Gradient Enhanced Bayesian MARS for Regression and Uncertainty Quantification

Hayes F. Stripling Ryan G. McClarren

Department of Nuclear Engineering, Texas A&M University

2011 ANS Winter Meeting
1 November 2011

This work describes an extension of the Bayesian MARS emulator to include gradient information, when available.

Most uncertainty quantification tasks boil down to: "Estimate the sensitivity and/or variability in some quantity $y = f(x)$ resulting from uncertainty or variability in its dependencies."

- Dimensionality of $\vec{x}$ and cost of $f(\cdot)$ may limit sampling density.
- **Emulators** (or response surfaces) "functionalize" the mapping $y = f(x)$ using a set of available samples, $y_i = f(x_i),\ i = 1 \ldots N$.
- An effective emulator
  1. is cheap to sample,
  2. provides accurate estimates of y at untried inputs, and
  3. gives an estimate of its own regression error.

This work describes an extension of the Bayesian MARS emulator to include gradient information, when available.

Most uncertainty quantification tasks boil down to: "Estimate the sensitivity and/or variability in some quantity $y = f(x)$ resulting from uncertainty or variability in its dependencies."

- Dimensionality of $\vec{x}$ and cost of $f(\cdot)$ may limit sampling density.
- **Emulators** (or response surfaces) "functionalize" the mapping $y = f(x)$ using a set of available samples, $y_i = f(x_i),\ i = 1 \ldots N$.
- An effective emulator
  1. is cheap to sample,
  2. provides accurate estimates of y at untried inputs, and
  3. gives an estimate of its own regression error.

How can we use $\nabla_x f$?

Adjoint and/or automatic differentiation methods can possibly provide gradient information about $f(x)$. How can we use this information to improve the effectiveness of our emulators?

Inclusion of gradient information requires differentiation of the emulator's functionalization.

Some examples:

1. Polynomial chaos: write the unknown as a multivariate polynomial expansion in x,

$$y \approx P(x) = \sum_i a_i \psi_i(x)$$

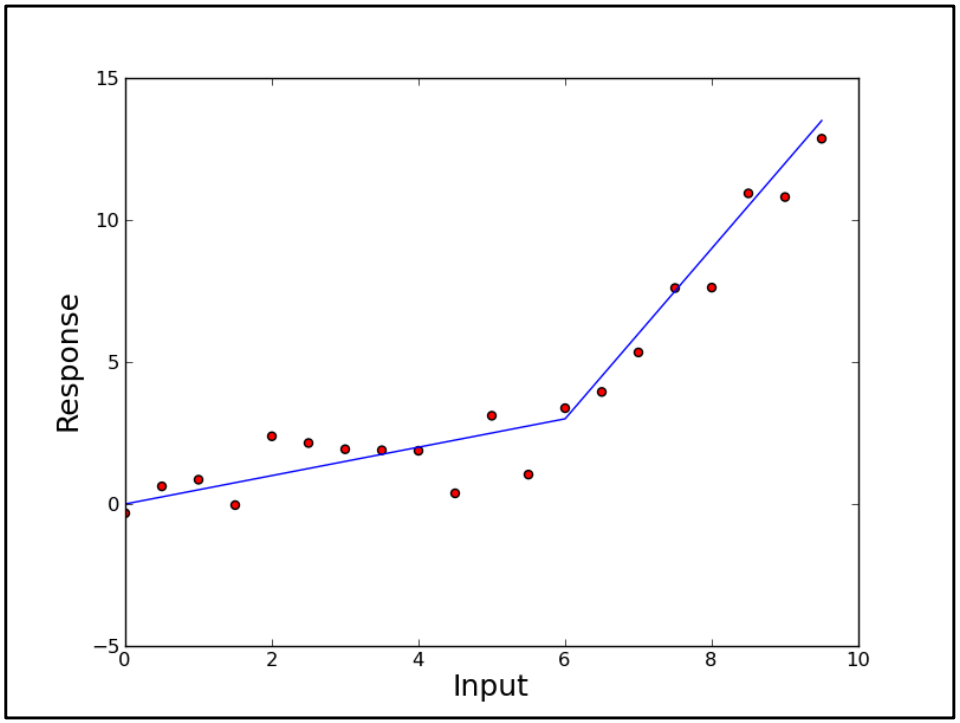
   and solve for the a_i's. Solution techniques vary, but gradient information is fairly straightforward to include.

2. Gaussian process regression: Model $f(x)$ as a multivariate random field specified by a mean and covariance function. Inclusion of gradient information requires differentiation of the mean and covariance function (Lockwood, Anitescu – Summer 2011 ANS meeting).

We apply the extension to the Multivariate Adaptive Regression Splines (MARS) emulator.

The MARS basis function is a summation of polynomial "spline" functions.

These splines are defined to be zero on part of the domain and polynomial in x on the remainder.



The MARS basis function is

$$B(x) = \beta_0 + \sum_{k=1}^{K} \beta_k \prod_{l=0}^{\mathbf{I}} (x_l - t_{k,l})_+^{o_k},$$

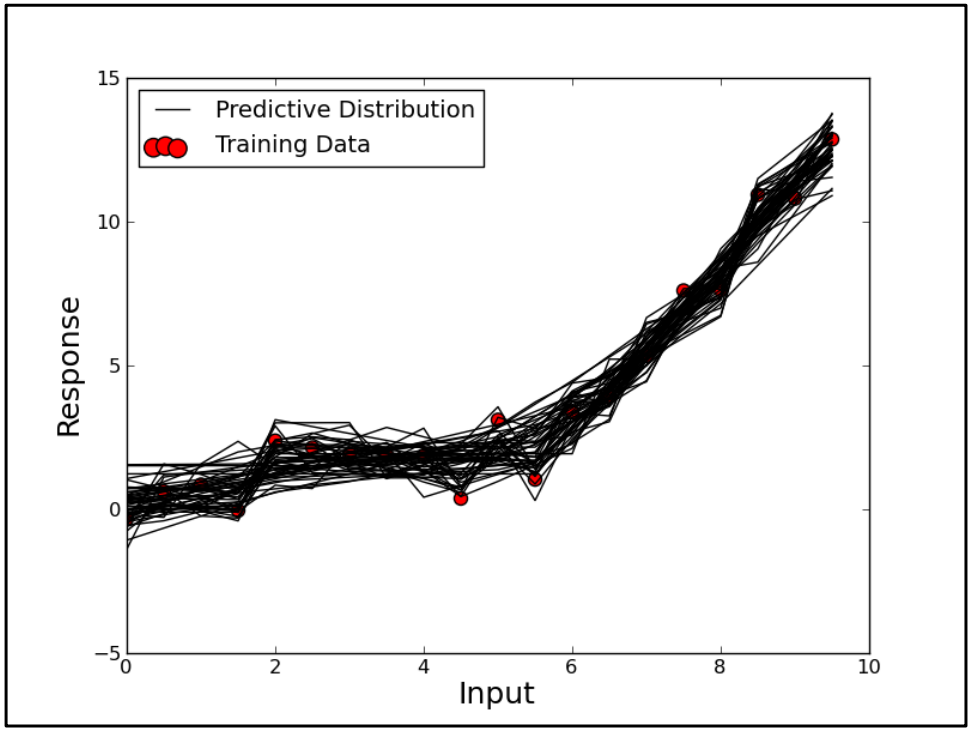
where the regression coefficients β_k are chosen to minimize the error in the approximation

$$B(x_i) \approx f(x_i)$$

The Bayesian extension of MARS is important for treating uncertainty in the response data.

Evolve the basis function using Markov chain Monte Carlo

1. Propose change to basis function
2. Recompute β's
3. Compute likelihood of new model (goodness of fit)
4. Accept/Reject proposal
5. Repeat



If we have noisy data, we'd prefer that our predictions give some estimate of the distribution of that noise. BMARS provides a posterior **distribution** of predictor functions.

The gradient of a BMARS basis function is another BMARS function.

The gradient in direction x_n is:

$$\nabla_{x_n} B(x) = \sum_{k=1}^{K} o_{k,n} \beta_k \prod_{l=0}^{\mathbf{I}} (x_l - t_{k,l})_+^{o^*_{k,l}}$$

where

$$o^*_{k,l} = \begin{cases} o_{k,l} - 1 & l = n \\ o_{k,l} & l \neq n \end{cases}.$$

Thus, we can use the same machinery to evaluate B and ∇B. Our regression task is now to minimize the error in the fit:

$$B(x_i) \approx f(x_i),$$
$$\nabla_x B(x_i) \approx \nabla_x f(x_i).$$

The regression coefficients are solved for using a Bayesian least-squares approach.

The least squares problem is written as an over-constrained linear system:

$$\left(\mathbf{A}^T\mathbf{A} + \tau^2 I_K\right)\beta = \mathbf{A}^T b,$$
$$\mathbf{A} \in \mathbb{R}^{P\times K}, \quad b \in \mathbb{R}^P, \quad P > K, \quad \tau \in \mathbb{R}.$$

where

- the first I rows of matrix $\mathbf{A}$ contain the K unscaled splines $\hat{B}$ evaluated at each x_i;
- the next N blocks of I rows contain the unscaled gradients, $\dfrac{\partial \hat{B}}{\partial x_n}$, evaluated at each x_i;
- b contains the function and gradient response data;
- β are the regression coefficients; and
- τ is a Bayesian precision parameter.

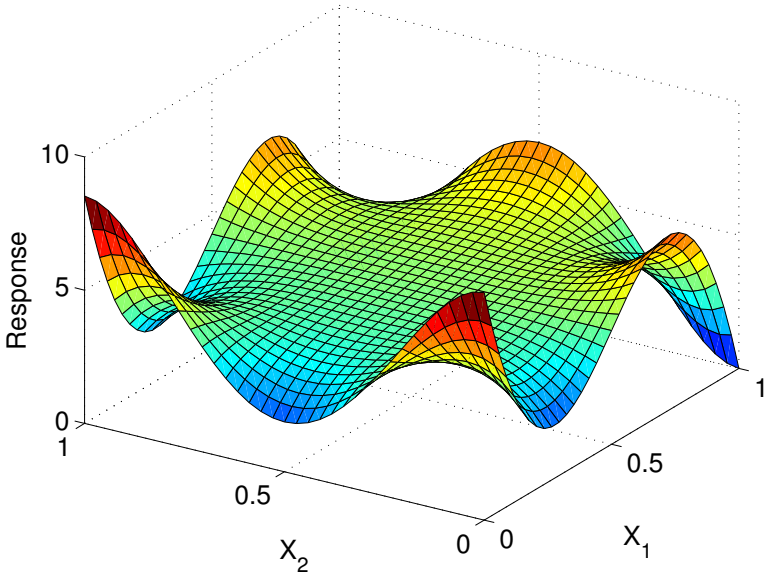
The regression coefficients are solved for using a Bayesian least-squares approach, ctd...

In explicit form (for $n = 1 \ldots N$):

$$\mathbf{A} = \begin{bmatrix} \hat{B}_1(\vec{x}_1) & \hat{B}_2(\vec{x}_1) & \ldots & \hat{B}_K(\vec{x}_1) \\ \hat{B}_1(\vec{x}_2) & \hat{B}_2(\vec{x}_2) & \ldots & \hat{B}_K(\vec{x}_2) \\ \vdots & \vdots & \ddots & \vdots \\ \hat{B}_1(\vec{x}_I) & \hat{B}_2(\vec{x}_I) & \ldots & \hat{B}_K(\vec{x}_I) \\ \frac{d\hat{B}_1}{dx_n}(\vec{x}_1) & \frac{d\hat{B}_2}{dx_n}(\vec{x}_1) & \ldots & \frac{d\hat{B}_K}{dx_n}(\vec{x}_1) \\ \vdots & \vdots & \ddots & \vdots \\ \frac{d\hat{B}_1}{dx_n}(\vec{x}_I) & \frac{d\hat{B}_2}{dx_n}(\vec{x}_I) & \ldots & \frac{d\hat{B}_K}{dx_n}(\vec{x}_I) \end{bmatrix}, \quad \vec{b} = \begin{bmatrix} f(\vec{x}_1) \\ f(\vec{x}_2) \\ \vdots \\ f(\vec{x}_I) \\ \nabla_{x_n} f(\vec{x}_1) \\ \vdots \\ \nabla_{x_n} f(\vec{x}_I) \end{bmatrix}$$

We verify the algorithm on a set of test functions proposed by the original BMARS authors (Denison, et. al., 1998).

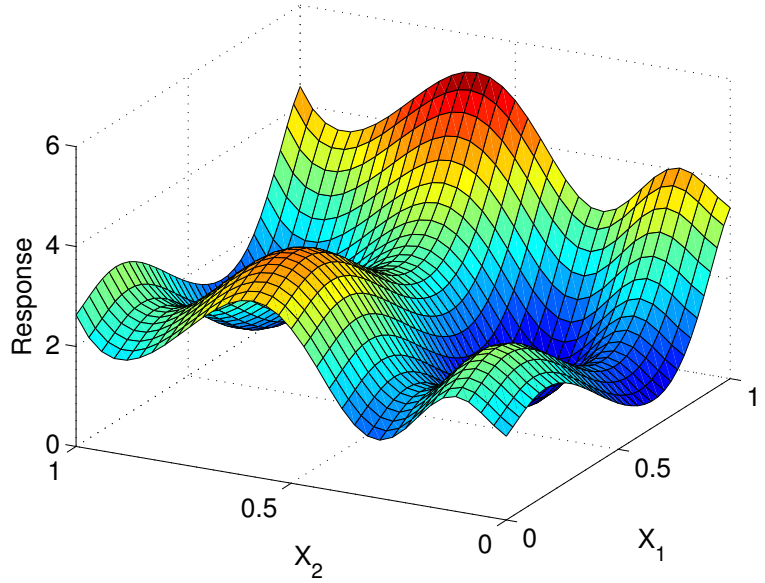
"Harmonic" Function



$$f(x_1,x_2) = 42.659\left[0.1+\hat{x}_1\left(0.05+\hat{x}_1^4-10\hat{x}_1^2\hat{x}_2^2+5\hat{x}_2^4\right)\right],$$

$$\hat{x}_n = x_n - .5$$

"Additive" Function



$$f(x_1,x_2) = 1.3356\Big\{1.5(1-x_1) + \exp(2x_1-1)\sin\left(3\pi(x_1-.6)^2\right) + \exp(3(x_2-.5))\sin\left(4\pi(x_2-.9)^2\right)\Big\}$$

Using gradient information decreases regression error.

Our regression metric is the Fraction of Variance Unexplained (FVU):

$$\text{FVU} = \frac{\sum_i \left(B(x_i) - f(x_i)\right)^2}{\sum_i \left(f(x_i) - \bar{f}\right)^2},$$

Using gradient information decreases regression error.

Our regression metric is the Fraction of Variance Unexplained (FVU):

$$\text{FVU} = \frac{\sum_i \left(B(x_i) - f(x_i)\right)^2}{\sum_i \left(f(x_i) - \bar{f}\right)^2},$$

Case (# Samples)	Training Data		Testing Data	
Harmonic	BMARS	gBMARS	BMARS	gBMARS
5^2	8.761e-01	3.254e-02	9.821e-01	1.176e-01
10^2	2.464e-03	1.179e-03	8.407e-02	3.221e-03
15^2	1.926e-03	3.683e-04	3.594e-03	5.553e-04
Additive	BMARS	gBMARS	BMARS	gBMARS
5^2	1.020e-03	1.112e-03	3.432e-01	4.009e-02
10^2	6.644e-04	8.399e-04	1.297e-02	3.696e-03
15^2	7.373e-04	4.269e-04	3.577e-03	8.133e-04

- Training data: 5^2, 10^2, or 15^2 uniform samples on unit square
- Testing data: 10 000 uniform samples on unit square
- Reporting mean FVU of 5 repititions

We apply gBMARS to do UQ on a mock traveling wave reactor problem.

- A model problem for the CESAR Exascale center (ANL, TAMU, and others)
- Coupled 3-nuclide Bateman Eqs. and 1D diffusion (nonlinear):

$$\frac{\partial}{\partial t}N(r,t) = \mathbf{M}\big(\phi(r,t),p\big)N(r,t)$$
$$\mathbf{M}(N(r,t),p)\phi(r,t) - \Sigma_a^{\text{ext}}(p)\phi(r,t) = 0$$
$$\phi^T\Sigma_f - P_0 = 0$$

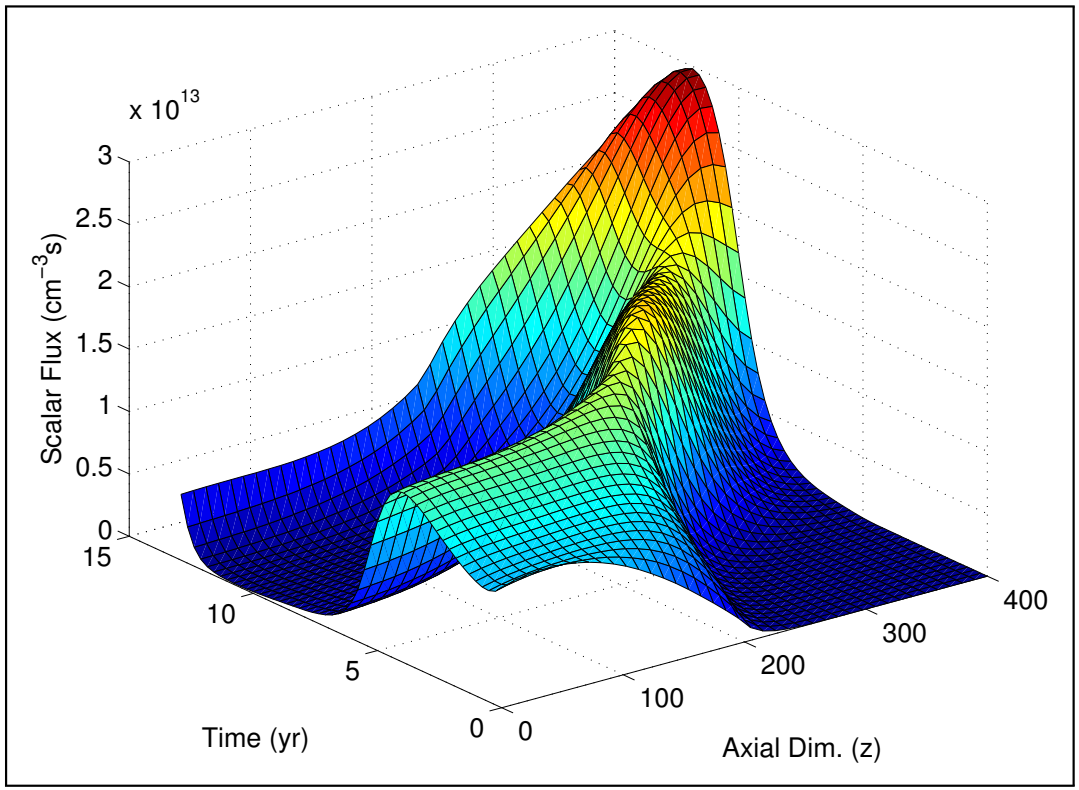
We apply gBMARS to do UQ on a mock traveling wave reactor problem.

- A model problem for the CESAR Exascale center (ANL, TAMU, and others)
- Coupled 3-nuclide Bateman Eqs. and 1D diffusion (nonlinear):

$$\frac{\partial}{\partial t}N(r,t) = \mathbf{M}\big(\phi(r,t),p\big)N(r,t)$$
$$\mathbf{M}(N(r,t),p)\phi(r,t) - \Sigma_a^{\text{ext}}(p)\phi(r,t) = 0$$
$$\phi^T\Sigma_f - P_0 = 0$$

- Σ_a^{ext}: Engineering degree of freedom
- p: a vector of uncertain parameters (eg: cross-sections)
- Initial conditions: power concentrated at one end of reactor
- "Downstream" fertile material breeds and reaction moves through the slab

We use a mock 1D Traveling Wave Reactor model to use gBMARS for Uncertainty Quantification.



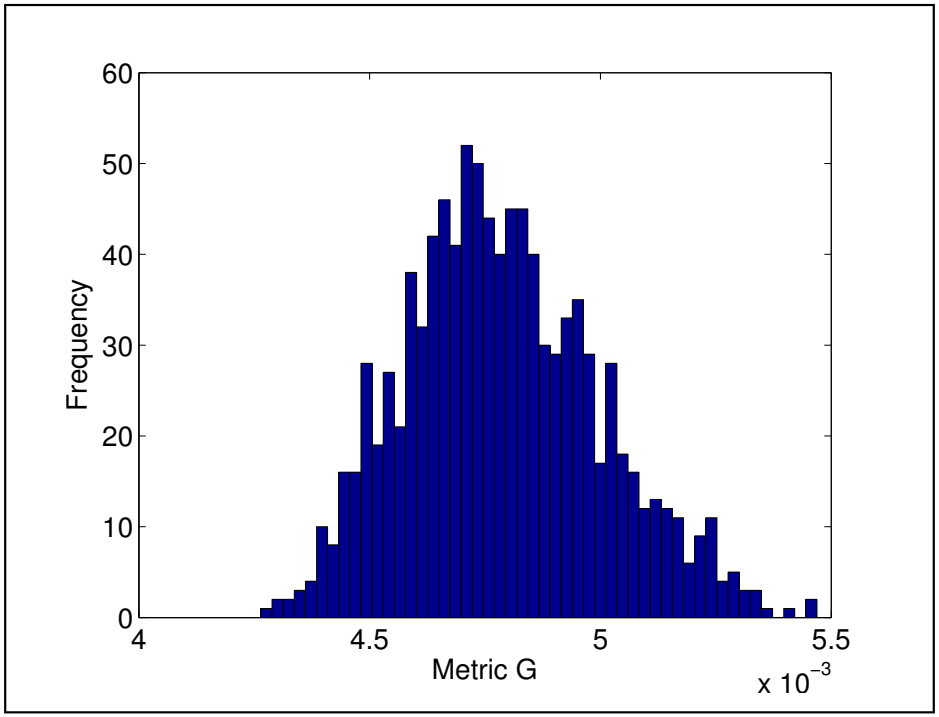
- We solve both the forward and adjoint problem.
- Define a neutron economy metric:

$$G = \frac{\text{Neutrons Used for Breeding}}{\text{Neutrons Lost to Leakage}}$$

- Adjoint problem gives sensitivity of G w.r.t parameters in p

Our UQ task is to estimate the mean, variance, and individual realizations of our neutron economy metric.

- Our 10 inputs are allowed to vary independently within 10% of their nominal value.
- We have 40 full forward/adjoint solution pairs (40 function/gradient evaluations) generated by LHS sampling of the inputs.
- We'll build and sample a BMARS model using 5, 10, 15, 20, 30, and 40 of these samples both with and without the gradient information.
- We also have 1000 forward LHS samples – we will verify the gBMARS predictions against these runs.



Distribution of our 1,000 "testing" runs.

Relative Error in Predicting the distribution mean

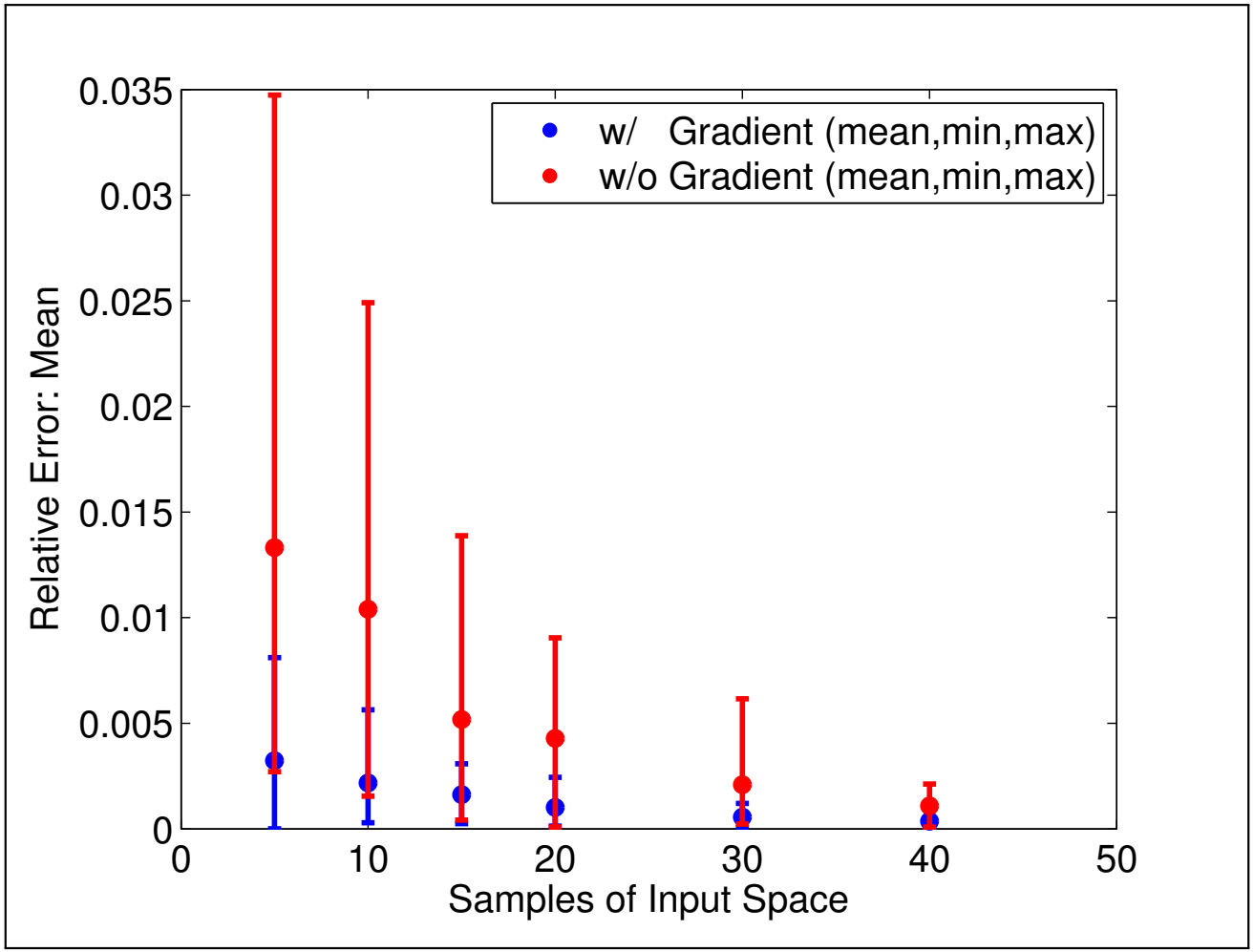


Figure: NOTE: 10 repetitions of each sample size

Relative Error in Predicting the distribution variance

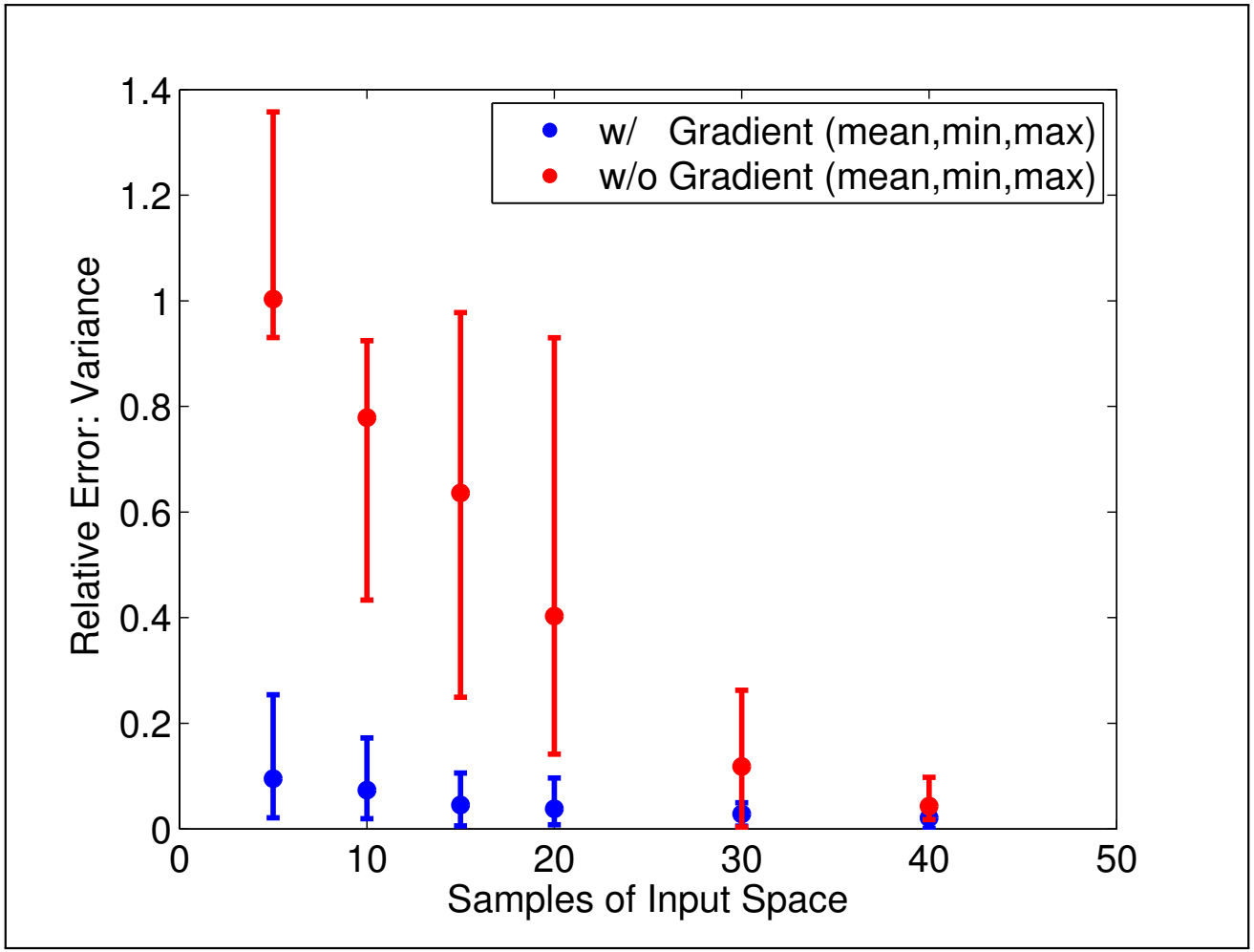
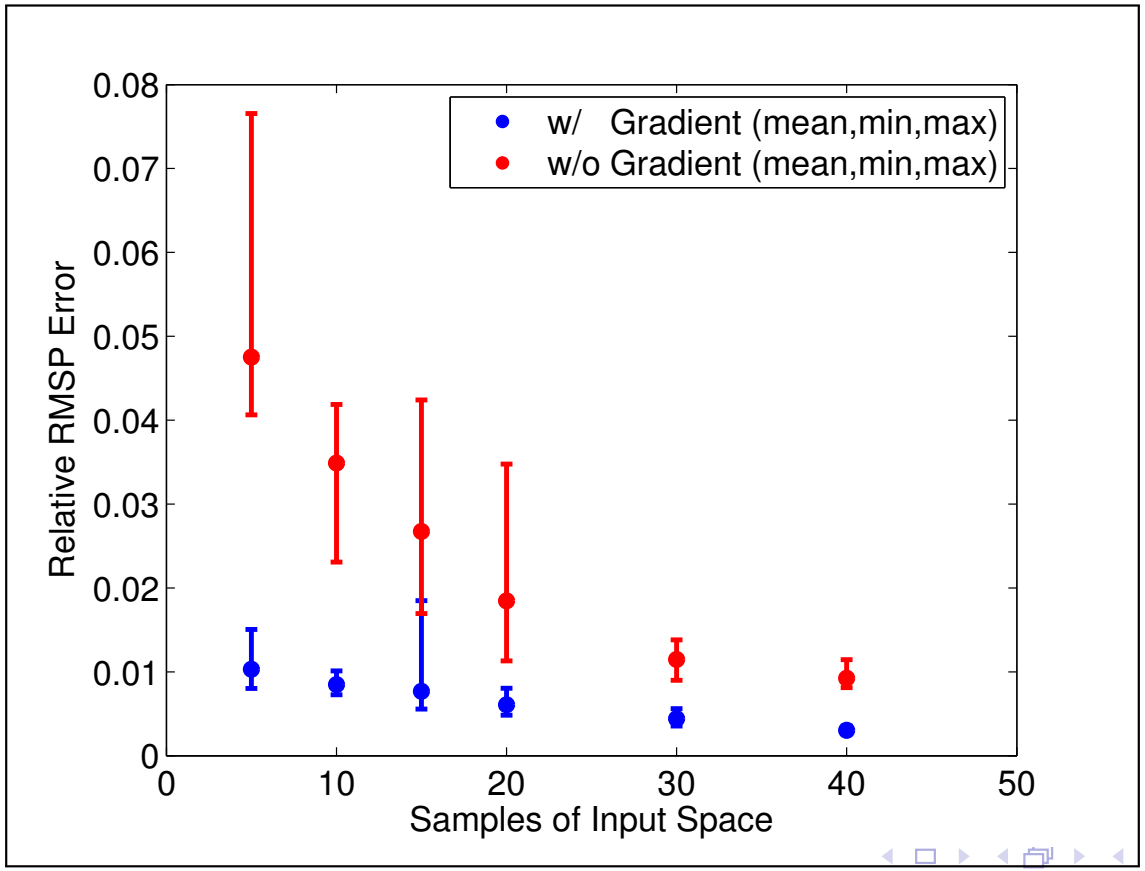


Figure: NOTE: 10 repetitions of each sample size

Root-Mean-Squared Predictive Error

$$\text{RMSPE} = \sqrt{\frac{1}{N}\sum_{i=1}^{N}\left[B(x_i) - f(x_i)\right]^2}$$



We demonstrate an extension of the BMARS algorithm to include gradient information for improved regression.

1. Gradient information is fairly straightforward to include in the BMARS emulator.
2. For both a suite of bivariate testing problems and a higher-dimensional reactor problem, the use of gradient information improved the regression of the underlying function and reduced the predictive variance.
3. Gradient information provided the greatest gains in:
    1. predicting both individual realizations and the variance of the metric distribution; and
    2. the cases of very sparse sampling of the input space.
4. Gradient enhanced emulators provide reasoning for a modeler to pay the extra cost of an adjoint or AD solve.

Questions?

Some gradient information produced using the INTLAB forward automatic differentiation package for MATLAB.