SAND2024-01829C

UQTk

The UQTk C++/Python Toolkit for Uncertainty Quantification: Overview and Applications

Caitlin Curry, Khachik Sargsyan, Cosmin Safta, and *Bert Debusschere*

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

Thursday Feb 29, 2024 – SIAM UQ 24
SAND2024-NNNN C

Acknowledgements

Funding:

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), Applied Mathematics Research (AMR), and Workforce Development for Teachers and Scientists (WDTS) programs.

Contract:

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.

Authors and Contributors

UQTk Authors (alphabetical by first name):

Bert Debusschere (SNL), Caitlin Curry (SNL), Cosmin Safta (SNL), Katherine Johnston, Kenny Chowdhary, Khachik Sargsyan (SNL), Luke Boll, Mohammad Khalil (SNL), Pieterjan Robbe (SNL), Prashant Rai, Tiernan Casey (SNL), Xiaoshu Zeng (USC)

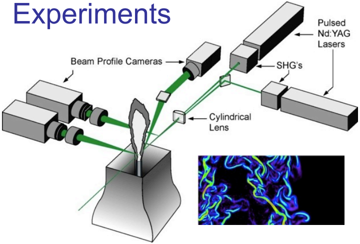
UQTk Contributors (alphabetical by first name):

Habib Najm (SNL), Helgi Adalsteinsson, Linus Seelinger (KIT), Majid Latif, Olivier Le Maître (LIMSI-CNRS), Omar Knio (KAUST), Roger Ghanem (USC), Sarah Castorena, Sarah de Bord, Xun Huan, and many others . . .

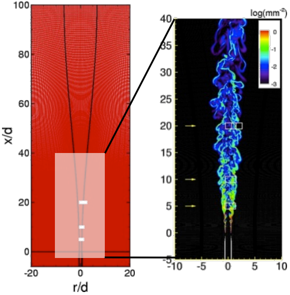
Outline

UQ is about enabling predictive simulations

Experiments



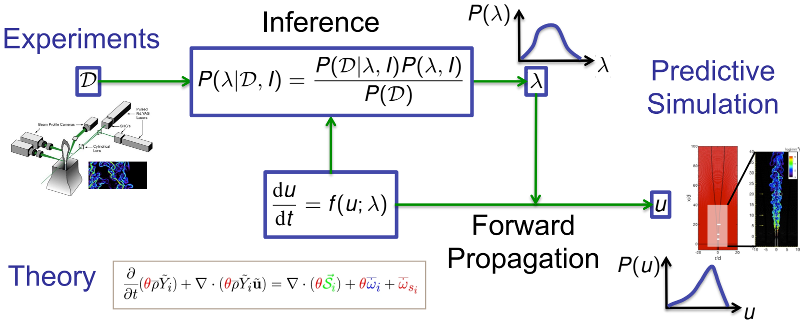
Predictive Simulation



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

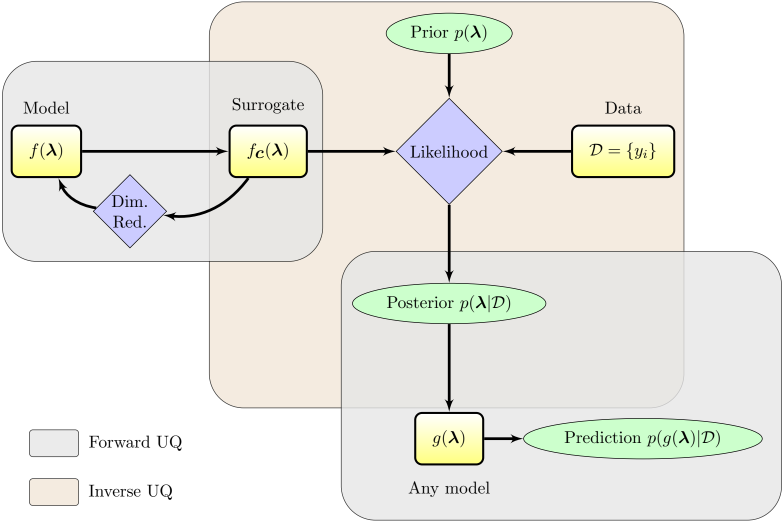
Theory

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 / Model calibration
  - Embedded model error
  - Global 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 geared towards research, education, prototyping

- 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 BSD 3-Clause License
  - `https://github.com/sandialabs/UQTk`
  - Current version 3.1.4
- Some dependencies included with UQTk. Others (Sundials) downloaded by CMake build system as needed

UQTk supports a wide variety of Polynomial Chaos Expansions (PCEs) operations

- 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

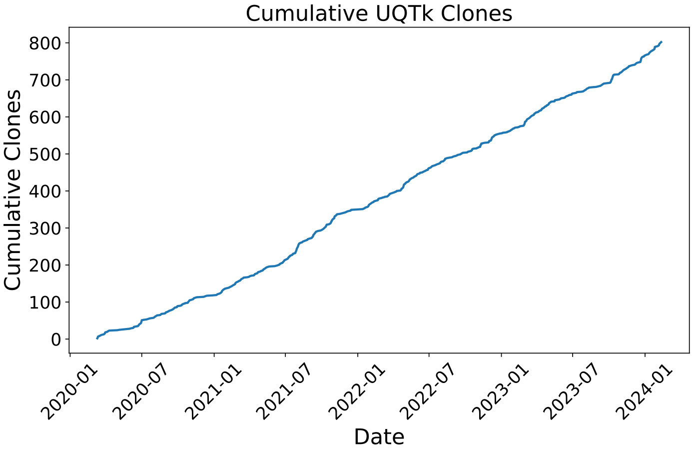
- Main libraries in C++
  - `PCBasis` and `PCSet` classes: PC tools (intrusive and non-intrusive)
  - `Quad` class: quadrature rules (full tensor and sparse tensor product rules)
  - `MCMC, Gproc,...`
- Functionality available via
  - Direct linking of C++ code
  - Standalone apps
  - Python interface using pybind11
- Examples of common workflows provided

Recent Features in UQTk 3.1.x

- New functionality (UQTk 3.1.0)
  - Canonical Low Rank Tensor (LRT) Representations
  - Data Free Inference (DFI) Library
  - tempered MCMC (tMCMC)
  - Basis adaptation
- Improved software engineering
  - Refactored MCMC Class (UQTk 3.1.1)
  - Interface to Python moved from swig to pybind11 (UQTk 3.1.2)
- Miscellaneous improvements
  - New DFI app, compatibility with Sundials 6.x, expanded Python pce_tools (UQTk 3.1.3)
  - UM-Bridge example, upgraded BCS interface (UQTk 3.1.4)
- Watch the UQTk repo and discussion at

`https://github.com/sandialabs/UQTk`

UQTk Github Repo Clones



- Clones from `https://github.com/sandialabs/UQTk/`
- $\approx$ 200 clones per year

UQTk is used in a variety of applications

- Direct collaborations
  - US DOE SciDAC FASTMath Inst. `https://scidac5-fastmath.lbl.gov/`
  - Variety of US DOE SciDAC partnership projects
  - DOE BER E3SM climate model analysis
- Many other groups at universities, National Labs, and industry
- Common uses: Surrogate Construction, Global Sensitivity Analysis, Bayesian Inference, Forward UQ
- Always welcome new applications / collaborations

Outline

1. UQTk Overview
2. Bayesian Compressed Sensing Illustration
3. Future Developments
4. Summary
5. References
6. Extra Material
7. Bayesian Model Inference and Comparison
8. SciDAC: Application in Climate Modeling

Compressed Sensing addresses sparsity in samples

Sparse regression (Compressive Sensing):

$$\begin{aligned} \boldsymbol{c}^{CS} &= \arg\min_{\boldsymbol{c}} \sum_{i=1}^{m} \left(f(\boldsymbol{x}(\boldsymbol{\xi}^{(i)})) - \sum_{\boldsymbol{\alpha}\in\mathcal{I}} c_{\boldsymbol{\alpha}} \Psi_{\boldsymbol{\alpha}}(\boldsymbol{\xi}^{(i)}) \right)^2 + \lambda \sum_{\boldsymbol{\alpha}\in\mathcal{I}} |\boldsymbol{c}_{\boldsymbol{\alpha}}| \\ &= \arg\min_{\boldsymbol{c}} \left[||\boldsymbol{f} - \boldsymbol{\Psi}\boldsymbol{c}||_2^2 + \lambda ||\boldsymbol{c}||_1 \right] \end{aligned}$$

. . . and in a Bayesian framework → Bayesian Compressive Sensing [Sargsyan, et al., 2014]:

$$\overbrace{p(\boldsymbol{c}|\mathcal{D})}^{\text{Posterior}} \propto \overbrace{p(\mathcal{D}|\boldsymbol{c})}^{\text{Likelihood}} \overbrace{p(\boldsymbol{c})}^{\text{Prior}} \longrightarrow \boldsymbol{c}^{MAP} = \arg\max_{\boldsymbol{c}} \log p(\boldsymbol{c}|\mathcal{D}) = \arg\max_{\boldsymbol{c}} \left[\log L_{\mathcal{D}}(\boldsymbol{c}) + \log p(\boldsymbol{c})\right]$$

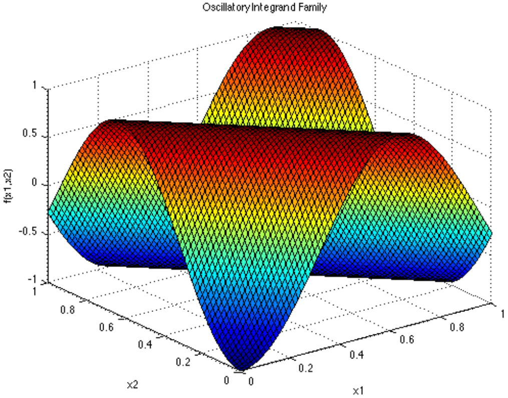
with Laplace sparsifying prior

$$p(\boldsymbol{c}) = \left(\frac{\lambda}{2}\right)^K \exp\left(-\lambda \sum_{\boldsymbol{\alpha}\in\mathcal{I}} |\boldsymbol{c}_{\boldsymbol{\alpha}}|\right)$$

UQTk BCS Function Call

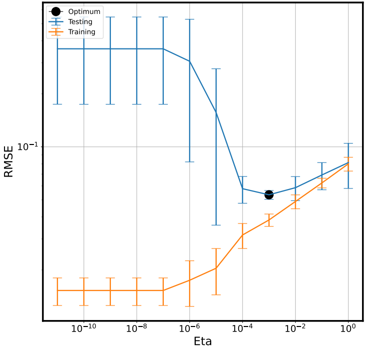
```
def UQTkBCS(pc_begin, xdata, ydata, eta=1.e-3, niter=1,\
            mindex_growth=None, ntry=1, eta_folds=5,\
            eta_growth=False, eta_plot = False,\
            regparams=None, sigma2=1e-8, npccut=None,\
            pcf_thr=None, verbose=0, return_sigma2=False):

Inputs:
pc_begin:   PC object with information about the starting basis
xdata:      Sampled input values [#samples, #dimensions]
ydata:      Function evaluations (QoIs)
eta:        NumPy array, list, or float with the threshold for
            stopping the evidence maximization algorithm.
niter:      Number of iterations for order growth
ntry:       Number of folds cross-validation of the retained basis

Outputs:
pc_model_final: PC object with retained basis terms
cfs_final:      Corresponding PC coefficients
```

Genz Oscillatory Function Test Case



- $f(x) = cos\left(\sum_{i=1}^{d} a_i x_i\right)$
- $d = 4, a_i = 2.0/i^2, \sigma_{data} = 0.05, nTrain = 100$
- `https://www.sfu.ca/~ssurjano/oscil.html`

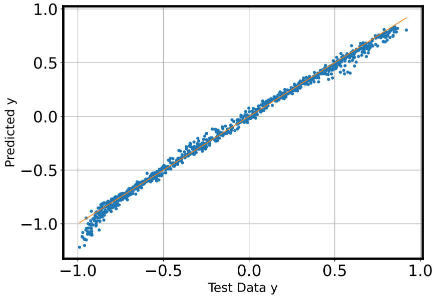
Cross-validation identifies the optimal stop criterion




```
import PyUQTk.pce as uqtkpce
import PyUQTk.PyPCE.pce_tools as pce_tools
nord = 4; ndim = 4; type = "LU"; alpha = 0.0; beta = 1.0
pc_begin = uqtkpce.PCSet("NISPnoq", nord, ndim, type, alpha, beta)
eta_range = 1/np.power(10,[i for i in range(0,12)])
pc_final, c_k = pcetools.UQTkBCS(pc_begin,xdata,ydata,eta=eta_range)
```

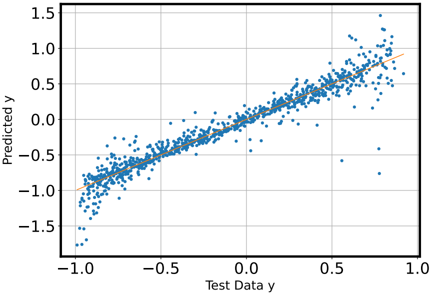
BCS approximates the Genz function quite well using only 10 basis terms

```
=====================================================
multi-indices(i,j), with i = 0 ... P, j = 1 ... nDim
with P = 9, and nDim = 4
-----------------------------------------------------
i\j | 1  2  3  4
------------------
0   | 1  0  0  0
1   | 0  1  0  0
2   | 0  0  0  0
3   | 2  2  0  0
4   | 0  0  1  0
5   | 2  0  0  1
6   | 2  0  1  0
7   | 3  0  0  0
8   | 0  0  0  1
9   | 0  0  0  2
```



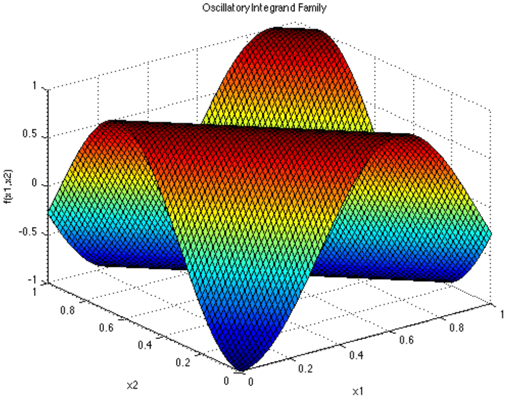
- Original full PCE has 70 terms
- Lower dimensions show up with highest order

Regression with full basis set has larger testing error



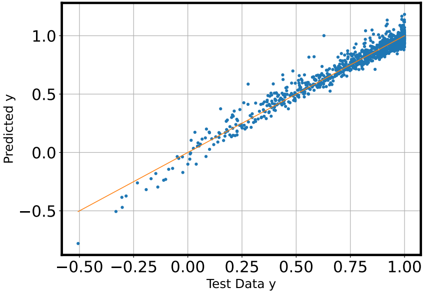
- Full PCE basis with 70 terms leads to overfitting

Higher-Dimensional Genz Oscillatory Function



- $f(x) = cos\left(\sum_{i=1}^{d} a_i x_i\right)$
- $d = 10, a_i = 2.0/i, \sigma_{data} = 0.05, nTrain = 400$
- `https://www.sfu.ca/~ssurjano/oscil.html`

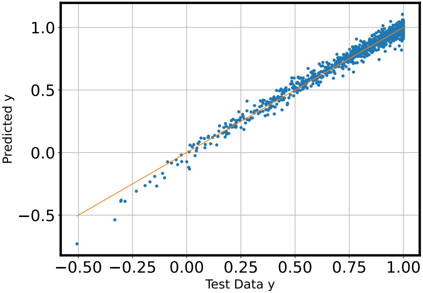
Compressed sensing on full basis is challenging



- Selects 31 basis terms out of a total of 1001
- Agreement is OK, but not stellar

Iterative basis growth better captures important terms

```
==== BCS with multiindex of size 66 ====
BCS has selected 53 basis terms out of 66
==== BCS with multiindex of size 267 ====
BCS has selected 37 basis terms out of 267
==== BCS with multiindex of size 222 ====
BCS has selected 65 basis terms out of 222
```




```
import PyUQTk.pce as uqtkpce
import PyUQTk.PyPCE.pce_tools as pce_tools
nord = 4; ndim = 4; type = "LU"; alpha = 0.0; beta = 1.0; n_it = 3
pc_begin = uqtkpce.PCSet("NISPnoq", nord, ndim, type, alpha, beta)
eta_r = 1/np.power(10,[i for i in range(0,12)])
pc_final,c_k = pcetools.UQTkBCS(pc_begin,xdata,ydata,
               mindex_growth='nonconservative',eta=eta_r,niter=n_it)
```

Outline

A Python-only implementation will provide more flexibility

- Current mix of C++ and Python is main source of install problems
- Original C++ data structures limit new developments (mixed PC bases) and make coupling to other packages challenging
- Original implementation pre-dates github development tools
- Some (large) Third Party Libraries (TPLs) support only legacy functionality (intrusive UQ)
- Plan to take main Python functionality into standalone Python Toolkit for UQ
    - Installation via pip install
    - Documentation directly into github pages

General Python Surrogate Modeling Interface

- There are many Python UQ tools and libraries in the community
- Most of them have different implementations of commonly used UQ operations
  - Surrogate models
  - Global Sensivity Analysis
  - Bayesian Inference
- Can we develop general interfaces to some of these operations?
  - Start with surrogate modeling interface
  - Avoid duplication and allow codes like DAKOTA to call surrogate models implemented by multitude of Python UQ toolboxes

Surrogate Modeling Interface Specifications

- Python class
  - Allow scalar and multivariate Quantities of Interest (QoIs)
  - Provide option to return derivatives?
  - Provide option to return Sobol' indices?
  - Provide option for adaptive refinement?
- Illustrative examples:
  - SMT: Surrogate Modeling Toolbox (`https://smt.readthedocs.io/`)
  - Others?
- Other suggestions?
- Input welcome at `bjdebus@sandia.gov`

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 through pybind11
- Available at `https://github.com/sandialabs/UQTk`
- Suggestions and questions welcome at `https://github.com/sandialabs/UQTk/discussions`
- Python-only version with general surrogate modeling interface in the planning stages

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.
- K. Sargsyan, H. Najm, and R. Ghanem. "On the Statistical Calibration of Physical Models". *International Journal for Chemical Kinetics*, 47(4): 246-276, 2015.
- K. Sargsyan, X. Huan, and H. Najm. "Embedded Model Error Representation for Model Calibration", in press, *International Journal for Uncertainty Quantification*, 2019.

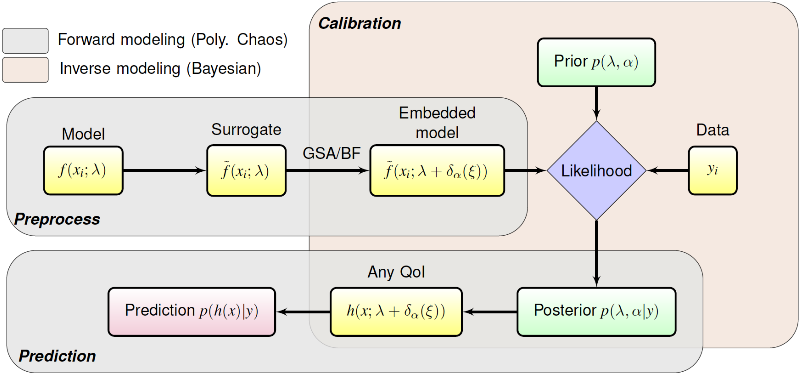
Outline

DAKOTA and UQTk implement similar methods but are geared towards different user groups.

- DAKOTA:
  - Geared towards end-user, analyst
  - Fully packaged, parallel workflow
- UQTk:
  - Geared towards developers, students, researchers
  - Components to build a workflow with
  - More lightweight and easier to get “under the hood”

There are plans to couple DAKOTA and UQTk through sharing libraries

General Uncertainty Quantification Workflow



- Predictive uncertainty decomposition: Total Variance =

Parametric uncertainty + Data noise + Model error + Surrogate error

Many flavors of MCMC are available

- Single Site MCMC (ssMCMC)
- Adaptive MCMC (aMCMC)
- Metropolis-adjusted Langevin algorithm (MALA) or Langevin sampling
- Tempered MCMC (tMCMC)

Surrogate models reduce the cost of computing Sobol' indices.

Variance-based decomposition:

$$f(x_1, x_2, \ldots, x_d) = f_0 + \sum_{1\leq i\leq d} f_i(x_i) + \sum_{1\leq i<j\leq d} f_{i,j}(x_i, x_j) + \sum_{1\leq i<j<k\leq d} f_{i,j,k}(x_i, x_j, x_k) + \ldots$$

- f_i, $f_{i,j}$, $f_{i,j,k}$, $\ldots$ are mutually orthogonal

Sobol' sensitivity indices measure fractional contributions of each parameter or group of parameters towards the total variance of selected QoIs

$$S_i = \frac{\mathrm{V}_{\boldsymbol{x}_i}\left[\mathrm{E}_{\boldsymbol{x}_{-i}}[f(\boldsymbol{x})]\,|\boldsymbol{x}_i]\right.}{\mathrm{V}\,[f(\boldsymbol{x})]}\ \text{(main)}, \quad S_i^T = \frac{\mathrm{E}_{\boldsymbol{x}_{-i}}\left[\mathrm{V}_{\boldsymbol{x}_i}[f(\boldsymbol{x})|\boldsymbol{x}_{-i}]\right]}{\mathrm{V}\,[f(\boldsymbol{x})]}\ \text{(total)}$$

- joint (most of the time between two variables) can also be informative

Sobol' indices estimates:

- Random Sampling → need computationally cheap (surrogate) models & slow to converge
- Polynomial Chaos Expansions → exploit orthogonality of basis terms

Compressed Sensing addresses sparsity in samples.

f (ELM-LF) is high-dimensional (47 input parameters)

- standard regression approaches are underdetermined $\rightarrow$

Sparse regression (Compressive Sensing):

$$\begin{aligned} \boldsymbol{c}^{CS} &= \arg\min_{\boldsymbol{c}} \sum_{i=1}^{m} \left(f(\boldsymbol{x}(\boldsymbol{\xi}^{(i)})) - \sum_{\boldsymbol{\alpha}\in\mathcal{I}} c_{\boldsymbol{\alpha}} \Psi_{\boldsymbol{\alpha}}(\boldsymbol{\xi}^{(i)}) \right)^2 + \lambda \sum_{\boldsymbol{\alpha}\in\mathcal{I}} |\boldsymbol{c}_{\boldsymbol{\alpha}}| \\ &= \arg\min_{\boldsymbol{c}} \left[||\boldsymbol{f} - \boldsymbol{\Psi}\boldsymbol{c}||_2^2 + \lambda ||\boldsymbol{c}||_1 \right] \end{aligned}$$

. . . and in a Bayesian framework $\rightarrow$ Bayesian Compressive Sensing [Sargsyan, et al., 2014]:

$$\overbrace{p(\boldsymbol{c}|\mathcal{D})}^{\text{Posterior}} \propto \overbrace{p(\mathcal{D}|\boldsymbol{c})}^{\text{Likelihood}} \overbrace{p(\boldsymbol{c})}^{\text{Prior}} \longrightarrow \boldsymbol{c}^{MAP} = \arg\max_{\boldsymbol{c}} \log p(\boldsymbol{c}|\mathcal{D}) = \arg\max_{\boldsymbol{c}} \left[\log L_{\mathcal{D}}(\boldsymbol{c}) + \log p(\boldsymbol{c})\right]$$

with Laplace sparsifying prior

$$p(\boldsymbol{c}) = \left(\frac{\lambda}{2}\right)^K \exp\left(-\lambda \sum_{\boldsymbol{\alpha}\in\mathcal{I}} |\boldsymbol{c}_{\boldsymbol{\alpha}}|\right)$$

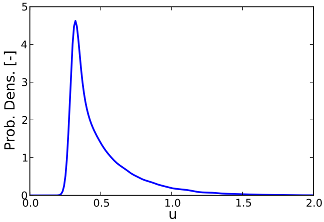
Longer Term Plans

- Coupling with other libraries
  - Better support for user specified third-party libraries, e.g. random number generators, integrators, ...
  - Coupling with DAKOTA (SNL) and MUQ (MIT) 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 `https://github.com/sandialabs/UQTk/discussions`

Polynomial Chaos Expansions represent random variables

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

- u: Random Variable (RV) represented with 1D PCE
- u_k: PC coefficients (deterministic)
- ψ_k: 1D Hermite polynomial of order k
- ξ: Gaussian RV

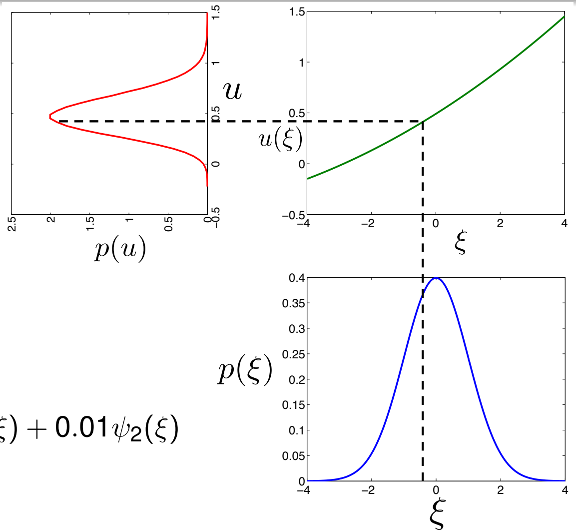


$$u = 0.5 + 0.2\psi_1(\xi) + 0.1\psi_2(\xi)$$

Expansion in terms of functions of random variables multiplied with deterministic coefficients

- Set of deterministic PC coefficients fully describes RV
- Separates randomness from deterministic dimensions

PCEs can be seen as a functional map from standard RVs to the represented RV



$$u = 0.5 + 0.2\psi_1(\xi) + 0.01\psi_2(\xi)$$

One-Dimensional Hermite Polynomials

$$\begin{aligned}\psi_0(\xi) &= 1 \\ \psi_k(\xi) &= (-1)^k e^{\xi^2/2} \frac{\mathrm{d}^k}{\mathrm{d}\xi^k} e^{-\xi^2/2}, \qquad k = 1, 2, \ldots\end{aligned}$$

$$\psi_1(\xi) = \xi, \quad \psi_2(\xi) = \xi^2 - 1, \quad \psi_3(\xi) = \xi^3 - 3\xi, \ \ldots$$

The Hermite polynomials form an orthogonal basis over $[-\infty, \infty]$ with respect to the inner product

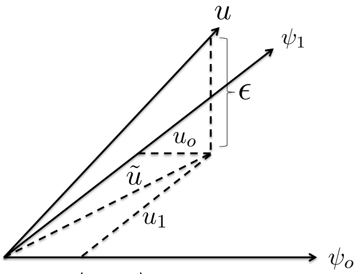
$$\langle \psi_i \psi_j \rangle \equiv \frac{1}{\sqrt{2\pi}} \int_{-\infty}^{\infty} \psi_i(\xi)\psi_j(\xi) w(\xi) d\xi = \delta_{ij} \left\langle \psi_i^2 \right\rangle$$

where $w(\xi) = e^{-\xi^2/2}$ is the weight function.

Note that $\frac{e^{-\xi^2/2}}{\sqrt{2\pi}}$ is the density of a standard normal random variable

Propagation of Uncertain Inputs Represented with PCEs

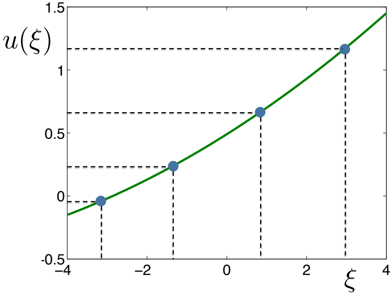
Galerkin Projection



$$u_k = \frac{\langle u\Psi_k\rangle}{\langle \Psi_k^2\rangle}, \quad k = 0, \ldots, P$$

Residual orthogonal to space covered by basis functions

Collocation



Match PCE to random variable at chosen sample points: interpolation or regression

Galerkin projection methods are either intrusive or non-intrusive

- Use same projection but in different ways

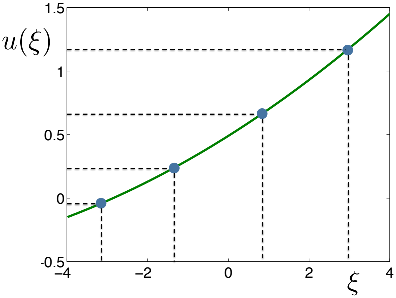
$$u_k = \frac{\langle u\Psi_k \rangle}{\langle \Psi_k^2 \rangle}, \quad k = 0, \dots, P$$

- Intrusive methods apply Galerkin projection to governing equations
  - Results in set of equations for the PC coefficients
  - Requires redesign of computer code
  - PCEs for all uncertain variables in system
- Non-intrusive approaches apply Galerkin projection to outputs of interest
  - Sampling to evaluate projection operator
  - Can use existing code as black box
  - Only computes PCEs for quantities of interest

Collocation approaches are non-intrusive and minimize errors at sample points

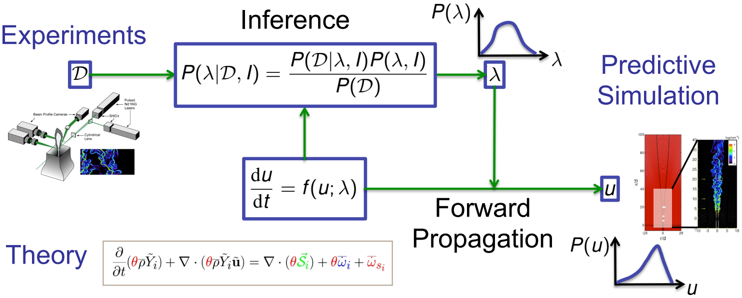
$$\sum_{k=0}^{P} u_k \Psi_k(\xi_i) = u(\xi_i)$$

$$i = 1, \ldots, N_c$$



- Use functional representation point of view
- Can use interpolation, e.g. Lagrange interpolants
- Or use regression approaches: $P + 1$ degrees of freedom to fit N_c points
- Can position points where most accuracy desired

Bayesian inference offers a probabilistic approach for inverse problems



- Bayesian inference can handle various sources of data
- Probabilistic formulation readily accommodates various sources of uncertainty

Bayes' rule updates prior belief with information extracted from data

- Bayes' rule

$$\overbrace{P(\boldsymbol{\lambda}|\mathcal{D})}^{\text{Posterior}} = \frac{\overbrace{P(\mathcal{D}|\boldsymbol{\lambda})}^{\text{Likelihood}}\,\overbrace{P(\boldsymbol{\lambda})}^{\text{Prior}}}{\underbrace{P(\mathcal{D})}_{\text{Evidence}}} \propto P(\mathcal{D}|\boldsymbol{\lambda})P(\boldsymbol{\lambda})$$

- Update prior distribution/knowledge about parameter $\boldsymbol{\lambda}$ to posterior distribution given data $\mathcal{D}$, using likelihood function $\mathcal{L}(\boldsymbol{\lambda}) \equiv P(\mathcal{D}|\boldsymbol{\lambda})$
- Data $\mathcal{D} = \{d_i\}_{i=1}^N$ - measurements of *some* quantities of interest (QoIs)
- Evidence $P(\mathcal{D})$ can be seen as a normalizing term

The prior distribution represents prior information about the inferred quantities

- Based on prior data, literature, or expert opinion
- Prior distribution helps to keep inference well defined, *e.g.* if quantity needs to remain positive
- If not much data available, posterior will be strongly influenced by the prior
- When a lot of data available, data will have predominant influence on posterior
- Prior is both powerful and dangerous
- If no prior information is available, non-informative priors can be used
  - *E.g.* uniform from $-\infty$ to $+\infty$

The likelihood function measures goodness-of-fit

- The key component that connects the model inputs to measured QoIs
- The *noise model* accounts for disagreement between model and data
  - Common case is i.i.d. Gaussian measurement noise in each data point

$$\mathcal{L}(\boldsymbol{\lambda}) = P(\mathcal{D}|\boldsymbol{\lambda}) = \frac{1}{(2\pi)^{N/2}\sigma^N} \exp\left(-\sum_{i=1}^{N} \frac{(d_i - f_i(\boldsymbol{\lambda}))^2}{2\sigma^2} \right)$$

  - If the model itself is uncertain, then the noise model needs to reflect that
- Generally the log-likelihood is used to avoid underflow

$$\ln \mathcal{L}(\boldsymbol{\lambda}) = \ln P(\mathcal{D}|\boldsymbol{\lambda}) = -\frac{N}{2}\ln(2\pi) - N\ln(\sigma) - \sum_{i=1}^{N} \frac{(d_i - f_i(\boldsymbol{\lambda}))^2}{2\sigma^2}$$

The posterior contains updated knowledge about inferred parameters

- Gives the inferred values of the parameters as well as their uncertainty based on all sources of uncertainty
- The maximum value is referred to as the *Maximum A Posteriori* (MAP) value
- Posterior distribution generally not analytically tractable
- Commonly people resort to MCMC sampling approaches to draw samples from this distribution
  - Samples can then be used to construct a PCE expansion for the inferred parameters
  - Can be fed into other models for forward propagation

Bayes' rule derives from elementary probability theory

Conditional probability:

$$P(A, B) = P(A|B)P(B) = P(B|A)P(A)$$

$$P(A|B) = \frac{P(B|A)P(A)}{P(B)}$$

Outline

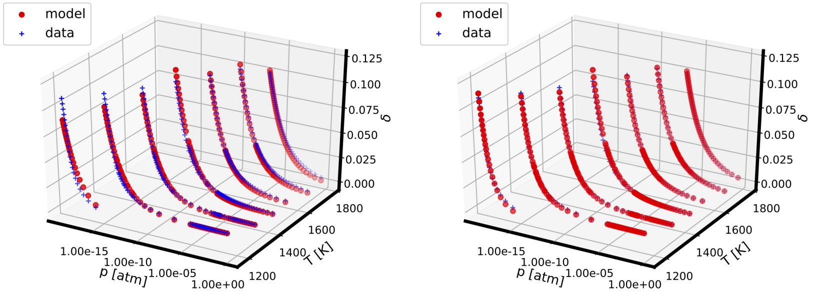
Bayesian Inference and Model Comparison

- 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 and Tony McDaniel at Sandia
- Funded by the DOE Office of Energy Efficiency and Renewable Energy (EERE)

Bayesian Inference and Model Comparison

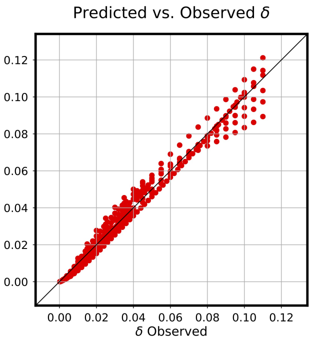
- Employed UQTk Python Bayesian Inference tools to infer parameters and compare the two models
- Model properties and numerical settings specified via flexible xml input file
- Python postprocessing and model evidence computation
- Workflow is an example included in the UQTk release

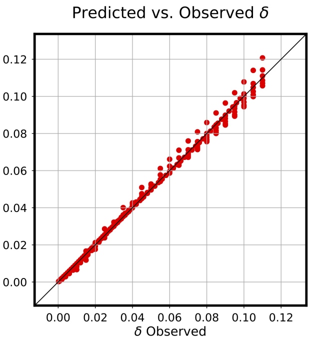
Both models agree well with data



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

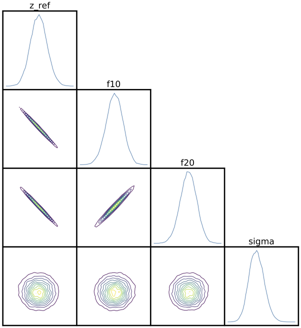
Both models agree well with data

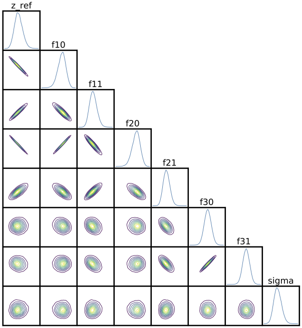




- Model B (right) has smaller residuals

Posterior distributions were sampled with adaptive MCMC





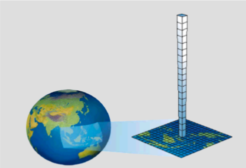
- Well-defined unimodal distributions
- Model B has more dependencies between its parameters

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 may be preferred

Outline

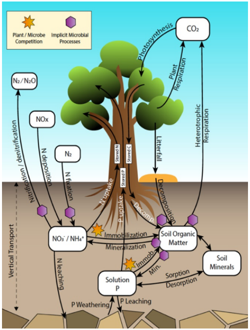
SciDAC BER Partnership Application

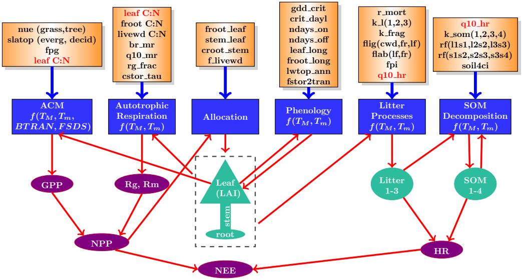


- Optimization of Sensor Networks for Improving Climate Model Predictions
- PI: Daniel Ricciuto at ORNL
- Joint work with MIT FASTMath team (Youssef Marzouk)
- Two applications of UQTk
    - Surrogate models for Global Sensitivity Analysis (GSA) – Cosmin Safta (SNL)
    - Bayesian calibration with model error – Khachik Sargsyan (SNL)

E3SM Land Model (ELM)

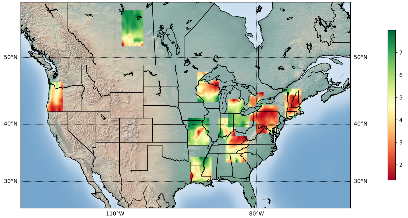
- US Department of Energy (DOE) sponsored Earth system model
- Land, atmosphere, ocean, ice, human system components
- High-resolution, employ DOE leadership-class computing facilities
- Some of the results shown here are with ELM-LF: a lower-fidelity, python version



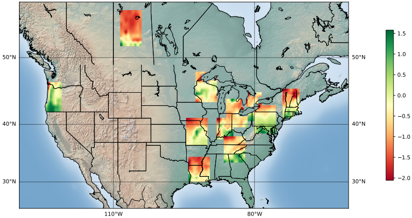


GSA is needed for multiple Quantities of Interest.

Gross Primary Production (GPP, July 2004)



Net Ecosystem Exchange (NEE, July 2004)

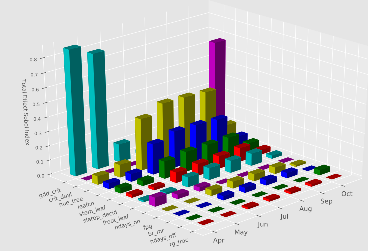


Quantities of Interest

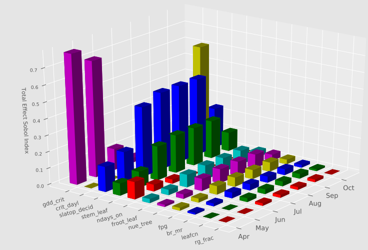
- Gross Primary Production (GPP)
- Total Leaf Area Index (LAI)
- Net Ecosystem Exchange (NEE)
- . . .

Total Effect Sobol' Indices for Model Parameters Relevant at US-Ha1 ($42.5^\circ N$, $72.2^\circ W$)

GPP



LAI

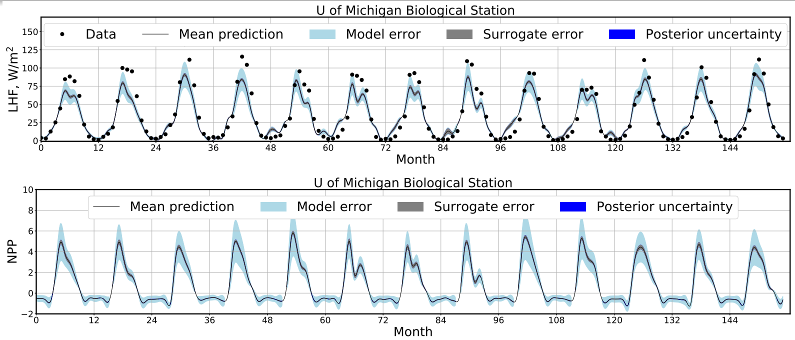


- Sparse regression model accuracy around 10%
- Identified a set of 8-12 parameters (out of 47) that control model outputs of interest.
- Expected time dependencies recovered via sparse regression techniques.

Calibration with *embedded* model error

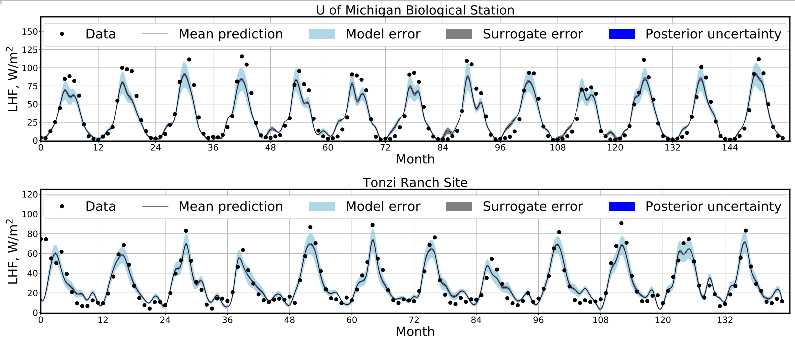
- Model structural error embedding approach [Sargsyan, et al. 2015, 2019]
  - $g(x) = f(x; \lambda + \delta(x)) + \epsilon$
  - Physics-driven model correction
  - Meaningful extrapolation to full set of QoI predictions
  - Disambiguation between model error and data noise
- Simultaneous Bayesian inference of physical parameters and embedded model correction parameters
- Likelihood computation requires uncertainty propagation of embedded stochastic terms
- UQTk provides machinery for both Bayesian inference (adaptive MCMC) and uncertainty propagation via Polynomial Chaos (non-intrusive spectral projection)

ELM calibration with FLUXNET observations



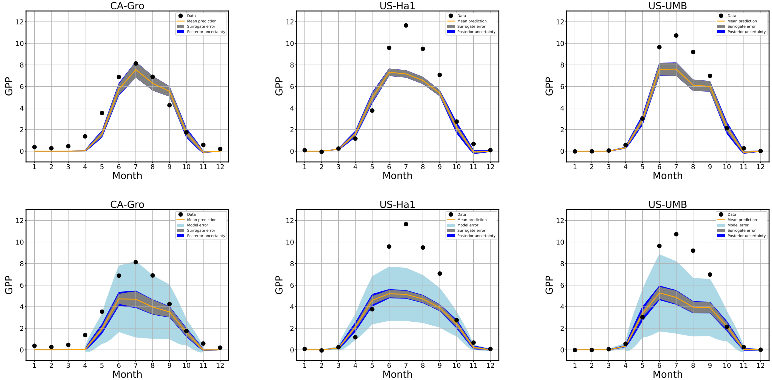
- Predictive variance decomposition with model-error component
- Allows meaningful prediction of other QoIs (e.g. no data/observable)
- Allows (a more dangerous) extrapolation to other sites

ELM calibration with FLUXNET observations



- Predictive variance decomposition with model-error component
- Allows meaningful prediction of other QoIs (e.g. no data/observable)
- Allows (a more dangerous) extrapolation to other sites

ELM-LF calibration with FLUXNET observations



- Embedding removes biases and avoids overfitting
- Model error is the dominant uncertainty component