SAND2015-6866 TR

Exceptional service in the national interest

Dakota Software Training

Surrogate Models

http://dakota.sandia.gov

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.

Module Learning Goals

- Define a surrogate model
- Identify situations where it may be appropriate to use a surrogate model
- Learn how to specify a surrogate model in Dakota
- Run a surrogate model in Dakota and examine outputs based on the surrogate model
- Identify some common diagnostics for surrogates
- Understand different ways surrogates are used in Dakota

Module Learning Goals

- Define a surrogate model
- Identify situations where it may be appropriate to use a surrogate model
- Learn how to specify a surrogate model in Dakota
- Run a surrogate model in Dakota and examine outputs based on the surrogate model
- Identify some common diagnostics for surrogates
- Understand different ways surrogates are used in Dakota

Surrogate Models

Surrogate model: an inexpensive parameter to response mapping (meta-model) that replaces a simulation code run

Motivation:

- Often can only afford limited code runs: dozens to 100s
- Constructed to provide a fast, cheap function evaluation for purposes of uncertainty quantification, sensitivity analysis, and optimization.
- Can smooth a noisy response

Broad classes of surrogates:

- Data-fit: response surface model or emulator constructed to fit a small number of code runs
- Multi-fidelity, e.g., low-fidelity physics approximating a high-fidelity model
- Reduced-order model (ROM, snapshot POD, PCA)

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.

Data-Fit Surrogate Models

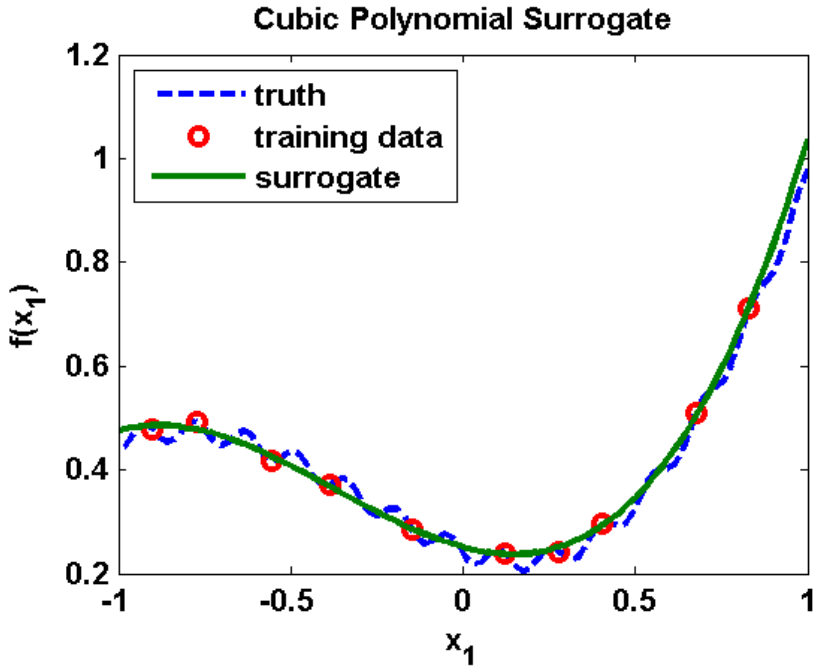
- Also called response surface model, or emulator
- Typically constructed over a small number of simulation model runs, generated using a design of computer experiments
- The code runs provide the training data (samples of input parameters and corresponding response values)

Example: Polynomial surrogate model

- Regression fit to training data
- Accurate in small regions
- Good at smoothing noisy data

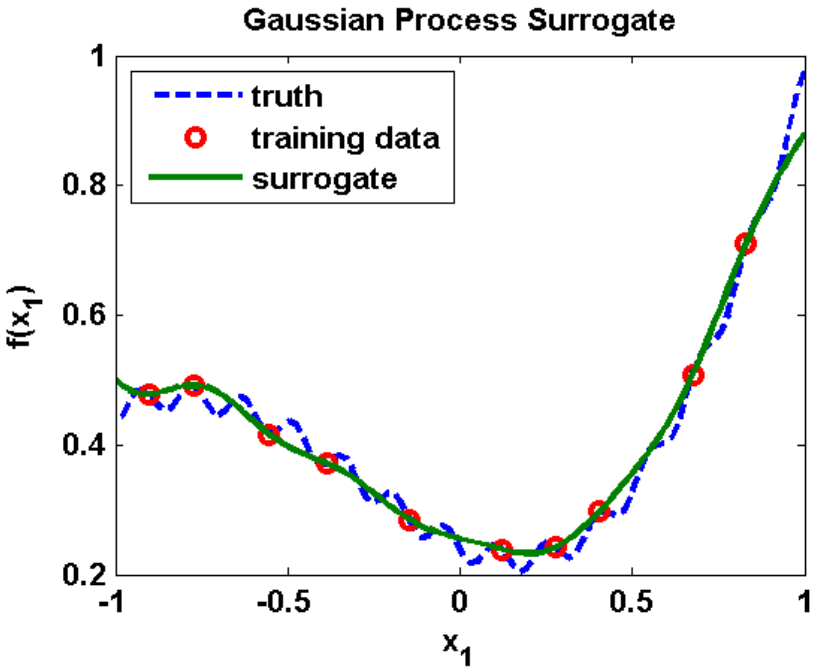
Linear: $$\hat{f}(\mathbf{x}) \approx c_0 + \sum_{i=1}^{n} c_i x_i$$

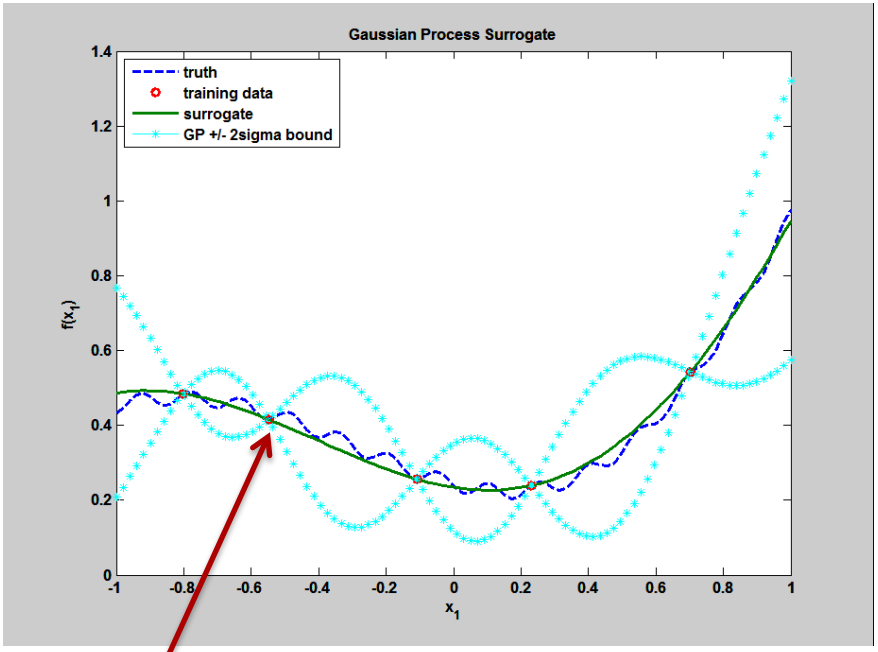
Quadratic: $$\hat{f}(\mathbf{x}) \approx c_0 + \sum_{i=1}^{n} c_i x_i + \sum_{i=1}^{n} \sum_{j \geq i}^{n} c_{ij} x_i x_j$$



Gaussian Process Models

- Also known as Kriging models, they typically interpolate data
- Popular surrogates of computer models in last 15 years
- Allow modeling of fairly complicated functional forms
- Offer both a prediction at a new point and an estimate of the uncertainty in that prediction





Note: reduction in variance closer to training data points

Other Dakota Data-fit Models

- Taylor series approximations
  - Provides local trends in the vicinity of a single point

$$\hat{f}(\mathbf{x}) \approx f(\mathbf{x}_0) + \nabla_{\mathbf{x}} f(\mathbf{x}_0)^T (\mathbf{x} - \mathbf{x}_0)$$

- Multivariate Adaptive Regression Splines (MARS)
  - Splines can represent complex multi-modal surfaces and smooth noisy data.

$$\hat{f}(\mathbf{x}) = \sum_{m=1}^{M} a_m B_m(\mathbf{x}) \quad \longleftarrow \text{basis functions}$$

- Artificial neural networks (ANN)
- Moving least squares (MLS)
- Radial basis functions (RBF)
- Emerging capability: piecewise-local versions of the above

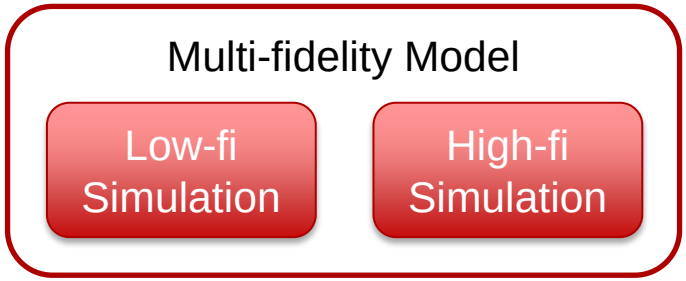
Other Dakota Surrogate Models

- Polynomial chaos expansions (PCE)
  - PCEs have also become popular surrogates of computer models over the last 15 years
  - Response approximation based on multidimensional orthogonal polynomials, tailored to the uncertain input distributions
  - See uncertainty quantification module…

- Multi-fidelity models
  - Typically assume high fidelity equals low fidelity plus a correction

$$\hat{f}_{hi_\alpha}(\mathbf{x}) = f_{lo}(\mathbf{x}) + \alpha(\mathbf{x})$$

  - Dakota will coordinate execution of the low and high fidelity models to meet an optimization or UQ goal



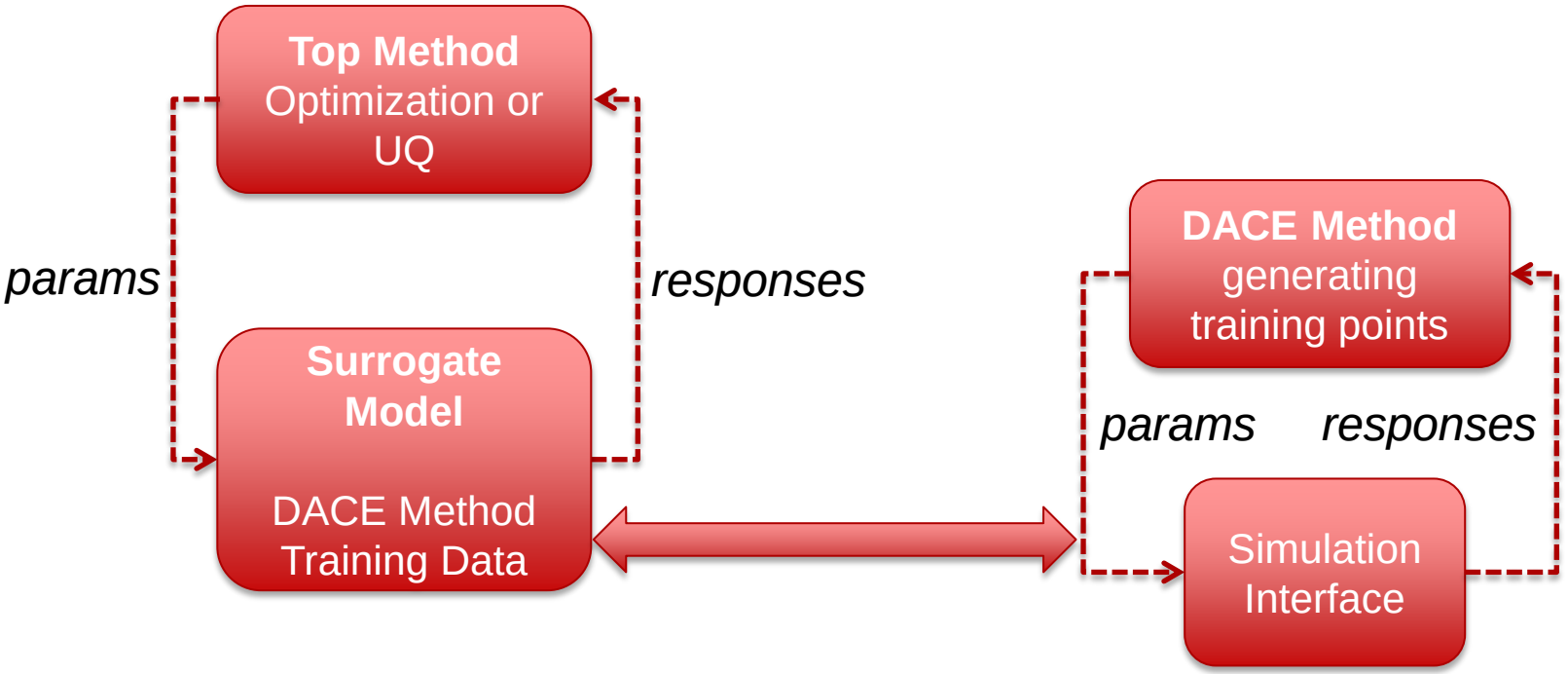
Module Learning Goals

- Define a surrogate model
- Identify situations where it may be appropriate to use a surrogate model
- Learn how to specify a surrogate model in Dakota
- Run a surrogate model in Dakota and examine outputs based on the surrogate model
- Identify some common diagnostics for surrogates
- Understand different ways surrogates are used in Dakota

Example: Sampling on a Surrogate

A single Dakota study can:

- Perform design of experiments on the truth model
- Construct a surrogate model
- Perform sampling-based UQ on the surrogate model



Surrogate Input File

```
environment,
        tabular_graphics_data
        method_pointer = 'METHOD_ON_SURR'

method,
       id_method = 'METHOD_ON_SURR'
       sampling
       sample_type lhs
       samples = 100
       seed = 3487
       model_pointer = 'SURR_MODEL'
       output verbose

model,
      id_model = 'SURR_MODEL'
      surrogate global
      dace_method_pointer = 'DACE'
      polynomial linear


method,
     id_method = 'DACE'
     model_pointer = 'DACE_M'
     sampling
     samples = 20
     seed = 3492
```

Perform 100 samples on the surrogate, for demonstration purposes

Linear surrogate ($\beta_0 + \beta_1 x_1 + \beta_2 x_2$)

DACE method can run the simulation code (truth model) to obtain the training points

Surrogate Input File

```
model,
        id_model = 'DACE_M'
        single
          interface_pointer = 'I1'
          variables_pointer = 'V1'
          responses_pointer = 'R1'

variables,
        id_variables = 'V1'
        uniform_uncertain = 2
          lower_bounds = 0. 0.
          upper_bounds = 1. 1.

interface,
        id_interface = 'I1'
         direct
         analysis_driver = 'text_book'

responses,
        id_responses = 'R1'
        num_response_functions = 3
        no_gradients
        no_hessians
```

The DACE model is the actual simulation model; typically points to variables, responses, and the interface to the actual simulation.

Note that a separate surrogate is constructed for each response specified. This may become expensive if you have many responses.

Surrogate Input File: Notes

```
model,
      id_model = 'SURR_MODEL'
      surrogate global
      dace_method_pointer = 'DACE'
      polynomial quadratic
      samples_file = 'training_data.dat' annotated
      # total_points INTEGER| minimum_points  |
        recommended_points
```

If you specify samples_file with the training points, specify samples = 0 in the DACE method

You can specify the number of training points desired.

```
method,
     id_method = 'DACE'
     model_pointer = 'DACE_M'
     sampling
     samples = 0
     seed = 3492
```

Example: Multi-fidelity Surrogate

```
model,
        id_model = 'SURROGATE'
        surrogate hierarchical
          low_fidelity_model  = 'LOFI'
          high_fidelity_model = 'HIFI'
          correction additive zeroth_order

model,
        id_model = 'LOFI'
        single
          interface_pointer = 'LOFI_FN'

interface,
        id_interface = 'LOFI_FN'
        direct
          analysis_driver = 'lf_rosenbrock'
          deactivate restart_file

model,
        id_model = 'HIFI'
        single
          interface_pointer = 'HIFI_FN'

interface,
        id_interface = 'HIFI_FN'
        direct
          analysis_driver = 'rosenbrock'
          deactivate restart_file
```

This shows an example of a low-fidelity and high-fidelity model.

Correction terms can be applied to surrogates for improved accuracy.

additive $\hat{f}_{hi_\alpha}(\mathbf{x}) = f_{lo}(\mathbf{x}) + \alpha(\mathbf{x})$

multiplicative $\hat{f}_{hi_\beta}(\mathbf{x}) = f_{lo}(\mathbf{x})\beta(\mathbf{x})$

convex combination

$$\hat{f}_{hi_\gamma}(\mathbf{x}) = \gamma \hat{f}_{hi_\alpha}(\mathbf{x}) + (1-\gamma)\hat{f}_{hi_\beta}(\mathbf{x})$$

Class Exercise

- Run dakota_cantilever_surrogate.in in the exercises/surrogate directory.
- How many samples are used to construct the surrogate?
- What method is used on the surrogate?
- How many times is the surrogate evaluated?
- Where are the surrogate responses?
- Run with “output verbose” in the outer method. Do you see the functional form of the surrogate? Can you identify some diagnostics?

Module Learning Goals

- Define a surrogate model
- Identify situations where it may be appropriate to use a surrogate model
- Learn how to specify a surrogate model in Dakota
- Run a surrogate model in Dakota and examine outputs based on the surrogate model
- Identify some common diagnostics for surrogates
- Understand different ways surrogates are used in Dakota

Surrogate Diagnostics

- Surrogate diagnostic metrics measure goodness-of-fit
- For each training point i, $Residual_i = RespFn_i - Surrogate_i$
- Dakota prints the first three by default; has others as well:
  - root_mean_squared: square root of the mean of the squared residuals
  - mean_abs: mean of the absolute value of the residuals
  - Rsquared: the proportion of the variability in the response that can be accounted for by the surrogate model (applicable for polynomials)
  - sum_squared: sum of the squared residuals
  - mean_squared: mean of the squared residuals
  - sum_abs: sum of the absolute value of the residuals
  - max_abs: max of the absolute value of the residuals
- Optimal Values of diagnostics
  - R-squared should be as close to 1 as possible. R-squared over 0.9 is considered good.
  - The other diagnostics should be as close to zero as possible (values will depend on the response function magnitude).

Predictive Quality of Surrogates

- Problem: diagnostics measure goodness of fit with respect to the training data, not a new set of data
- A cross-validation procedure can help assess the robustness of the surrogate in predicting data not in the training set
- For example, build the surrogate based on 80% of the data, predict on the remaining 20%, and repeat the process with a different 80% of the data (a different "fold").

- Two options in Dakota
  - `press` (prediction residual error sum of squares): leave-one-out
  - `cross_validation` (`folds` or `percent`): k-fold cross validation, divide the training set into k subsets and leave out one.

Example: Surrogate Diagnostics

- Input:

```
model,
        id_model = 'UQ_M'
        surrogate global
          dace_method_pointer = 'DACE'
          polynomial quadratic
          metrics 'mean_abs'
          cross_validation folds = 5
```

- Output:

```
Constructing global approximations with no anchor, 20 DACE samples, and 0
reused points.
--- User-requested surrogate metrics; function 1
mean_abs goodness of fit: 1.6451047024e-05

--- Cross validation (5 folds) of user-requested surrogate metrics;
function 1

mean_abs goodness of fit: 3.4088478529e-05
```

Module Learning Goals

- Define a surrogate model
- Identify situations where it may be appropriate to use a surrogate model
- Learn how to specify a surrogate model in Dakota
- Run a surrogate model in Dakota and examine outputs based on the surrogate model
- Identify some common diagnostics for surrogates
- Understand different ways surrogates are used in Dakota

Surrogate Use in Dakota

- **Simplest case:** method points to user-specified data-fit surrogate
  - Input file specifies a surrogate model type and training data or DACE method
  - Built once, then used with any method
- Methods with embedded surrogates that the user can specify
  - Bayesian methods can choose GP, PCE, SC (or no emulator)
  - Uses the emulator keyword (no need for a separate model block)
- General adaptive methods with user-specified surrogates
  - Trust region surrogate-based optimization with data fit or multi-fidelity
  - Iteratively builds/validates the surrogate in each trust region
- Adaptive methods, typically using Gaussian processes (no user control)
  - Efficient global optimization (EGO), efficient global reliability analysis, EGO interval estimation, EGO evidence theory, importance sampling (GPAIS)
- Stochastic expansions: these are not classified as a surrogate but serve as one for UQ methods, where the expansion settings determine the surrogate type and training points

Surrogate-Based Optimization

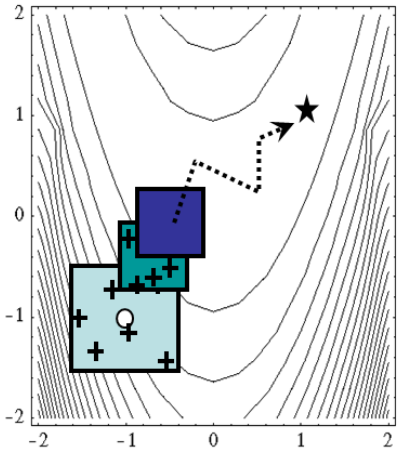
Sandia National Laboratories

Purpose:

- Reduce the number of expensive, high-fidelity simulations by using a succession of approximate (surrogate) models
- Approximations generally have a limited range of validity
- Trust regions adaptively manage this range based on efficacy during optimization
- With trust region globalization and local 1st-order consistency, SBO algorithms can be provably-convergent

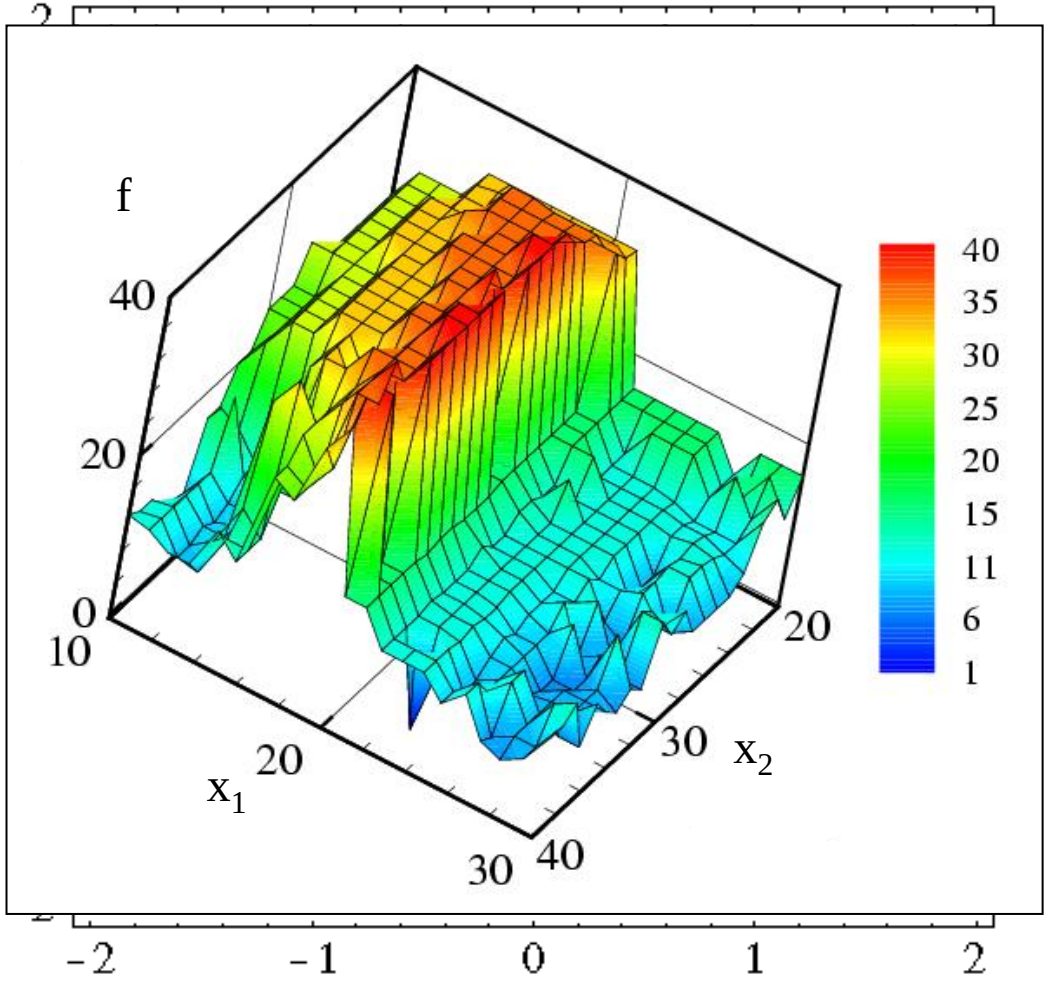
Surrogate models of interest:

- Data fits (local, multipoint, global)
- Multifidelity (special case: multigrid optimization)
- Reduced-order models

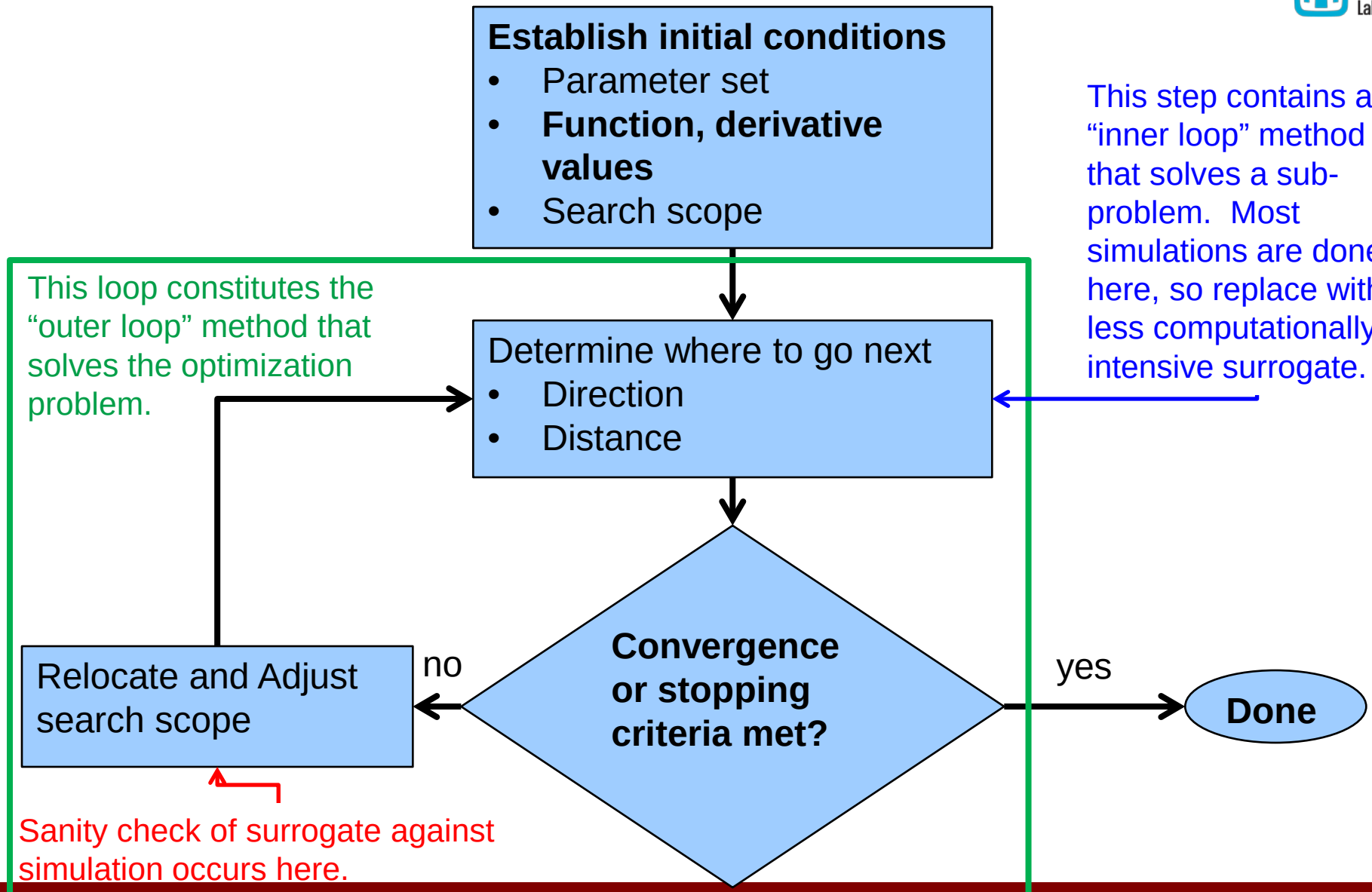


Trust Region Surrogate-Based Optimization (SBO)

Sequence of trust regions

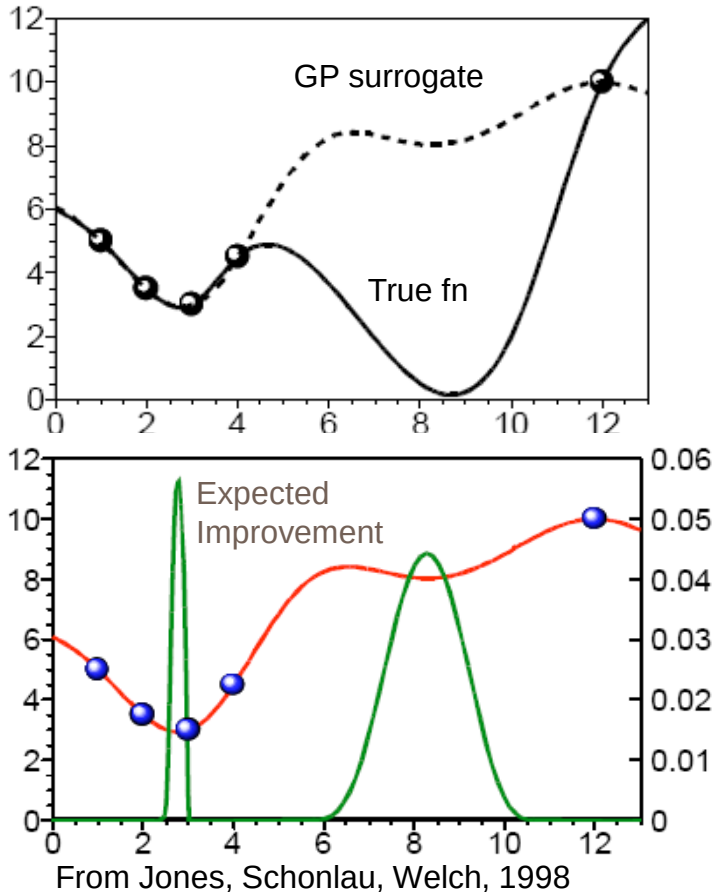


Structure of Surrogate-based Optimization



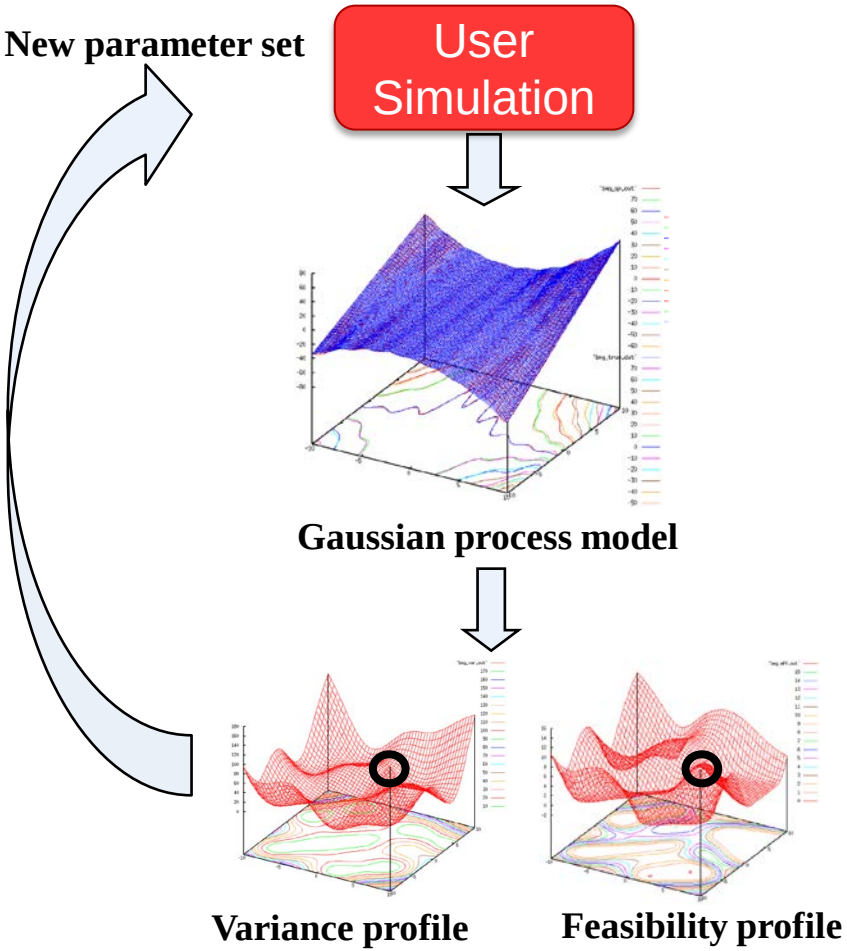
Efficient Global Optimization

- 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"
- Derivative-free, very efficient for low-dim.



From Jones, Schonlau, Welch, 1998

Efficient Global Reliability Analysis (EGRA)



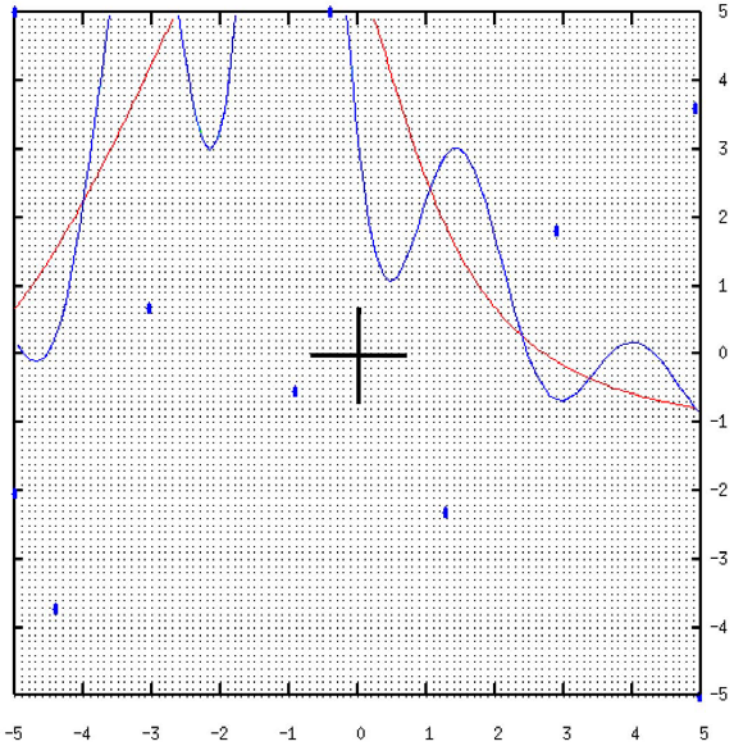
- Construct initial Gaussian process (GP) over small set of simulation samples
- Iteratively refine GP by balances exploration of unknown space with refinement around failure boundary
- Perform importance sampling on final GP to get probability of failure

Bichon, B.J., Eldred, M.S., Swiler, L.P., Mahadevan, S., and McFarland, J.M., "Efficient Global Reliability Analysis for Nonlinear Implicit Performance Functions," *AIAA Journal*, Vol. 46, No. 10, October 2008, pp. 2459-2468.

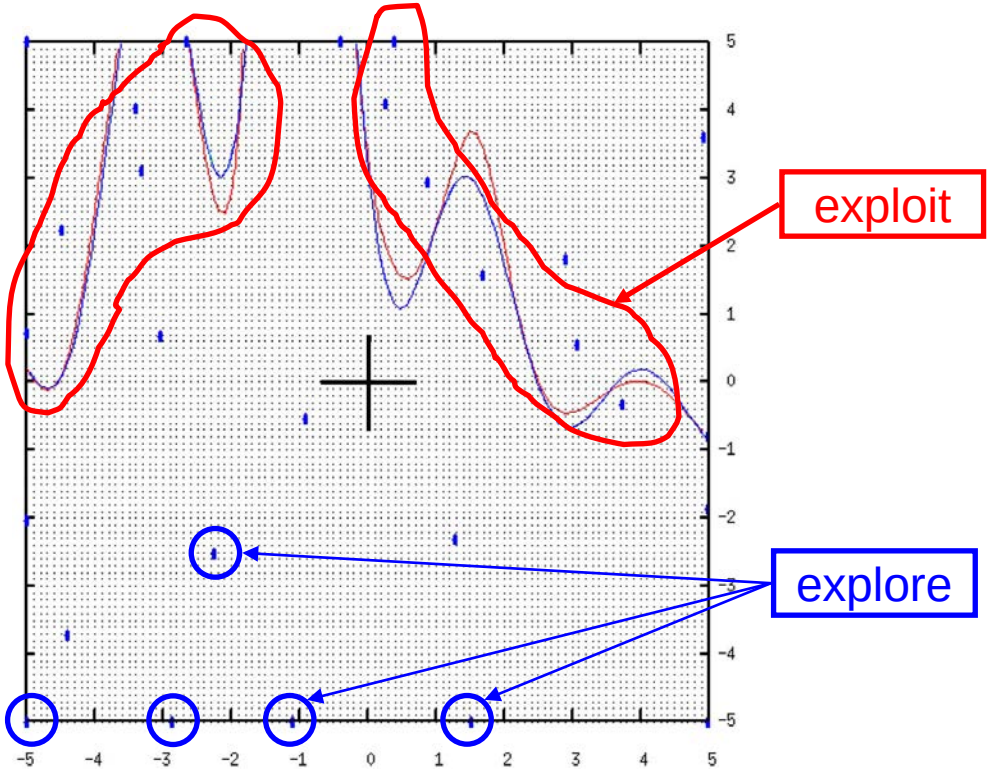
Efficient Global Reliability Analysis

Gaussian process model of reliability limit state with

10 samples



28 samples



Backup

DAKOTA example:

dakota_uq_textbook_lhs_approx.in, test 0

Sandia National Laboratories

```
method,
        id_method = 'UQ'
        model_pointer = 'UQ_M'
        sampling
          samples = 5000 seed = 5
          sample_type lhs
          response_levels = 2.e-5 3.e-5 4.e-5

model,
        id_model = 'UQ_M'
        surrogate global
          dace_method_pointer = 'DACE'
          polynomial quadratic

method,
        id_method = 'DACE'
        model_pointer = 'DACE_M'
        sampling
          samples = 2
          seed = 50
          sample_type lhs

model,
        id_model = 'DACE_M'
        single
          interface_pointer = 'I1'
```

DAKOTA example:

dakota_uq_textbook_lhs_approx.in, test 0

```
>>>>> Executing environment.

>>>>> Running random_sampling iterator.

NonD lhs Samples = 5000 Seed (user-specified) = 5

>>>>> Building global_polynomial approximations.

NonD lhs Samples = 6 Seed (user-specified) = 50
------------------------------
Begin       I1 Evaluation    1
------------------------------
Parameters for evaluation 1:
                     1.0898530162e+00 TF1ln
                     9.8321746393e-01 TF1ln

(Asynchronous job 1 added to I1 queue)

------------------------------
Begin       I1 Evaluation    2
------------------------------
Parameters for evaluation 2:
                     9.4844682596e-01 TF1ln
                     1.1492565898e+00 TF1ln

(Asynchronous job 2 added to I1 queue)
```

This is the outer loop sampling on the surrogate.

This is the inner loop sampling on the simulation to generate the training points for the surrogate.

DAKOTA example:

dakota_uq_textbook_lhs_approx.in, test 0

```
Active response data for I1 evaluation 6:
Active set vector = { 1 }
                     5.2588873552e-05 response_fn_1

Constructing global approximations with no anchor, 6 DACE samples, and 0 reused points.

Surfpack polynomial model
f(x) = sum_k{c_k * prod_k[x(i) ^ p(k,i)]}; where

inputs = 2
bases = 6

c (1 x bases) =
-4.8000038070265726e-02  3.7860149255203986e-02  9.9076783886974992e-03
-5.9599998476198845e-02  6.0935469676657854e-02 -1.0473904755832735e-03

p (bases x inputs) =
  0   0
  1   0
  2   0
  1   1
  0   1
  0   2
--- Default surrogate metrics; function 1
root_mean_squared goodness of fit: 2.0495082841e-17
mean_abs goodness of fit: 1.9446500040e-17
rsquared goodness of fit: 1.0000000000e+00
<<<<< global_polynomial approximation builds completed.
```

The last function evaluation completes.

Surrogate construction. If you specify “output debug” in the outer method, the surrogate details are printed.

Coefficients of regression model

Diagnostics to assess surrogate goodness-of-fit, based only on the 6 training points.

DAKOTA example:

dakota_uq_textbook_lhs_approx.in, test 0

```
-----------------------------------
Begin Approximate Fn Evaluation    1
-----------------------------------
Parameters for approximate fn evaluation 1:
                     1.0489504413e+00 TF1ln
                     1.0544994326e+00 TF1ln


Active response data for approximate fn evaluation 1:
Active set vector = { 1 }
                    -2.1808627603e-04 response_fn_1

<<<<< Function evaluation summary (APPROX_INTERFACE): 5000 total (5000 new, 0 duplicate)
  response_fn_1: 5000 val (5000 n, 0 d), 0 grad (0 n, 0 d), 0 Hess (0 n, 0 d)
<<<<< Function evaluation summary (I1): 6 total (6 new, 0 duplicate)
  response_fn_1: 6 val (6 n, 0 d), 0 grad (0 n, 0 d), 0 Hess (0 n, 0 d)

Statistics based on 5000 samples:

Moment-based statistics for each response function:
                            Mean           Std Dev          Skewness          Kurtosis
 response_fn_1  8.5672865092e-05  2.3321707752e-04  5.0540242996e-02  4.8865690512e+00

95% confidence intervals for each response function:
                     LowerCI_Mean      UpperCI_Mean    LowerCI_StdDev    UpperCI_StdDev
 response_fn_1  7.9206970775e-05  9.2138759410e-05  2.2873426833e-04  2.3788038944e-04

ETC.
```

This is the first of the 5000 samples on the surrogate.

Statistics based on sampling the surrogate

* Note the dakota_tabular.dat file will have results based on the surrogate *

Gaussian Processes

- 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?

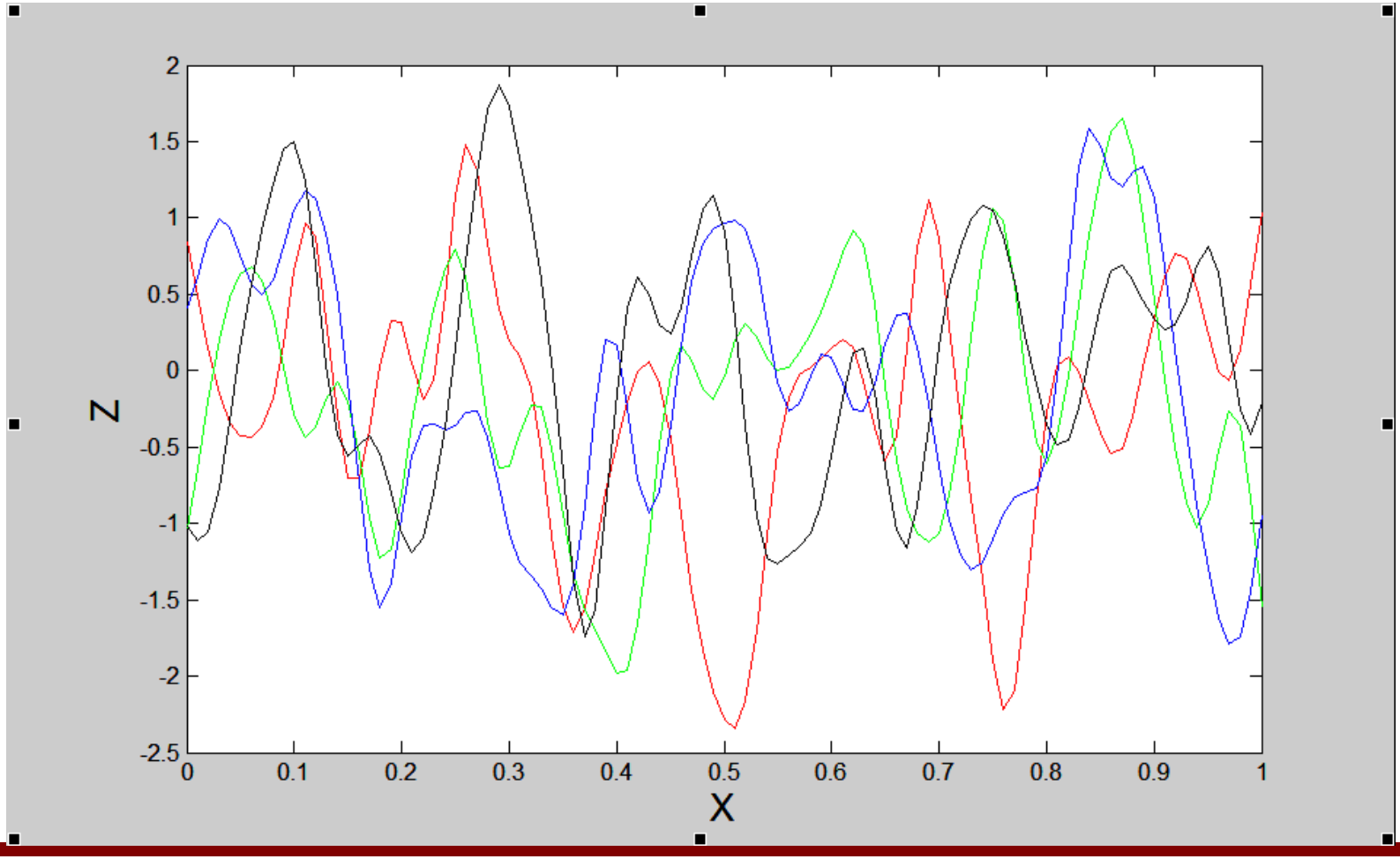
- Start with a set of runs of a computer code: at each sample $\mathbf{x}_i$ we have output $y_i(\mathbf{x}_i)$.
- The output at a new input value, $\mathbf{x}_{new}$, is uncertain.
- This is what a GP will predict.
- Related to regression.
- Related to random functions. From our set of samples, we have 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?

- 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 $\mathbf{C} = \mathbf{L}\mathbf{L}^t$.
  - Define $\mathbf{Z} = \mathbf{L}\mathbf{S}$.
  - Plot the points $\{\mathbf{x}_i, Z_i = Z(\mathbf{x}_i)\}$
  - Connect the dots

Example covariance function in 1-D

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



Gaussian Process

- 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

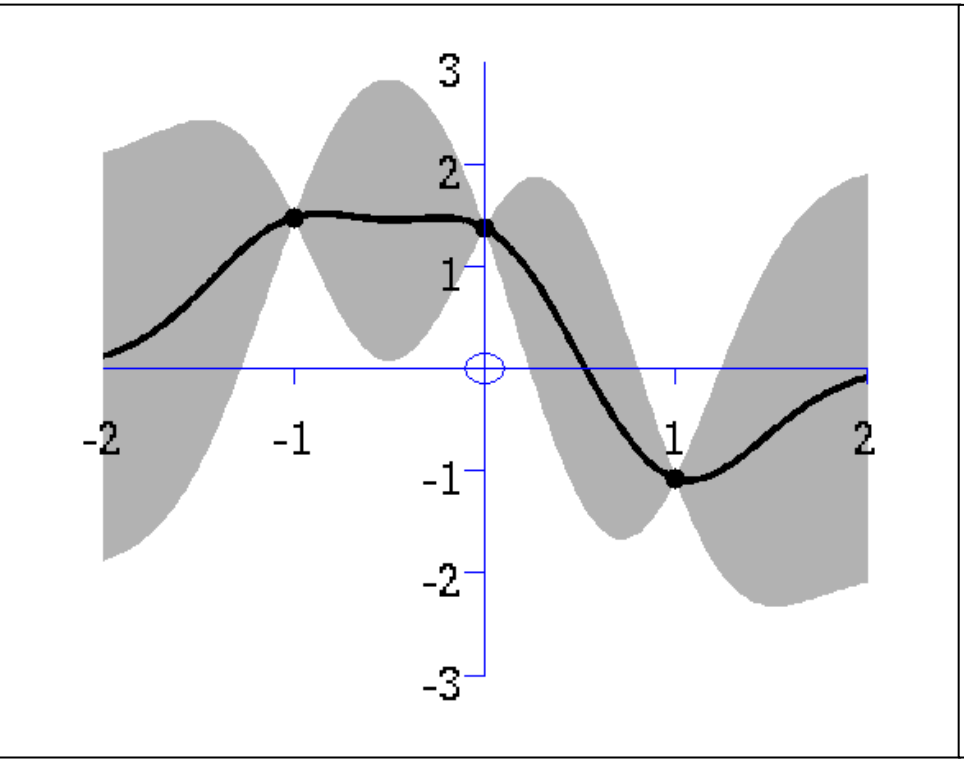
- NOW: what is the prediction for a new point?

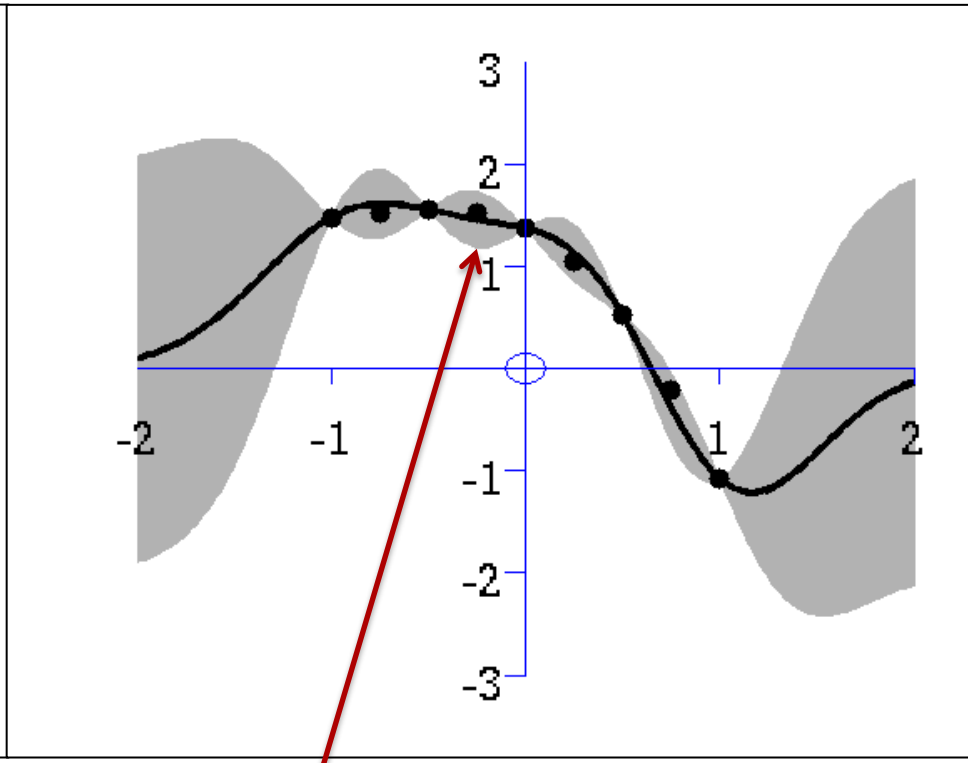
$$\mathrm{E}[y(\boldsymbol{x}^*)|\boldsymbol{Y}] = f(\boldsymbol{x}^*)^T\boldsymbol{\beta} + r(\boldsymbol{x}^*)^T\mathbf{R}^{-1}[\mathbf{Y} - \mathbf{F}\boldsymbol{\beta}]$$

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

- 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?

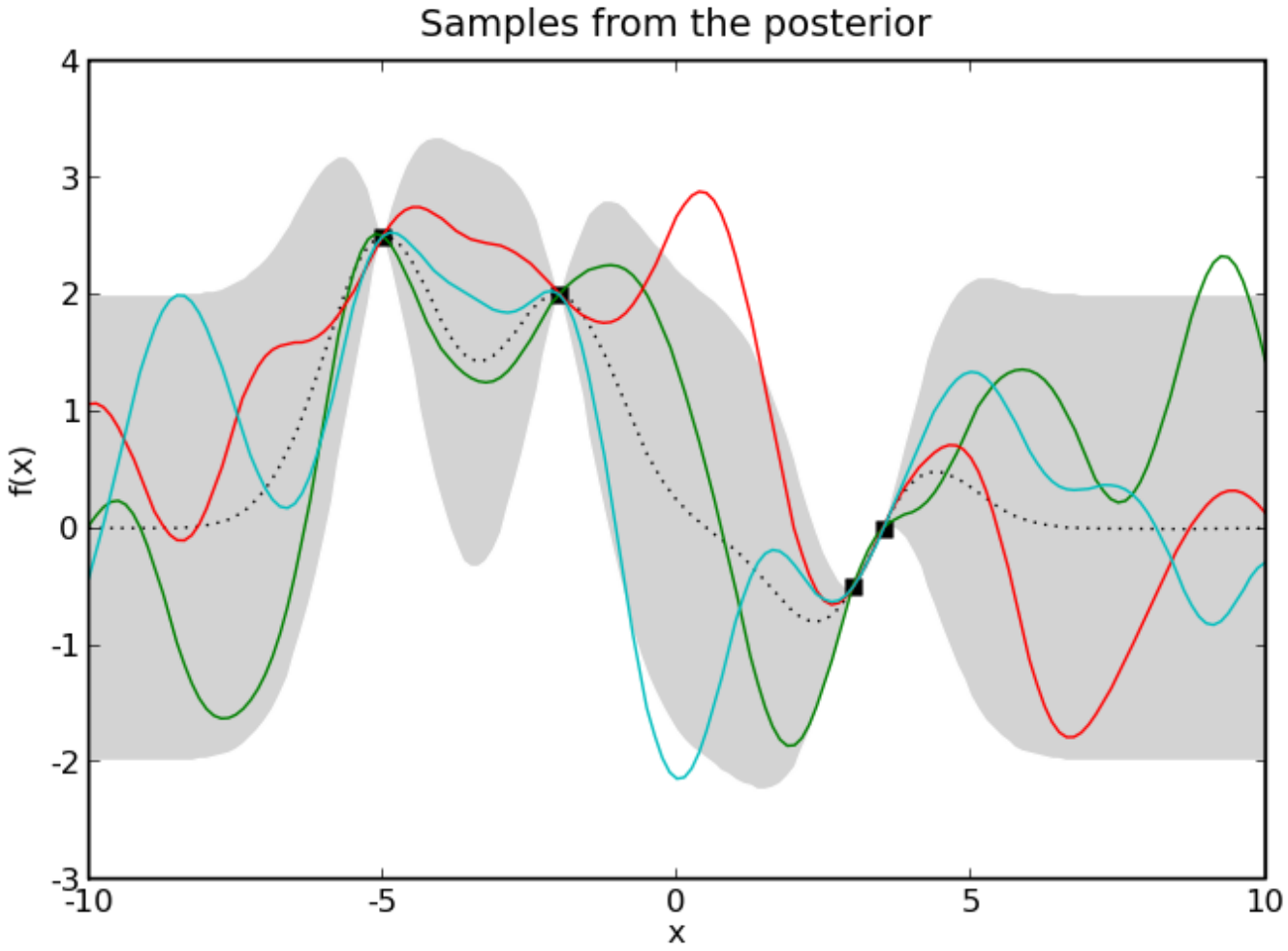




staffwww.dcs.shef.ac.uk

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 (or squared 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.

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

Correlation Function

Matern

$$R(\boldsymbol{x}, \boldsymbol{x}') = \prod_{j=1}^{d} \frac{2^{1-\nu}}{\Gamma(\nu)} (-\theta_j \left| x_j - x_j' \right|^{\nu}) K_{\nu} (-\theta_j \left| x_j - x_j' \right|^{\nu})$$

- Is equal to the exponential covariance function when $\nu = ½$.
- Is equal to the squared exponential when $\nu \rightarrow \infty$
- Typically, $\nu = ½, 3/2$, or $5/2$, going a process that looks rough to a process that is fairly smooth.
-
- Other covariances are possible: Cauchy, polynomial functions, etc.

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.