UQTk

A Flexible Python/C++ Toolkit for Uncertainty Quantification

Bert Debusschere, Khachik Sargsyan, Cosmin Safta, Prashant Rai, Kenny Chowdhary

bjdebus@sandia.gov
Sandia National Laboratories,
Livermore, CA, USA

Thursday April 19, 2018 – SIAM UQ 18

FASTMATH

Acknowledgements

Cosmin Safta Khachik Sargsyan Kenny Chowdhary Habib Najm	Sandia National Laboratories Livermore, CA, USA
Omar Knio	KAUST, Saudi Arabia
Roger Ghanem	University of Southern California Los Angeles, CA, USA
Olivier Le Maître	LIMSI-CNRS, Orsay, France
and many others ...	

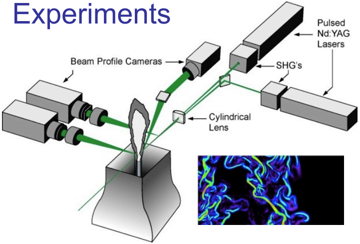
This material is based upon work supported by the U.S. Department of Energy, Office of Science, Office of Advanced Scientific Computing Research (ASCR), Scientific Discovery through Advanced Computing (SciDAC) and Applied Mathematics Research (AMR) programs.

Sandia National Laboratories is a multimission laboratory managed and operated by National Technology & Engineering Solutions of Sandia, LLC, a wholly owned subsidiary of Honeywell International Inc., for the U.S. Department of Energy's National Nuclear Security Administration under contract DE-NA0003525. The views expressed in this talk do not necessarily represent the views of the U.S. Department of Energy or the United States Government.

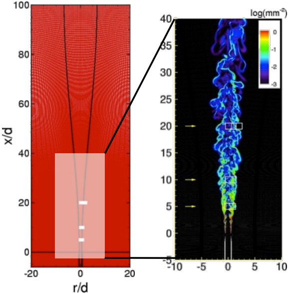
Outline

UQ is about enabling predictive simulations

Experiments



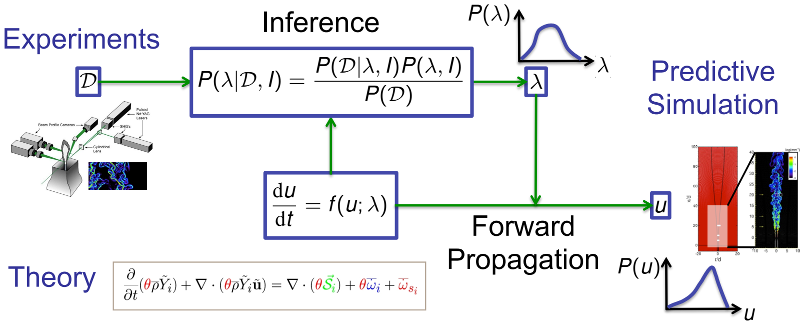
Predictive Simulation



$$\frac{\partial}{\partial t}(\theta\bar{\rho}\tilde{Y}_i) + \nabla\cdot(\theta\bar{\rho}\tilde{Y}_i\tilde{\mathbf{u}}) = \nabla\cdot(\theta\vec{S}_i) + \theta\bar{\dot{\omega}}_i + \bar{\dot{\omega}}_{s_i}$$

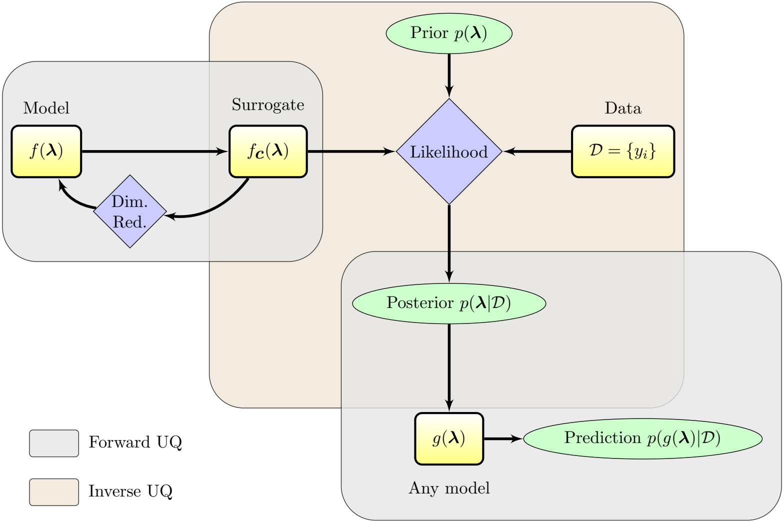
Theory

UQ methods extract information from all sources to enable predictive simulation



- UQ not just about propagating uncertainties
- The term UQ covers a wide range of methods

An example UQTk workflow



UQTk provides tools to build a general UQ workflow

- Tools for
  - Representation of random variables and stochastic processes
  - Forward uncertainty propagation
  - Inverse problems
  - Sensitivity analysis
  - Dimensionality reduction
  - Bayesian Compressive Sensing
  - Low Rank Tensors
  - Gaussian Processes
  - ...
- Tools can be used stand-alone or combined into a general workflow

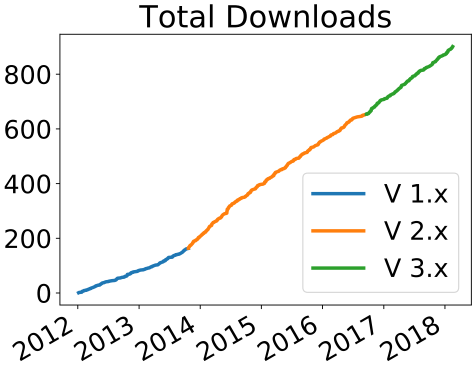
UQTk is meant to be straightforward to download, install and use

- Target usage:
  - Rapid prototyping of UQ workflows
  - Algorithmic research in UQ
  - Tutorials / educational
  - Expertise in UQ methods (or a desire to acquire it) helpful
- Released under the GNU Lesser General Public License
  - `http://www.sandia.gov/UQToolkit/`
  - Current version 3.0.4
  - Version 3.1.0 planned for later this year
- No massive third party libaries to download, install, and configure

UQTk is used in a variety of applications

- Direct collaborations
  - US DOE SciDAC FASTMath institute
    `https://fastmath-scidac.llnl.gov/`
  - Variety of US DOE SciDAC partnership projects
  - Part of US DOE BER E3SM climate model analysis tools
- Many other research groups at universities, National Labs, and industry
- Always welcome new applications / collaborations
- Mailing lists
  - `uqtk-announce@software.sandia.gov`
  - `uqtk-users@software.sandia.gov`
  - Join at `http://www.sandia.gov/UQToolkit`

UQTk Downloads



- Downloads from `http://www.sandia.gov/UQToolkit`
- $\approx$ 900 total downloads
- $\approx$ 200 downloads of version 3.x

We rely on Polynomial Chaos expansions (PCEs) to represent uncertainty

- Standard PC Basis types supported:
  - Gauss – Hermite
  - Uniform – Legendre
  - Gamma – Laguerre
  - Beta – Jacobi
- Also support for custom orthogonal polynomials
  - Defined by user-provided three-term recurrence formula
- Both intrusive and non-intrusive PC tools provided
  - Primarily Galerkin projection methods
  - Some regression approaches offered through Bayesian Compressed Sensing module
  - See also Debusschere, *et al.* 2004; Sargsyan, *et al.* 2014

UQTk uses a combination of C++ and Python

- Main libraries in C++
  - `PCBasis` and `PCSet` classes: PC tools (intrusive and non-intrusive)
  - `Quad` class: quadrature rules (full tensor and sparse tensor product rules)
  - `MCMC, Gproc,...`
- Functionality available via
  - Direct linking of C++ code
  - Standalone apps
  - Python interface based on Swig (UQTk version 3.0)
- Download as tar file and configure with `CMake`
- Examples of common workflows provided

Upcoming Features in UQTk 3.1.0

- New Functionality
  - Canonical Low Rank Tensor (LRT) Representations
  - Data Free Inference (DFI) Library
  - tempered MCMC (tMCMC)
- Enhanced functionality
  - Python scripts for model evidence computation
  - Python class for Bayesian Compressed sensing
  - Additional examples and tutorials
- Expected Fall 2018
  - Sign up for the announcement mailing list `uqtk-announce@software.sandia.gov` at `http://www.sandia.gov/UQToolkit`

Longer Term Plans

- Coupling with other libraries
  - Better support for user specified third-party libaries, e.g. random number generators, integrators, ...
  - Coupling with DAKOTA and MUQ for leveraging functionality
- Mixed PC basis types
- More general multi-index specification
- Data structures amenable to parallelization and GPU acceleration
- Other developments you would like to see?
  - Let us know at `uqtk-users@software.sandia.gov`

Outline

Bayesian Compressed Sensing

- N training data points $(\boldsymbol{x}_n, u_n)$ and K basis terms $\Psi_k(\cdot)$
- Projection matrix $\boldsymbol{P}^{N\times K}$ with $\boldsymbol{P}_{nk} = \Psi_k(\boldsymbol{x}_n)$
- Find regression weights $\boldsymbol{c} = (c_0, \ldots, c_{K-1})$ so that

$$\boldsymbol{u} \approx \boldsymbol{P}\boldsymbol{c} \quad \text{or} \quad u_n \approx \sum_k c_k \Psi_k(\boldsymbol{x}_n)$$

- The number of polynomial basis terms grows fast; a p-th order, d-dimensional basis has a total of $K = (p+d)!/(p!d!)$ terms.
- For limited data and large basis set ($N < K$) this is a sparse signal recovery problem $\Rightarrow$ need some regularization/constraints.

- Least-squares $argmin_{\boldsymbol{c}} \left\{ ||\boldsymbol{u} - \boldsymbol{P}\boldsymbol{c}||_2^2 \right\}$
- The 'sparsest' $argmin_{\boldsymbol{c}} \left\{ ||\boldsymbol{u} - \boldsymbol{P}\boldsymbol{c}||_2^2 + \alpha ||\boldsymbol{c}||_0 \right\}$
- Compressive sensing $argmin_{\boldsymbol{c}} \left\{ ||\boldsymbol{u} - \boldsymbol{P}\boldsymbol{c}||_2^2 + \alpha ||\boldsymbol{c}||_1 \right\}$

Bayesian Compressed Sensing

- N training data points $(\boldsymbol{x}_n, u_n)$ and K basis terms $\Psi_k(\cdot)$
- Projection matrix $\boldsymbol{P}^{N\times K}$ with $\boldsymbol{P}_{nk} = \Psi_k(\boldsymbol{x}_n)$
- Find regression weights $\boldsymbol{c} = (c_0, \ldots, c_{K-1})$ so that

$\boldsymbol{u} \approx \boldsymbol{P}\boldsymbol{c}$ or $u_n \approx \sum_k c_k \Psi_k(\boldsymbol{x}_n)$

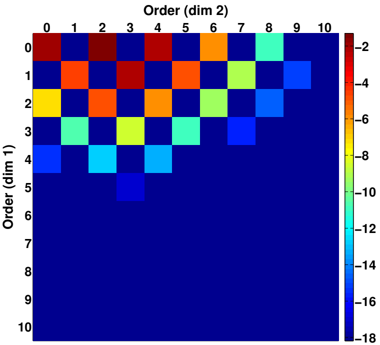
- The number of polynomial basis terms grows fast; a p-th order, d-dimensional basis has a total of $K = (p+d)!/(p!d!)$ terms.
- For limited data and large basis set ($N < K$) this is a sparse signal recovery problem $\Rightarrow$ need some regularization/constraints.

- Least-squares $argmin_{\boldsymbol{c}} \left\{ ||\boldsymbol{u} - \boldsymbol{P}\boldsymbol{c}||_2^2 \right\}$
- The 'sparsest' $argmin_{\boldsymbol{c}} \left\{ ||\boldsymbol{u} - \boldsymbol{P}\boldsymbol{c}||_2^2 + \alpha ||\boldsymbol{c}||_0 \right\}$
- Compressive sensing $argmin_{\boldsymbol{c}} \left\{ ||\boldsymbol{u} - \boldsymbol{P}\boldsymbol{c}||_2^2 + \alpha ||\boldsymbol{c}||_1 \right\}$

Bayesian Likelihood Prior

BCS removes unnecessary basis terms

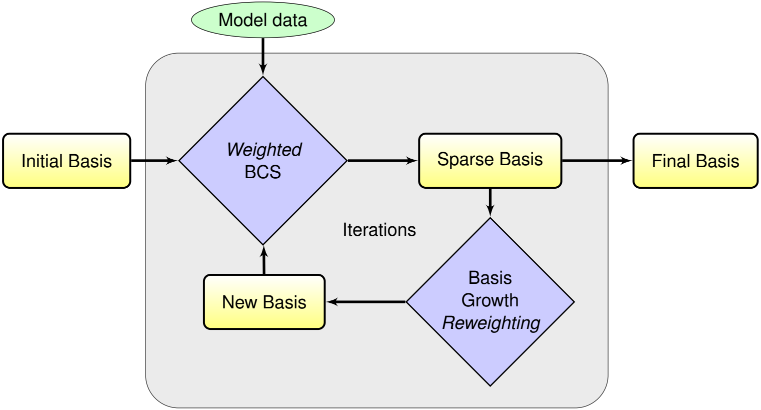
$f(x,y) = \cos(x+4y)$ $f(x,y) = \cos(x^2+4y)$



The square (i,j) represents the (log) spectral coefficient for the basis term $\psi_i(x)\psi_j(y)$.

Iterative Bayesian Compressive Sensing (iBCS)

- *Iterative BCS*: iteratively increase the order for the relevant basis terms while maintaining the dimensionality reduction [Sargsyan *et al.* 2014], [Jakeman *et al.* 2015].
- Combine basis growth and reweighting!



Low-Rank Approximations

- Univariate function representation: $p+1$ coefficients

$$u(\xi) = \sum_{k=0}^{P} u_k \psi_k(\xi)$$

- Multivariate function representation: $P+1 = \frac{(n+p)!}{n!p!}$

$$u(\xi_1, \ldots, \xi_n) = \sum_{k=0}^{P} u_k \prod_{i=1}^{n} \psi_{\alpha_i^k}(\xi_i)$$

- Low-rank approximation: $r * (p_1 + p_2 + \ldots + p_n)$

$$u(\xi_1, \ldots, \xi_n) = \sum_{k=1}^{r} w_k^{(1)}(\xi_1) \ldots w_k^{(n)}(\xi_n)$$

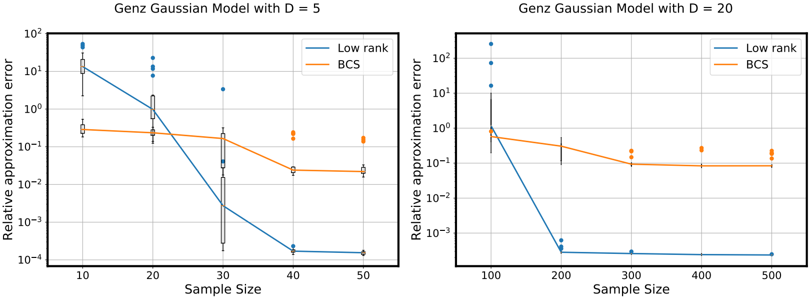
Minimization Problem for Coefficients

- Minimization problem:

$$\min_{\tilde{u}\in\mathcal{M}} \|u(\xi) - \tilde{u}(\xi)\|^2 + \lambda\mathcal{R}(\tilde{u}(\xi))$$

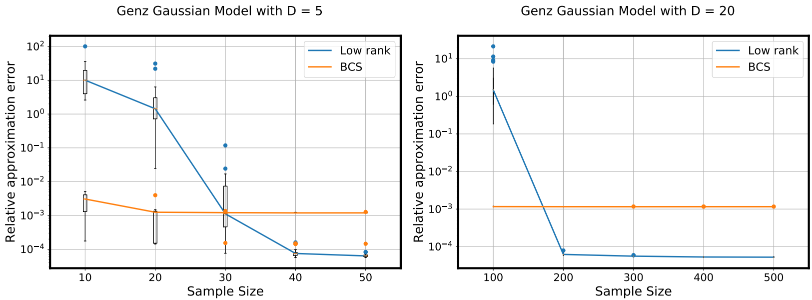
  where $\mathcal{M}$ is a suitable tensor subset (Canonical, Tensor Train) and $\mathcal{R}$ is a regularization function
- Selection of optimal rank r and regularization coefficient λ using cross validation
- Pros: Linear increase in parameters with dimension
- Cons: Non-linear optimization problem. Optimal approximation $\tilde{u}(\xi)$ is often not known

Low Rank Tensor and Bayesian Compressed Sensing Applied to Gauss Genz Function



- $u(x) = \exp\left(-\sum_{i=1}^{d} w_i^2 (x_i - b)^2\right), \;\; w_i = 1, b = 0.5$
- Box plot over 21 replica tests
- Low-rank approach performs well on function with inherent low-rank structure

Low Rank Tensor and Bayesian Compressed Sensing Applied to Gauss Genz Function



- $u(x) = \exp\left(-\sum_{i=1}^{d} w_i^2 (x_i - b)^2\right), \ w_i = \frac{1}{i^3}, b = 0.5$
- Box plot over 21 replica tests
- Bayesian compressive sensing does well on function with inherent sparsity and low number of samples

Outline

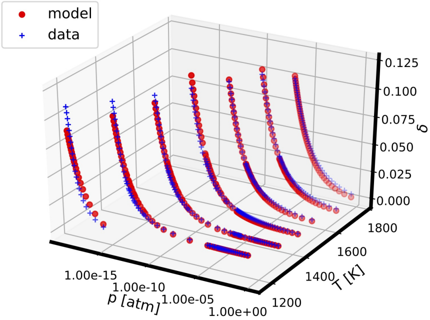
Bayesian Inference and Model Comparison

- Model for thermodynamic properties of RedOx active materials
- Used in design of materials for solar thermochemical Hydrogen production
- General model form $\delta = f(p_{O_2}, T)$
  - Model A: 4 parameters
  - Model B: 8 parameters
- Bayesian parameter inference and model comparison
- Joint work with Dr. Ellen Stechel at Arizona State University
- Funded by the DOE Office of Energy Efficiency and Renewable Energy (EERE)

Bayesian Inference and Model Comparison

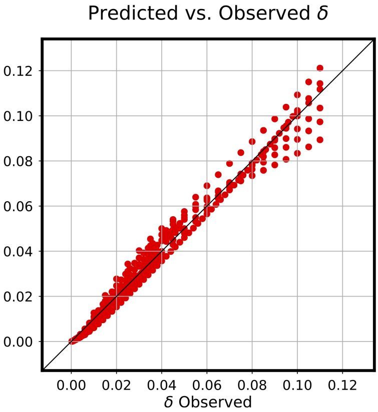
- Employed UQTk Python Bayesian Inference tools to infer parameters and compare the two models
- Model properties and numerical settings specified via flexible xml input file
- Python postprocessing and model evidence computation
- Workflow will be part of UQTk 3.1.0 release later this year

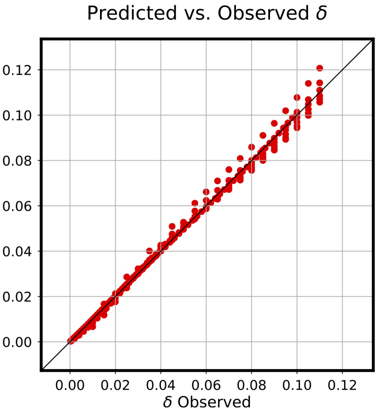
Both models agree well with data



- Model A (left) and Model B (right)

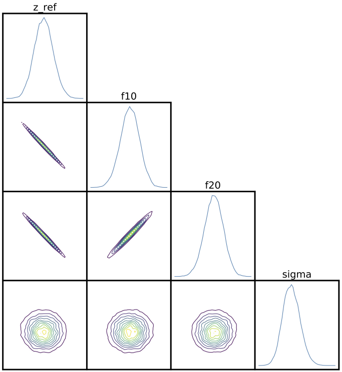
Both models agree well with data

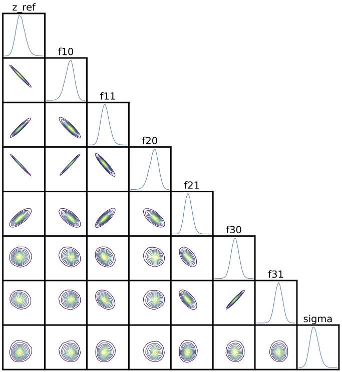




- Model B (right) has smaller residuals

Posterior distributions were sampled with adaptive MCMC





- Well-defined uni-modal distributions
- Model B has more dependencies between its parameters

Model evidence favors model B

- Model evidence computed from posterior samples, using a Gaussian approximation
  - Model A: Ln(evidence) = 1580
  - Model B: Ln(evidence) = 1939
- Despite its higher complexity, model B is clearly favored.
- For situations with more measurement noise, or fewer data points, a simpler model maybe preferred

Summary

- UQTk provides a powerful set of tools for building general UQ workflows
- Multiple ways to access functionality
  - Direct linking of C++ code
  - Standalone apps
  - Python interface based on Swig
- Available at
  `http://www.sandia.gov/UQToolkit`
- Suggestions for improvements welcome!
- Do not hesitate to contact us
  `uqtk-users@software.sandia.gov`

References

- B. Debusschere, H. Najm, P. Pébay, O. Knio, R. Ghanem and O. Le Maître, "Numerical Challenges in the Use of Polynomial Chaos Representations for Stochastic Processes", *SIAM J. Sci. Comp.*, 26:2, 2004.
- K. Sargsyan, *et al.*, "Dimensionality reduction for complex models via bayesian compressive sensing", *Int. J. of Uncertainty Quantification*, 4, 1:63–93, 2014.
- "Handbook of Uncertainty Quantification", R. Ghanem, D. Higdon, H. Owhadi (Eds.), Springer, 2016, `http://www.springer.com/us/book/9783319123844`
- Jakeman, J. D., Eldred, M. S., and Sargsyan, K., "Enhancing ℓ_1-minimization estimates of polynomial chaos expansions using basis selection," *J. of Computational Physics*, 289, 18–34, 2015.
- UQ Tutorials: `http://www.quest-scidac.org/outreach/tutorials/`