Algorithms for Sparse and Missing Data

Kenny Chowdhary, John Jakeman, Mike Eldred, Cosmin Safta, Habib Najm
Sandia National Laboratories

www.quest-scidac.org

www.sandia.gov

Support for this work was provided through the Scientific Discovery through Advanced Computing (SciDAC) project funded by the U.S. Department of Energy, Office of Science, Advanced Scientific Computing Research.
Sandia National Laboratories is a multi-program laboratory managed and operated by Sandia Corporation, a wholly owned subsidiary of Lockheed Martin Corporation, for the U.S. Department of Energy's National Nuclear Security Administration under contract DE-AC04-94AL85000.

Algorithmic developments focus on key challenges in extreme scale computing:

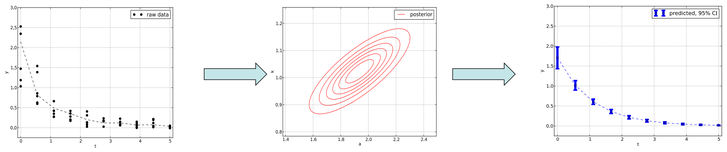
- Data-Free Inference techniques to handle missing data
- Random field representation with sparse sample sizes
- Compressed sensing techniques for approximating high-dimensional uncertainty

Data Free Inference (DFI) with Processed Data Products

Uncertainty propagation typically involves two steps:

1. Estimating the uncertainty in model parameters from direct observations or data (e.g., using Bayesian inference).
2. Propagating this parametric uncertainty (e.g., posterior density) back through the model to obtain predictions.

Example: Fitting rate parameters of elementary reactions



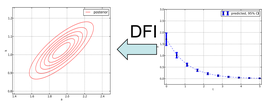
Raw data comes from a double exponential concentration decay model

Step 1: Posterior on the initial concentration and decay rate parameters determined via Bayesian inference

Step 2: Posterior is used to predict elementary concentration levels at difference times

Oftentimes, both steps are done by two different researchers, and **information is lost at the interface**.

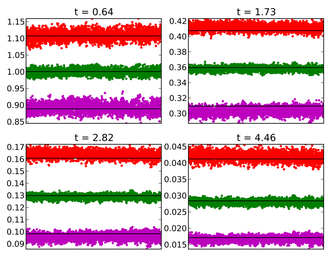
- In particular, we loose information about the uncertainty structure (e.g., posterior density) on model parameters.
- Typically, we also do not have access to the raw data.
- What we do have are processed data products, e.g. error bars on the raw data or predicted values.



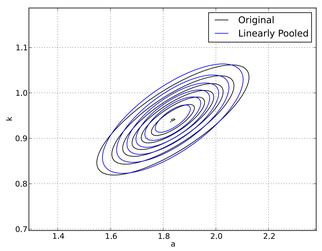
With DFI, we are able to accurately recover the uncertainty structure in the model parameters by exploring the space of surrogate data sets which are consistent with the given processed data products, e.g. moments or quantiles on raw data or predicted quantities.

How does the DFI algorithm work?

- We create an MCMC chain on the data, where each state defines a hypothetical, raw data set
- For each data set, we evaluate the relevant statistics on the posterior and/or the predictive quantities.
- We accept the data set if the relevant statistics are consistent with the given information.
- We then pool (e.g., average) all posteriors for each surrogate data set.



Comparison of summary statistics for hypothetical data sets (shown in colored dots) produced by the DFI algorithm vs. the summary statistics of the original raw data (shown as black lines). Here, the summary statistics are given as means and 2.5-97.5% quantiles on the posterior samples propagated through the model (i.e., pushed forward posterior samples).



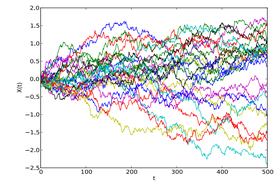
Comparison of averaged (i.e., linearly pooled) posterior density on model parameters for hypothetical data sets produces by the DFI algorithm vs. the posterior calculated from the original, raw data (which is missing).

[1] R. Berry, H. Najm, B. Debusschere, et al., Data-free inference of the joint distribution of uncertain model parameters, J. Comp. Phys., 231 (2012) 2180-2198.
[2] R. Berry, C. Safta, K. Sargsyan, B. Debusschere, Inference of uncertain parameters in chemical models, Int. J. for Uncertainty Quantification.

Random Field Representation with Sparse Sample Sizes

- High dimensional random fields (or random vectors) are prevalent in many applications, e.g. Wiener processes, subsurface flows with uncertain permeability, etc.
- It is convenient and oftentimes necessary to approximate these random fields or vectors with a low-dimensional approximation.
- The **Karhunen-Loeve Expansion (KLE)** allows us to represent a random field or vector as a linear combination of orthogonal and uncorrelated random variables (analogous to a Fourier Series expansion):

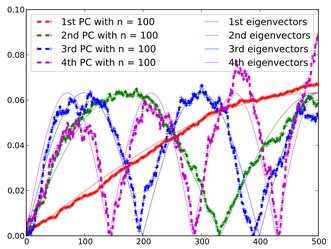
$$X(t) = \sum_{k=1}^{\infty} Z_k e_k(t), \quad \mathrm{corr}(Z_k Z_j) = 0, \quad \langle e_k, e_j \rangle = \delta_{kj}, \quad \forall k, j.$$



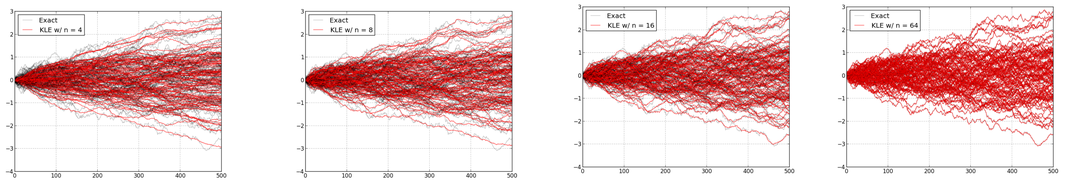
500-dimensional Sample Brownian motion paths

- In the discrete setting, the KLE requires the evaluation of the eigenvectors of the covariance matrix.
- However, if the number of samples used to estimate the covariance matrix is less than the dimensionality of the random vector, the covariance matrix will be singular.
  - In this case, we can use the Singular Value Decomposition (SVD), a.k.a Principle Component Analysis (PCA), to perform the KLE.
  - Each high-dimensional random vector sample can be projected onto to the sub-space spanned by the left-singular values of the samples covariance matrix:
  - We can use these projections as random samples in order to approximate the densities on the KLE coefficients.

$$X(t) \approx \sum_{k=1}^{n} \langle X(t), \phi_k(t) \rangle \phi_k(t), \quad \langle e_k, e_j \rangle = \delta_{kj}, \quad \forall k, j$$



First four principle components using 100 Brownian motion sample paths (of a 500-dimensional random field) plotted against exact eigenvectors. Vectors are shown in absolute value.



Projection of 100 Brownian motion paths onto *n* principle components for *n = 4,16,32,* and *64* (from left to right) using only 100 sample paths to perform the SVD for this 500-dimensional random field.

Compressed Sensing with Cross-Validation for Polynomial Chaos Expansions (PCE)

- Polynomial Chaos is a technique to represent a function of a random variable as a sum of basis polynomials, which in turn depend on the random variables themselves. One can use this polynomial expansion as a surrogate to obtain statistics on the function of interest.
- The following is a truncated polynomial chaos expansion along with the corresponding polynomial basis set over *d* dimensions:

$$f(\boldsymbol{\xi}) \approx \sum_{\psi_i \in \mathcal{A}^d_{p,q}} \alpha_i \psi_i(\boldsymbol{\xi}), \quad (\psi_i, \psi_j)_{L_2(\rho)} = \delta_{i,j}$$

$$\mathcal{A}^d_{p,q} = \{\psi_{\boldsymbol{\lambda}} : \|\boldsymbol{\lambda}\|_q \leq p\}, \quad \boldsymbol{\lambda} = (\lambda_1, \ldots, \lambda_d)$$

- Given a set of function evaluations, **f**, we can find the PCE by solving the following linear system:

$$\begin{bmatrix} \psi_1(\boldsymbol{\xi}_1) & \ldots & \psi_P(\boldsymbol{\xi}_1) \\ \vdots & \ddots & \vdots \\ \psi_1(\boldsymbol{\xi}_m) & \ldots & \psi_P(\boldsymbol{\xi}_m) \end{bmatrix} \begin{bmatrix} \alpha_1 \\ \vdots \\ \alpha_P \end{bmatrix} = \mathbf{A}\boldsymbol{\alpha} = \mathbf{f} + \boldsymbol{\varepsilon} = \begin{bmatrix} f_1 \\ \vdots \\ f_m \end{bmatrix} + \begin{bmatrix} \varepsilon_1 \\ \varepsilon_2 \\ \vdots \\ \varepsilon_N \end{bmatrix}$$

- We elect to solve this linear system via compressed sensing methods, which searches for the sparsest set of coefficients that *best* solves the linear system

Different Compressed Sensing Methods

- Basis Pursuit (BP)

$$\boldsymbol{\alpha} = \arg\min \|\boldsymbol{\alpha}\|_{\ell_1} \quad \text{such that} \quad \mathbf{A}\boldsymbol{\alpha} = \mathbf{f}$$

- Basis Pursuit Denoising (BPDN)

$$\boldsymbol{\alpha} = \arg\min \|\boldsymbol{\alpha}\|_{\ell_1} \quad \text{such that} \quad \|\mathbf{A}\boldsymbol{\alpha} - \mathbf{f}\|_{\ell_2} \leq \varepsilon$$

- Orthogonal Matching Pursuit (OMP)

$$\boldsymbol{\alpha} = \arg\min \|\boldsymbol{\alpha}\|_{\ell_0} \quad \text{such that} \quad \|\mathbf{A}\boldsymbol{\alpha} - \mathbf{f}\|_{\ell_2} \leq \varepsilon$$

- Least Angle Regression (LARS)

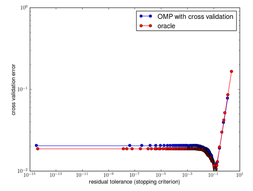
$$\boldsymbol{\alpha} = \arg\min \|\mathbf{A}\boldsymbol{\alpha} - \mathbf{f}\|^2_{\ell_2} \quad \text{such that} \|\boldsymbol{\alpha}\|_{\ell_1} \leq \tau$$

Cross Validation

- To determine the optimal optimization parameters, e.g. tolerances and polynomial index sets, we use K-fold cross-validation techniques.
- K-fold cross validation separates the data into training and test sets.
  1. One performs the optimization on the training set and then evaluates the performance on the testing set.
  2. The optimal parameters are chosen so that they minimize the errors for the test set, i.e. the cross validation error.

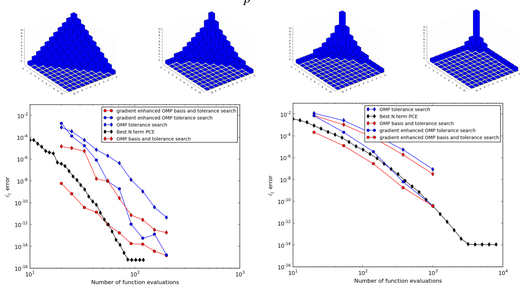
Test Function: $f_{\mathrm{OS}}(\mathbf{x}) = \cos\left(-\sum_{i=1}^{d} c_i \xi_i\right), \quad \boldsymbol{\xi} \in [0,1]^d$

- 20-fold Cross validation error for different tolerance levels for a five-dimensional PCE. The oracle is the true error for the OMP solution at each tolerance level. We are interested in tolerance with the minimum cross validation error.



- We can also perform cross validation over different index sets of polynomials. From left to right, we have *q = 1, .8, .6* and *.4* all (from left to right) with *d = 3* and

$$p = \arg\max_{\hat{p}} \text{ s.t. } |A^d_{p,q}| < 3500$$



- We use cross validation to find the optimal set of index sets and tolerance levels for dimensions 2 (left) and 5 (right).
- For each optimal set, we compute the error in the polynomial approximation using coefficients calculated via OMP as a function of the number of polynomial terms.