow** on a problem from literature!

Got a complex computer model?

Q: MCMC and related methods require **A LOT of model evaluations** to converge…*that'll never work for my model!*

A: Have you considered building a **statistical emulator of your model**?

Statistical Emulators

*"A **statistical emulator** is a **fast** proxy for a **complex computer model** which **predicts** model **output** at **arbitrary parameter settings** from a **limited ensemble** of training data[1]"*

Model output is deterministic in the inputs, **but unknown** at any point which we do not evaluate it directly

Take the Bayesian approach, which is "put a prior on it!"

Represent your computer model as an **unknown (random) function** and represent its uncertainty with a **stochastic prior process** (e.g., a Gaussian Process prior)

$$\eta(.)|\, B, \Sigma, \phi \sim GP(m(.)., \Sigma c(.))$$

m: non-constant mean function with regression coefficients B
c: correlation function with correlation length hyperparameter ϕ

Training (updating) on data yields **GP posterior**

Kennedy, M.C. and O'Hagan, A., 2001. Bayesian calibration of computer models. Journal of the Royal Statistical Society: Series B (Statistical Methodology), 63(3), pp.425-464.

Urban, N. M., & Fricker, T. E. (2010). A comparison of Latin hypercube and grid ensemble designs for the multivariate emulation of an Earth system model. *Computers & Geosciences*, *36*(6), 746-755.

Emulation Workflow

(1) Preparation Work

Elicit **prior** parameter distributions

$p(\theta)$ →

Design an **ensemble** of training data

$(\theta, y)^n$ →

- Sample from priors
- Uniform space-filling (e.g., Sobol Quasi-random or Latin hypercube)
- Mixture (e.g. 50:50) of both

(2) Dimension Reduction

Do **dimension reduction on training inputs** if needed (sensitivity analysis, e.g., elementary effects method)

Do **dimension reduction** on training outputs if needed (e.g., **PCA**, **autoencoder**, …)

(3) Fit Emulator

Loss is often *k*-fold cross validation error, or can be negative log likelihood for maximum likelihood estimation if emulator provides likelihood (as in Gaussian process)

(4) Validate Emulator

Validate on held-out test set of ensemble data

(If available) Validate emulator prediction uncertainties (e.g., width of predictive intervals – does 95% interval contain 95% of hold-out data)

Tips to avoid perfunctory statistical modeling

- **Assumptions matter!** As we saw, a lot of UQ is based on modeling assumptions
- **Don’t use uniform priors** (they are not “non-informative”)
  - Ex: flat prior on x is not flat under re-parameterization $x \mapsto f(x)$
- **Informative priors**: use **domain knowledge**
  - Prior sensitivity analysis in case you’re not confident in your knowledge; change width of prior, etc.
  - This doesn’t give a unique inference, but this is fine. **Statistics isn’t magic**. Conclusions depend on assumptions.
  - “Weakly” informative priors: make them wider than you think they should be to hedge against misspecification
- **Put a lot of effort into the likelihood function**, which is the data-generating process that specifies your statistical model
  - Ex: in regression, observation = model + error, what is “error”?
  - “Error” could be non-Gaussian, heteroskedastic (varying in magnitude), correlated: **check for this**!
  - “Error” could include model bias (systematic error), which is another uncertainty
  - **Think of the likelihood as a forward model of your data**, including everything you have to do to your (computer model, regression function) in order to make it resemble what is measured
  - If measurements are partially observed, biased, missing-not-at-random, etc., you have to model those in your likelihood too
  - If samples from the likelihood don’t look like your data (conditional on samples from your prior), you have more work to do

Bayesian Deep Learning

Q: Neural network prediction problems?

- NNs are often **underspecified** (parameters >> data)
- MAP maximization training leads to a **point estimate** for parameters → degeneracy in optima!

- **Bayesian deep learning:** applying posterior inference to the parameters (weights, biases) of a neural network
- **How can it be done, practically (approximately)?**
  - Hamiltonian or random-walk Monte Carlo don't scale well with deep networks due to the large number of parameters to estimate
  - **Variational inference** → Normal approximations for weights that are learned via optimization
  - **Dropout** → A technique for regularizing neural network predictions by randomly turning off parts of the network and forcing its predictions to be robust to that degradation, can be shown to have a Bayesian UQ interpretation under specific assumptions
  - **Deep ensembles** → A popular non-Bayesian technique where you train ensembles of neural networks and let the multi-network variance be a proxy for uncertainty, and can be shown to have an approximate Bayesian interpretation
  - **Bayesian last layer methods** → Freeze most of the network weights after training, but perform fully Bayesian inference on the weights in the last layer(s), under the premise that the early layers are just doing feature extraction, and the actually regression/classification task occurs at the end

Jupyter Notebook

TuringLang/**Turing.jl**

Bayesian inference with probabilistic programming.

78 Contributors · 103 Issues · 11 Discussions · 2k Stars · 197 Forks

SciML/
DifferentialEquations.jl

Multi-language suite for high-performance solvers of differential equations and scientific machine learning (SciML) components

22 Contributors · 124 Issues · 2k Stars · 189 Forks

Further Reading

- Gelman, A., Carlin, J.B., Stern, H.S. and Rubin, D.B., 1995. ***Bayesian data analysis***. Chapman and Hall/CRC.

- Smith, R.C., 2013. ***Uncertainty quantification: theory, implementation, and applications*** (Vol. 12). Siam.

- Kennedy, M.C. and O'Hagan, A., 2001. ***Bayesian calibration of computer models***. Journal of the Royal Statistical Society: Series B (Statistical Methodology), 63(3), pp.425-464.

- Conti, Stefano, and Anthony O’Hagan. ***Bayesian emulation of complex multi-output and dynamic computer models***. Journal of statistical planning and inference 140.3 (2010): 640-651.

- Abdar, Moloud, et al. ***A review of uncertainty quantification in deep learning: Techniques, applications and challenges***. Information Fusion 76 (2021): 243-297.

- Wilson, A.G., 2020. ***The case for Bayesian deep learning***. arXiv preprint arXiv:2001.10995.

- Wilson, A.G. and Izmailov, P., 2020. ***Bayesian deep learning and a probabilistic perspective of generalization***. Advances in neural information processing systems, *33*, pp.4697-4708.