SAND2015-XXXX C
SAND2015-4711PE

Exceptional service in the national interest

Sandia National Laboratories

Gaussian Process Models

Laura Swiler

IMA Introduction to Uncertainty Quantification Course

June 17, 2015

U.S. DEPARTMENT OF ENERGY | NNSA National Nuclear Security Administration

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. SAND NO. 2011-XXXXP

Overview

Sandia National Laboratories

- Metamodels
- Gaussian Process background
- Gaussian Process formulation
- Examples
- Use of Gaussian Processes within other methods (e.g. optimization)

Metamodels

Sandia National Laboratories

- Metamodels: also called surrogate, response surface model, or emulator.
- Typically constructed over a small number of simulation model runs ("code runs")
- The simulation is very costly to run, we can only afford a limited number of runs, often dozens to a few hundred
- The code runs provide the training data (e.g. sets of input parameters and corresponding response values)
- The metamodel is constructed to provide a fast, cheap function evaluation for the purposes of uncertainty quantification, sensitivity analysis, and optimization.

- Simpson, T. W., V. Toropov, V. Balabanov, and F.A.C. Viana. Design and analysis of computer experiments in multidisciplinary design optimization: A review of how far we have come or not. In Proceedings of the 12th AIAA/ISSMO Multidisciplinary Analysis and Optimization Conference), Victoria, British Columbia, Canada, September 2008. AIAA Paper 2008-5802.

Metamodels

Sandia National Laboratories

- Taylor series approximations
- Linear regression models
- Neural networks
- Moving least squares
- Radial Basis Functions
- Multivariate Adaptive Regression Splines (MARS)
- Gaussian process models
- Polynomial Chaos expansions
- Multi-fidelity models
- Reduced-order models

Gaussian Processes

Sandia National Laboratories

- Why are GPs popular emulators of computer models?
  - They allow modeling of fairly complicated functional forms
  - They do not just offer a prediction at a new point but an estimate of the uncertainty in that prediction

- Classic references:
  - Sacks, J., W.J. Welch, T.J. Mitchell, and H.P. Wynn. Design and analysis of computer experiments. Statistical Science, 4(4):409–435, 1989.
  - Santner, T., B. Williams, and W. Notz. The Design and Analysis of Computer Experiments. New York, NY: Springer, 2003.
  - Rasmussen, C.E. and C.K.I. Williams. *Gaussian Processes for Machine Learning*. MIT Press, 2006. e-book:
    - http://www.gaussianprocess.org/gpml/chapters/

Gaussian Process

- A stochastic process is a collection of random variables $\{y(\mathbf{x}) \mid \mathbf{x} \in \mathbf{X}\}$ indexed by a set $\mathbf{X}$ in $\Re^d$, where d is the number of inputs.
- A Gaussian process is a stochastic process for which any finite set of y-variables has a joint multivariate Gaussian distribution. That is, the joint probability distribution for every finite subset of variables $y(\mathbf{x}_1), ..y(\mathbf{x}_k)$ is multi-variate normal.
- A GP is fully specified by its mean function $\mu(\mathbf{x}) = E[y(\mathbf{x})]$ and its covariance function $C(\mathbf{x}, \mathbf{x}')$.

What does this mean?

Sandia National Laboratories

- Start with a set of runs of a computer code: at each sample $\mathbf{x}_i$ we have output $y_i(\mathbf{x}_i)$.
- The output at a new input value, $\mathbf{x}_{new}$, is uncertain.
- This is what a GP will predict.
- Related to regression.
- Related to random functions. From our set of samples, we have a "deterministic" function that is a set of points $\{\mathbf{x}, y(\mathbf{x})\}$ or $\{\mathbf{x}, f(\mathbf{x})\}$. Instead of $f(\mathbf{x})$, if we use the outcome of a random draw from some joint distribution of random variables $\{Z(\mathbf{x}_1), \ldots Z(\mathbf{x}_n)\}$, we get a realization of a random function.
- This is a stochastic process (e.g. generate many draws and get many functions).

How do we simulate realizations of a random function?

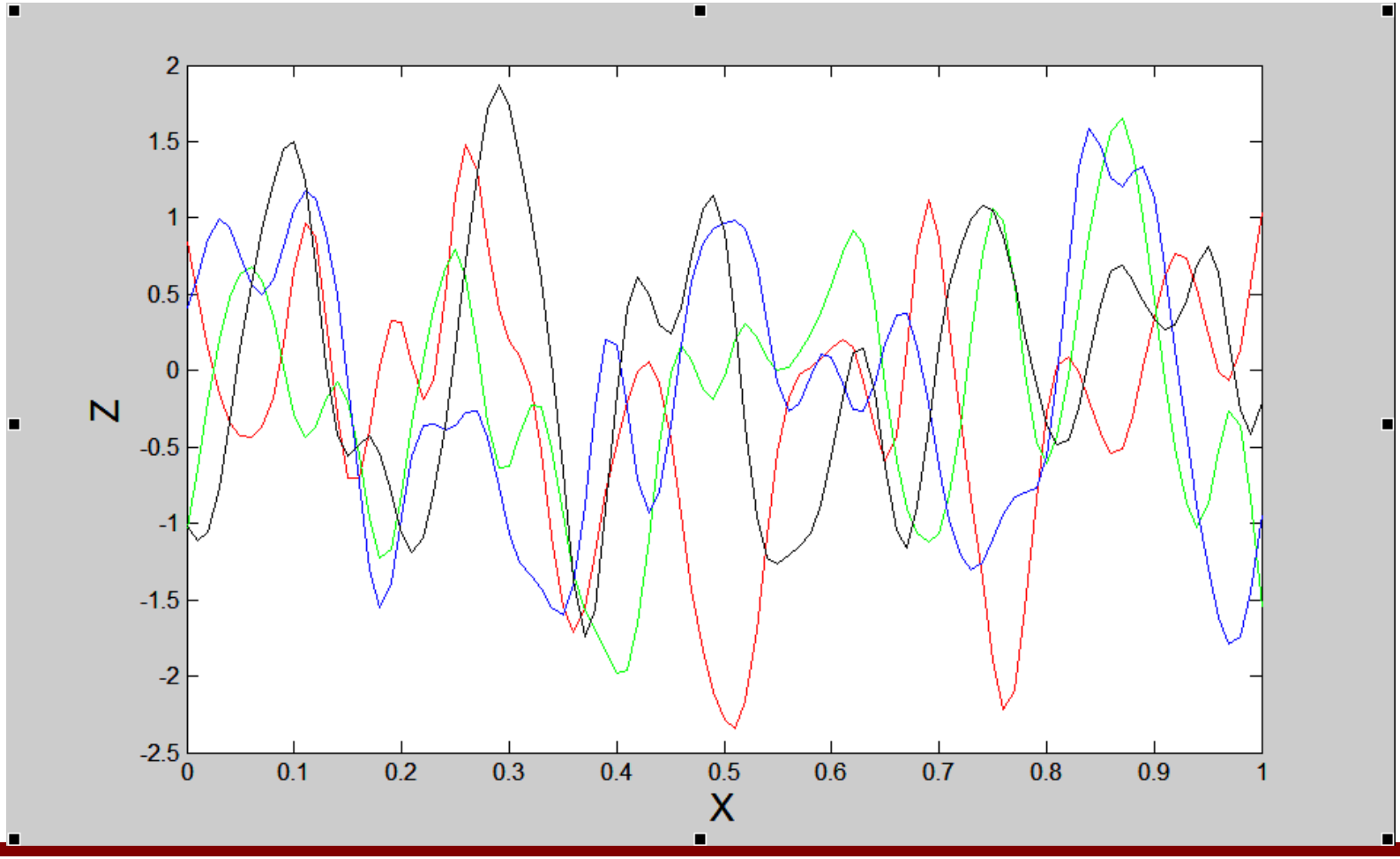
Sandia National Laboratories

- Start with $\{Z(\mathbf{x}_1), \ldots Z(\mathbf{x}_n)\}$ from a multivariate normal distribution with mean 0 and covariance matrix $\mathbf{C}$=Cov$[Z(\mathbf{x}_i), Z(\mathbf{x}_j)]$.
- To simulate a random draw:
  - Generate *n* standard normal(0,1) random variables, **S**.
  - Perform a Cholesky decomposition $\mathbf{C= LL^t}$.
  - Define **Z = LS**.
  - Plot the points $\{\mathbf{x}_i, Z_i = Z(\mathbf{x}_i)\}$
  - Connect the dots

Example covariance function in 1-D

Sandia National Laboratories

- $\mathrm{Cov}[Z(x_i), Z(x_j)] = \exp(-\theta |x_i - x_j|^2)$



Gaussian Process

Sandia National Laboratories

- We have the capability to generate random functions
- We can add a mean function (typically a constant or a simple polynomial regression)
- We can multiply the covariance by a constant to scale the vertical axis.
- Now, we can vary θ to get a certain amount of "wiggle" in the random function (smaller θ leads to less wiggle).
- **NOW: we want to constrain these random functions to be consistent with the data points we have**
- We can either take a Bayesian approach or a maximum likelihood (MLE) approach to estimate the parameters governing the Gaussian process
- Start with a MLE approach

Gaussian Process

- Typical formulation: a Gaussian process is defined by its mean and covariance function. We assume:

$$\mathrm{E}[y(\boldsymbol{x})] = f(\boldsymbol{x})^T\boldsymbol{\beta} \qquad \text{Mean}$$

$$\mathrm{Cov}[y(\boldsymbol{x}), y(\boldsymbol{x}')] = \sigma^2 r(\boldsymbol{x}, \boldsymbol{x}') \qquad \text{Covariance}$$

$$\boldsymbol{Y} \sim N(f(\boldsymbol{X})^T\boldsymbol{\beta}, \sigma^2\boldsymbol{R}) \qquad \text{Multivariate Normal}$$

- A few notes:
  - $\boldsymbol{x}$ is one set of inputs of dimension *d*. We have N samples, $\boldsymbol{x}_i$, for i=1…N. Each $\boldsymbol{x}_i = \{x_{i1}, x_{i2}, \ldots, x_{id}\}$. $\boldsymbol{X}$ denotes the (d*N) set of all samples, and $\boldsymbol{\beta}$ is the d*1 vector of regression coefficients. It may just be a constant β.
  - It is more typical to write the covariance as the product of a scaling factor σ^2 times the correlation $r(\boldsymbol{x}, \boldsymbol{x}')$.
  - The full $\mathrm{N} * \mathrm{N}$ correlation matrix between all points is $\boldsymbol{R}$
  - $\boldsymbol{Y}$ is the (N*1) vector of response values.

Gaussian Process

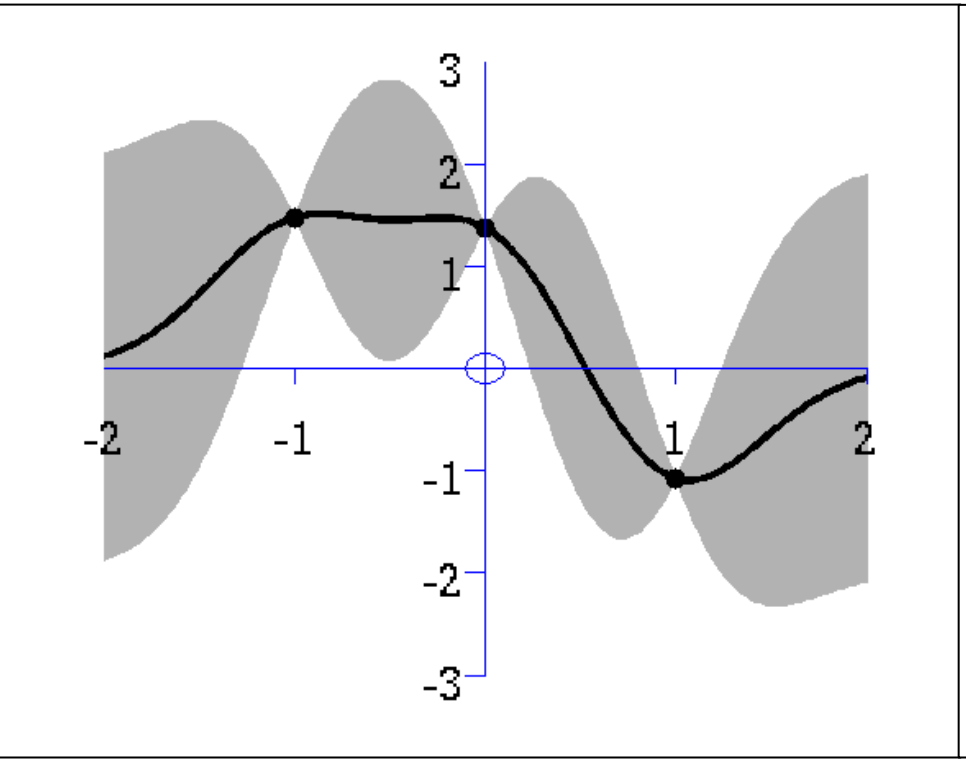
- NOW: what is the prediction for a new point?

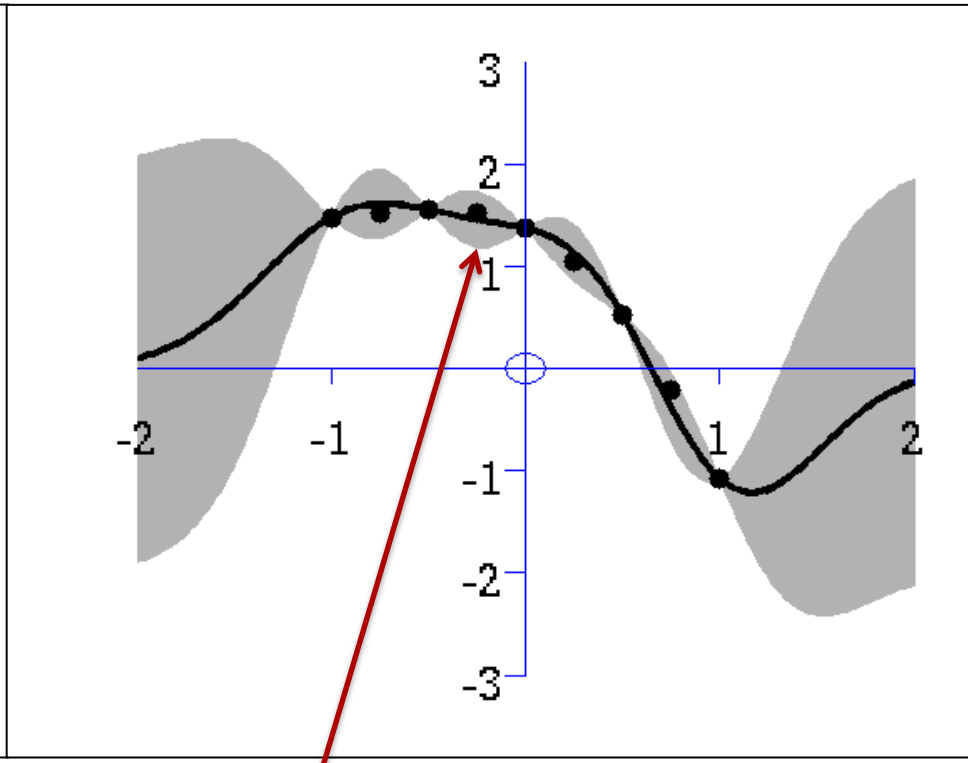
$$\mathrm{E}[y(\boldsymbol{x}^*)|\boldsymbol{Y}] = f(\boldsymbol{x}^*)^T\boldsymbol{\beta} + r(\boldsymbol{x}^*)^T\mathbf{R}^{-1}[\mathbf{Y} - \mathbf{F}\boldsymbol{\beta}]$$

$$\mathrm{Var}[y(\boldsymbol{x}^*)|\boldsymbol{Y}] = \sigma^2(1 - r(\boldsymbol{x}^*)^T\mathbf{R}^{-1}r(\boldsymbol{x}^*))$$

- The correlation matrix for the training points is $\boldsymbol{R}$.
- $r(\boldsymbol{x}^*)$ is the vector of correlations between the new point $\boldsymbol{x}^*$ and the existing N points. It is of size N*1.
- **F** is the set of basis functions for the original full data set **X**.
- These are the ***conditional predictions*** (conditional on the data).

What does this look like?

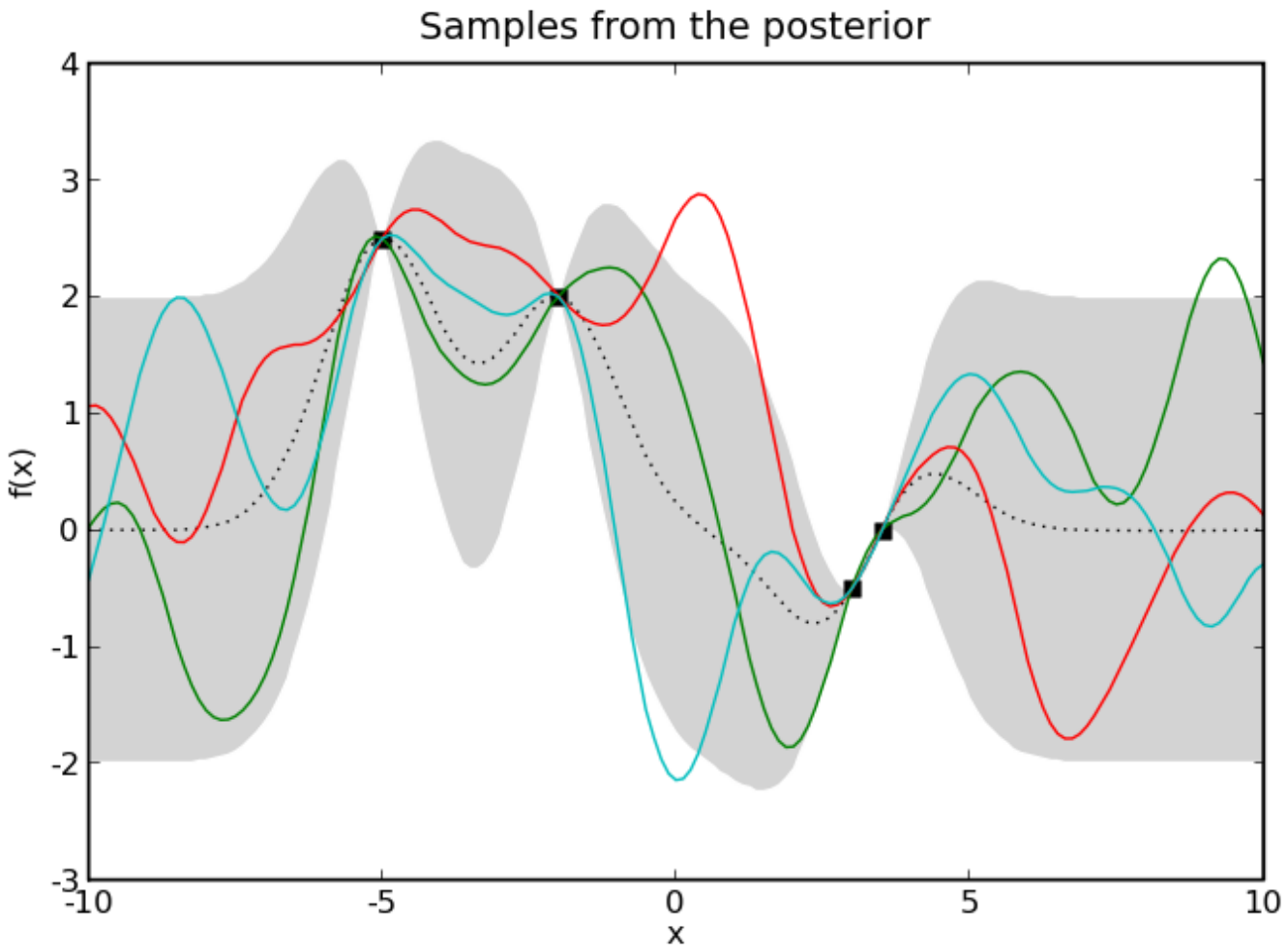




staffwww.dcs.shef.ac.uk

Note the reduction in variance as you have more data

What does this look like?



https://pythonhosted.org/infpy/gps.html

This plot shows mean and variance plus random realizations

Properties of the GP approximation

- The mean prediction interpolates the data.

$$\mathrm{E}[y(\boldsymbol{x}^*)|\boldsymbol{Y}] = f(\boldsymbol{x}^*)^T\boldsymbol{\beta} + r(\boldsymbol{x}^*)^T\mathbf{R}^{-1}[\mathbf{Y} - \mathbf{F}\boldsymbol{\beta}]$$

- The mean prediction is a linear combination of basis functions
- The predicted variance increases the further away the new point is from existing points.

$$\mathrm{Var}[y(\boldsymbol{x}^*)|\boldsymbol{Y}] = \sigma^2(1 - r(\boldsymbol{x}^*)^T\mathbf{R}^{-1}r(\boldsymbol{x}^*))$$

Correlation Function

- Want to capture the idea that nearby inputs have highly correlated outputs.
- The correlation in some dimensions may be more important than others…different "length-scales" in each dimension
- Common correlation functions include

Power-exponential:

- Typically the exponent p_j is 2, which gives smooth realizations. If p_j is 1, you get much rougher realizations.
- Larger values of θ_j mean smaller correlation in the x_j direction.

$$R(\boldsymbol{x}, \boldsymbol{x}') = \exp\{-\sum_{j=1}^{d} \theta_j (x_j - x_j')^{p_j}\} = \prod_{j=1}^{d} \exp(-\theta_j (x_j - x_j')^{p_j})$$

Matern, Cubic

Putting it all together

- Start with N runs of a computer code, with points $\{\mathbf{x}_i, y_i\}$. Ideally, the N points will be a well-spaced design such as Latin Hypercube.
- Define the mean function for the Gaussian process.
  - Often, zero mean or constant mean is used.
- Define the covariance function for the Gaussian process.
  - Typically, the power-exponential function is used.
- Estimate the parameters governing the Gaussian process, including $\boldsymbol{\beta}$, σ, and any parameters of the correlation function R such as θ_j.
  - Can use maximum likelihood or Bayesian methods
- Substitute the parameters in the prediction equations and obtain mean and variance estimates for new points $\mathbf{x}^*$

Parameter Estimation (MLE)

- The observed training values represent a realization of a multivariate normal distribution.

$$f(\boldsymbol{Y}) = (2\pi)^{-\frac{N}{2}}|\Sigma|^{-\frac{1}{2}}exp\left[-\frac{1}{2}(\boldsymbol{Y}-\boldsymbol{\mu})^T\Sigma^{-1}(\boldsymbol{Y}-\boldsymbol{\mu})\right]$$

- The basic idea of MLE is to find the particular mean vector and covariance matrix that define the most likely multivariate normal distribution to result in the observed data.
- Take the Log Likelihood and maximize it:
- $\log(f(\boldsymbol{Y})) = -\frac{N}{2}\log(2\pi) - \frac{1}{2}(\sigma^{2N}|\boldsymbol{R}|) - \frac{1}{2\sigma^2}(\boldsymbol{Y}-\boldsymbol{F\beta})^T\,\boldsymbol{R}^{-1}(\boldsymbol{Y}-\boldsymbol{F\beta})$
- Drop the -1/2 term, and the first constant term and minimize the negative log-likelihood:
- $NLL = \mathrm{N}\log(\sigma^2) + \log(|R|) + \frac{1}{\sigma^2}(\boldsymbol{Y}-\boldsymbol{F\beta})^T\,\boldsymbol{R}^{-1}(\boldsymbol{Y}-\boldsymbol{F\beta})$

Parameter Estimation (MLE)

- Use global optimization methods to optimize the NLL

  OR

- Use gradient-based optimization to optimize the NLL. The derivations have been worked out with respect to **β**, **σ**, and correlation parameters of R.
- Conditional on fixed values of the correlation parameters, the optimal values for **β** and **σ** are given by the generalized least squares formulation:

$$\widehat{\boldsymbol{\beta}} = (\boldsymbol{F}^{\boldsymbol{T}}\boldsymbol{R}^{-\boldsymbol{1}}\boldsymbol{F})^{-\boldsymbol{1}}(\boldsymbol{F}^{\boldsymbol{T}}\boldsymbol{R}^{-\boldsymbol{1}}\boldsymbol{Y})$$

$$\widehat{\sigma^2} = \frac{1}{N}(\boldsymbol{Y} - \boldsymbol{F\beta})^T\,\boldsymbol{R}^{-1}(\boldsymbol{Y} - \boldsymbol{F\beta})$$

- One can use an iterative method, and obtain optimal correlation parameters **θ**, then calculate **R** and substitute it into above expressions above for **β** and **σ.**
- This optimization has been studied fairly thoroughly. A good reference is:

  Jay Martin. "Computational Improvements to Estimating Kriging Metamodel Parameters." Journal of Mechanical Design. Aug. 2009, Vol. 131, p. 084501:1-7.

Bayesian parameter estimation

- Denote all of the parameters governing the GP as:
  - $\Theta = (\beta, \sigma, \theta_j)$.
- Bayesian approach to estimate posterior distribution on hyperparameters Θ:

$$\pi(\Theta \mid \boldsymbol{X}, Y) \propto \pi(\Theta) L(\boldsymbol{X}, Y \mid \Theta)$$

- Likelihood is the same as before with MLE
- Use Markov Chain Monte Carlo (MCMC) to solve it
- Requires thousands of evaluations of the likelihood function
- Large amount of work done in the statistical community about priors on these parameters, estimation of marginal likelihoods.
- Jeffreys-independent prior, reference priors, are often assumed
- Need to be careful that priors are not improper
- Reference: Paulo, Rui. Default priors for Gaussian processes. Ann. Statist. 33 (2005), no. 2, 556--582. doi:10.1214/009053604000001264.

Experimental Design

- The training set of $\{\mathbf{x}_i\}$ points, i= 1…N is usually a space-filling design such as a Latin Hypercube design or a maxi-min LHS
- Want the points to be well spaced
- Don't want highly collinear points (close together)
- PROBLEM:
  - The prediction calculations require the inversion of the correlation matrix
  - Often the correlation matrix is ill-conditioned and may be numerically singular
  - Happens even with a few hundred points in 2-D
  - One can't invert **R** to use in the prediction calculation

Techniques to handle ill-conditioning of the correlation matrix

- Remove points in a random or structured way ("Sparsification")
- Often, a small "jitter" or noise term σ_ϵ is added to the diagonal terms of the covariance matrix to make the matrix better conditioned.

$$C = \sigma^2 R \rightarrow C = \sigma^2 R + {\sigma_\epsilon}^2 I,$$

- Adding a nugget term
  - Estimate the nugget as part of the measurement error
  - Fix the measurement error and **add a nugget**, may have to do this iteratively until the nugget is big enough to make R well-conditioned

Techniques to handle ill-conditioning of the correlation matrix (cont'd)

- Linear algebra tricks
  - Don't take the inverse of **R**, take the Cholesky factorization
  - Pseudo-inverse
    - Discards small singular values
  - Pivoted Cholesky Factorization
    - discard additional copies of the information that is most duplicated
    - Decrease the maximum eigenvalue and increase the minimum eigenvalue
  - Gradient-enhanced kriging
- SAND Report 2013-7022. *Efficient and Robust Gradient Enhanced Kriging Emulators*, by Keith Dalbey.

Sandia National Laboratories

Software and Resources

Resources

- Websites: www.gaussianprocess.org
- Managing Uncertainty in Computer Models (MUCM):
  - UK project headed by Prof. Tony O'Hagan, University of Sheffield
  - http://www.mucm.ac.uk/Pages/ReadingList.html
- Books:
  - *Gaussian Processes for Machine Learning,* Carl Edward Rasmussen and Chris Williams, MIT Press, 2006.
  - *Statistics for Spatial Data*, Noel A. C. Cressie, Wiley, 1993.
  - *The Design and Analysis of Computer Experiments*. Santner, T., B. Williams, and W. Notz. Springer, 2003.

Software and Resources

Sandia National Laboratories

- Software:
  - R: tgp (Gramacy and Lee), gptk (Kalaitzis, Lawrence, et al.), GPfit MacDonald, Chipman, and Ranjan)
  - Matlab: gpml (Rasmussen, Williams, Nickisch), GPmat (Sheffield Group)
  - Python: scikit-learn. http://scikit-learn.org/stable/
  - Python: GPy, gptools, pyGPs, etc.
  - C++: https://github.com/mblum/libgp
  - MIT Group: MUQ/GPEXP (Python)
  - Dakota/Surfpack (C++)
  - Lots of others….

Sandia National Laboratories

Example Use Cases

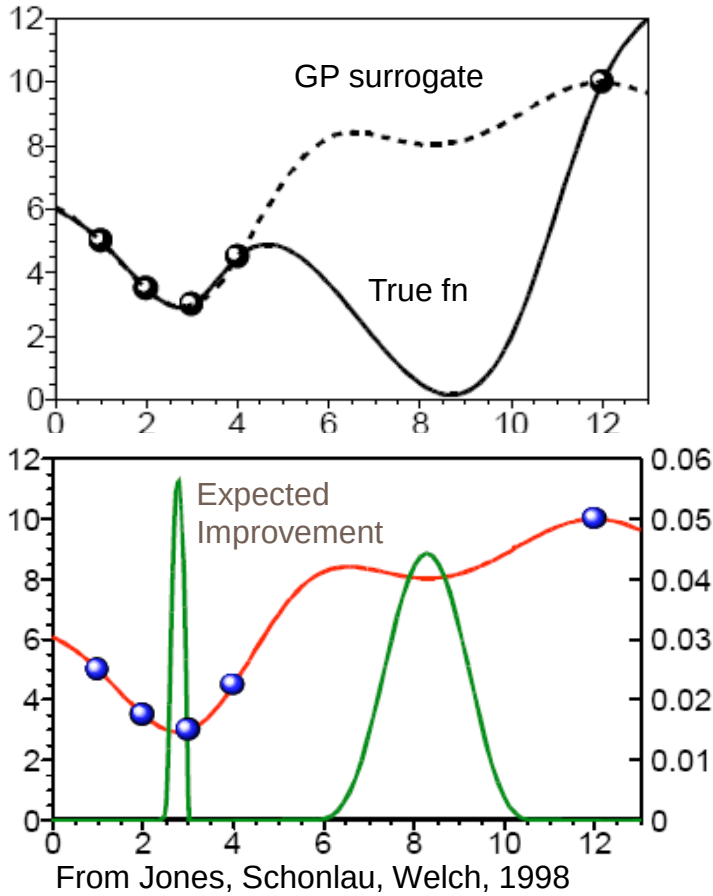
Efficient Global Optimization

- Technique due to Jones, Schonlau, Welch
- Build global Gaussian process approximation to initial sample
- Balance global exploration (add points with high predicted variance) with local optimality (promising minima) via an "expected improvement function"

$$\mathrm{E}[I(\mathbf{x})] \equiv \mathrm{E}\left[\max(f_{\min} - Y, 0)\right]$$

$$\mathrm{E}[I(\mathbf{x})] = (f_{\min} - \hat{y})\Phi\left(\frac{f_{\min} - \hat{y}}{s}\right) + s\phi\left(\frac{f_{\min} - \hat{y}}{s}\right)$$

- Iteratively add points that have maximized EI, we use a DIRECT global optimization algorithm to identify that point
- Derivative-free, very efficient for low-dim.



From Jones, Schonlau, Welch, 1998

Efficient Global Reliability Analysis (EGRA)

Sandia National Laboratories

- Reliability methods find “failure surface” or “limit state contour” between “safe” and “failure” regions, often defined as g(**x**)=0
- Integral of the probability density of the inputs over the failure region is the probability of failure

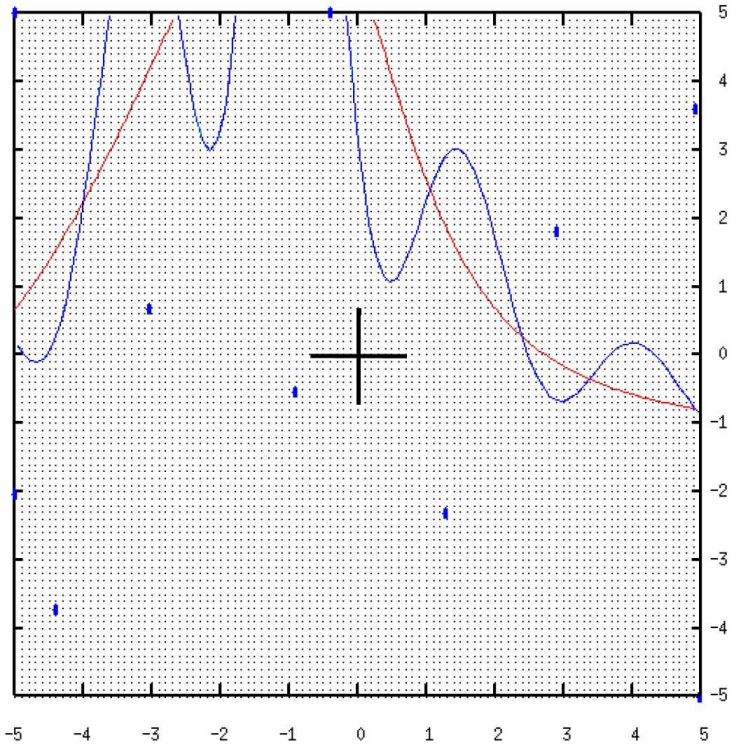
$$p_f = \int \ldots \int_{g()<0} f_X(x_1, x_2, \ldots, x_n) dx_1 dx_2 \ldots dx_n$$

- Local reliability methods have problems with the nonsmooth, multimodal, and highly nonlinear failure surfaces
- EGRA is a global reliability analysis that uses a variant of EGO
  - The expected improvement is now the expected feasibility: penalize points from being away from the g(x)=0 boundary
  - Balance explore and exploit in locating the limit state (EGRA)
  - Handles nonsmooth, multi-modal, highly nonlinear response functions

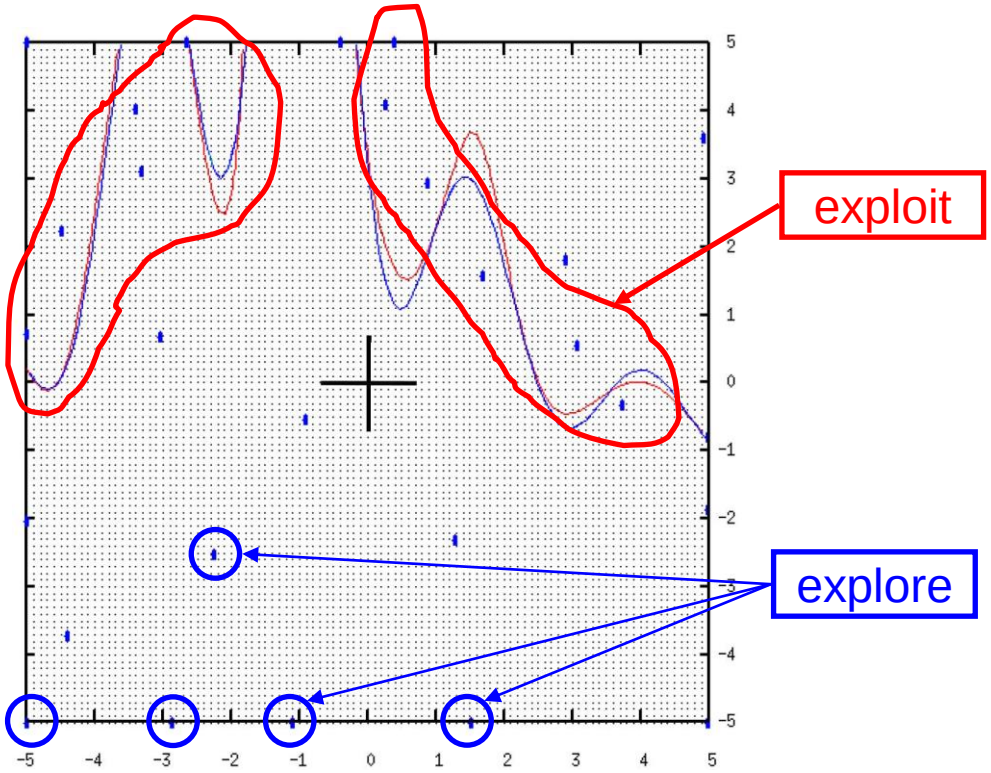
Efficient Global Reliability Analysis

Gaussian process model of reliability limit state with

10 samples

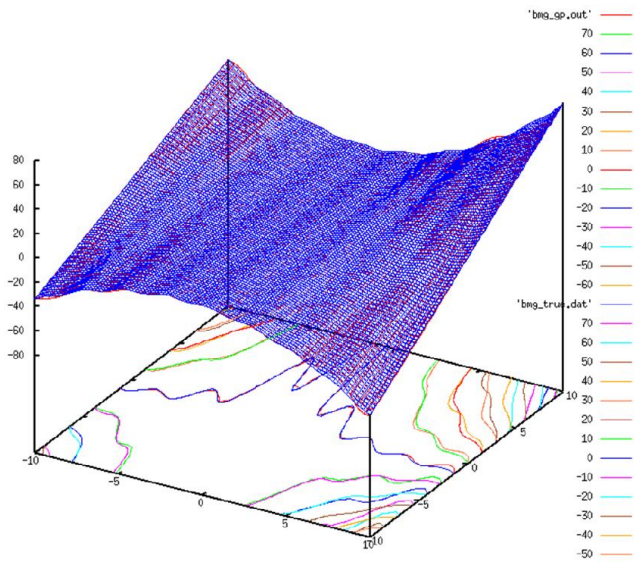


28 samples

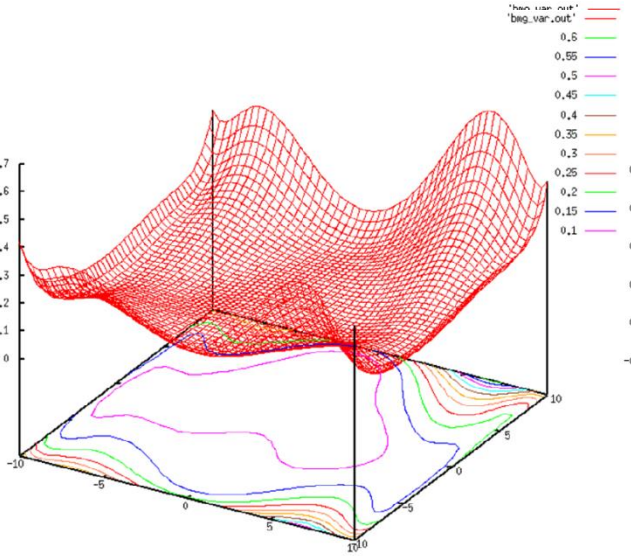


Efficient Global Reliability Analysis

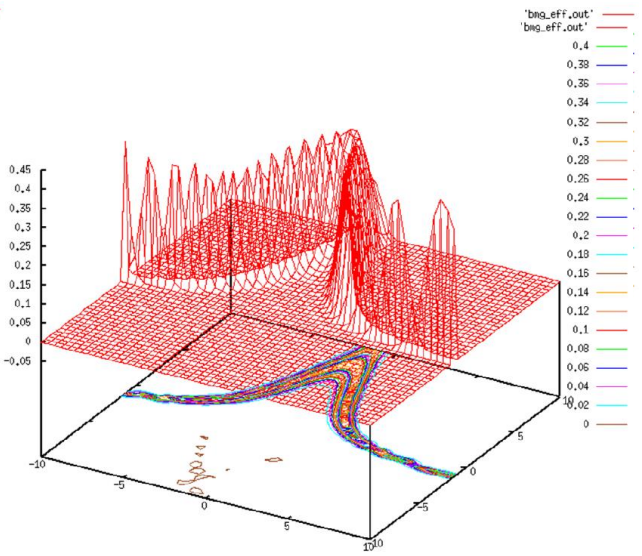
Mean



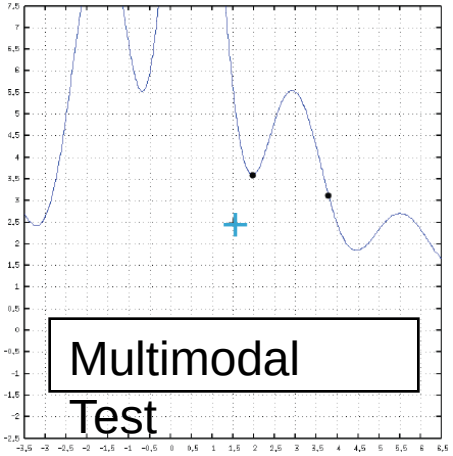
Variance



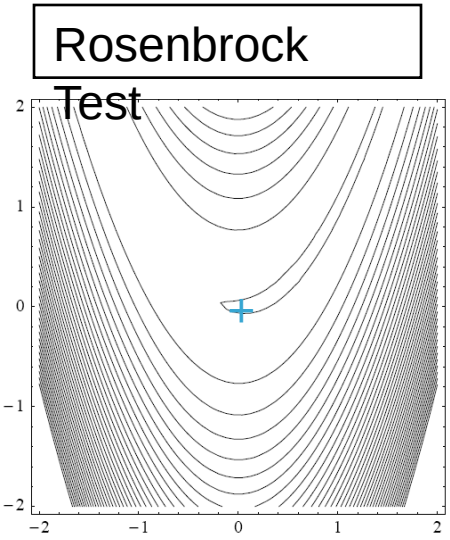
Expected Feasibility

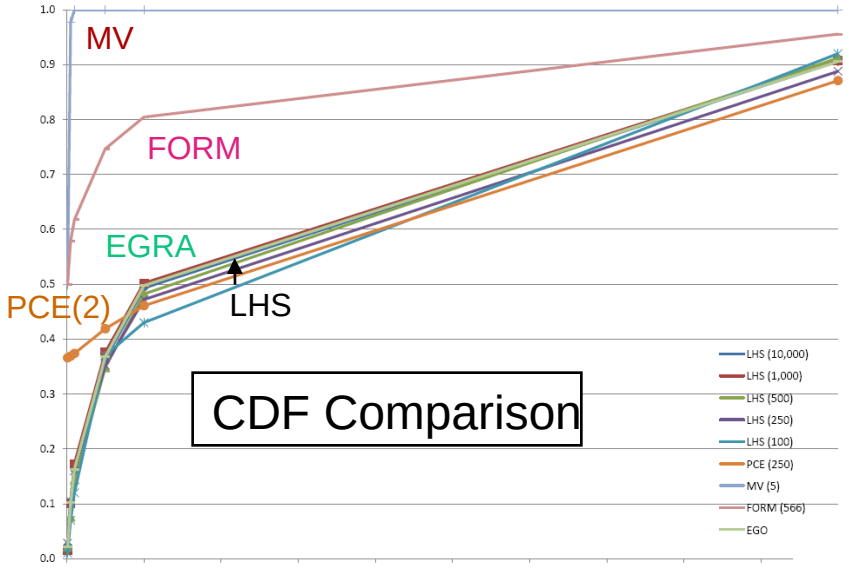


EGRA: Benchmark performance



Reliability Method	Function Evaluations	First-Order p_f (% Error)	Second-Order p_f (% Error)	Sampling p_f (% Error, Avg. Error)
No Approximation	70	0.11797 (277.0%)	0.02516 (-19.6%)	—
x-space AMV^2+	26	0.11797 (277.0%)	0.02516 (-19.6%)	—
u-space AMV^2+	26	0.11777 (277.0%)	0.02516 (-19.6%)	—
u-space TANA	131	0.11797 (277.0%)	0.02516 (-19.6%)	—
LHS solution	10k	—	—	0.03117 (0.385%, 2.847%)
LHS solution	100k	—	—	0.03126 (0.085%, 1.397%)
LHS solution	1M	—	—	0.03129 (truth , 0.339%)
x-space EGRA	35.1	—	—	0.03134 (0.155%, 0.433%)
u-space EGRA	35.2	—	—	0.03133 (0.136%, 0.296%)



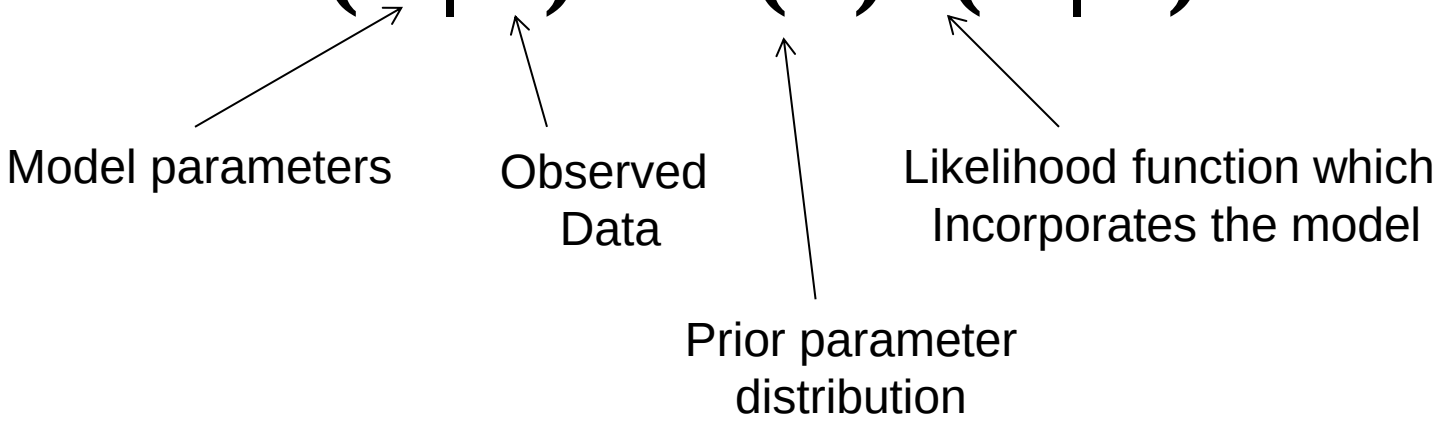


Accuracy similar to exhaustive sampling at cost similar to local reliability assessment

Bayesian Formulation

- Generate posterior distributions on model parameters, given
  - Experimental data
  - A prior distribution on model parameters
  - A presumed probabilistic relationship between experimental data and model output that can be defined by a likelihood function

$$\pi(\theta \mid d) \propto \pi(\theta) L(d \mid \theta)$$



Bayesian Calibration of Computer Models

Sandia National Laboratories

- Experimental data = Model output + error

$$d_i = M(\boldsymbol{\theta}, \mathbf{x}_i) + \varepsilon_i$$

- If we assume error terms are independent, zero mean Gaussian random variables with variance σ^2, the likelihood is:

$$L(\boldsymbol{\theta}) = \prod_{i=1}^{n} \frac{1}{\sigma\sqrt{2\pi}} \exp\left[-\frac{(d_i - M(\boldsymbol{\theta}, \mathbf{x}_i))^2}{2\sigma^2} \right]$$

- How do we obtain the posterior?
  - It is usually too difficult to calculate analytically
  - We use a technique called Markov Chain Monte Carlo (MCMC)
  - In MCMC, the idea is to *generate a sampling density that is approximately equal to the posterior.* We want the sampling density to be the stationary distribution of a Markov chain.

Bayesian Calibration: Approach

- Take initial set of samples from simulation
  - Use LHS or space-filling design
- Develop Gaussian process approximation of the simulation
- Put priors on the input parameters
- Perform Bayesian analysis using MCMC
- Generate and analyze posterior distributions
- NOTE: GP surrogate adds a layer of uncertainty. However, this is explicitly modeled in the revised likelihood:

$$L(\boldsymbol{\theta}) = 2\pi^{-n/2}\left|\Sigma\right|^{-1/2}\exp\left[-\frac{1}{2}(d_i-\mu_{GP})^T\Sigma^{-1}(d_i-\mu_{GP})\right]$$

$$\Sigma = \sigma^2 I + \Sigma_{GP}$$

- Total uncertainty = (observation + model uncertainty) + surrogate uncertainty

Acknowledgments

- Summer School on the Design and Analysis of Computer Experiments, Aug. 11-15, 2006. Part of a SAMSI Program, taught in part by Jerry Sacks and Will Welch.
- John McFarland (now at Southwest Research Institute)
  - Dissertation: UNCERTAINTY ANALYSIS FOR COMPUTER SIMULATIONS THROUGH VALIDATION AND CALIBRATION. Vanderbilt University, May 2008.
- Discussions with Tony O'Hagan (University of Sheffield) over the years.
- Many discussion with Sandians, including Brian Rutherford, Patty Hough, Brian Adams, Mike Eldred, and Keith Dalbey.
- Discussions with Brian Williams at Los Alamos.

Sandia National Laboratories

THANK YOU!
QUESTIONS?