INPUT SUBSPACE DETECTION FOR DIMENSION REDUCTION IN HIGH DIMENSIONAL APPROXIMATION

PAUL G. CONSTANTINE∗ AND QIQI WANG†

Abstract. Many multivariate functions encountered in practice vary primarily along a few directions in the space of input parameters. When these directions correspond with coordinate directions, one may apply global sensitivity measures to determine the parameters with the greatest contribution to the function’s variability. However, these methods perform poorly when the directions of variability are not aligned with the natural coordinates of the input space. We present a method for detecting the directions of variability of a function using evaluations of its derivative with respect to the input parameters. We demonstrate how to exploit these directions to construct a surrogate function that depends on fewer variables than the original function, thus reducing the dimension of the original problem. We apply this procedure to an exercise in uncertainty quantification using an elliptic PDE with a model for the coefficients that depends on 250 independent parameters. The dimension reduction procedure identifies a 5-dimensional subspace suitable for constructing surrogates.

Key words. dimension reduction, high dimensional approximation, interpolation, surrogate models

1. Introduction & Motivation. In modern science and engineering practice, computational simulation is routinely employed to help test hypotheses and explore new designs. As the speed and capability of computers increase, so does the complexity of simulations through greater resolution and higher fidelity physical models. Expensive simulations requiring extensive time on massive supercomputers are now commonplace. Due to the cost of these high-fidelity simulations, one often wishes to approximate the output at many points in the space of inputs using a surrogate function or a meta-model. The parameters of the surrogate are tuned with a budget-constrained number of costly high-fidelity runs, and the tuned surrogates are used to study sensitivities or uncertainties in the simulation output with respect to variation in the input parameters.

However, many surrogate models suffer from the so-called curse of dimensionality. Loosely speaking, the work required to construct and evaluate an accurate surrogate increases exponentially as the dimension of the parameter space increases. For example, this curse limits the applicability of polynomial-based surrogates to problems with a handful of input parameters. Even methods whose application is independent of the dimension of the parameter space – such as radial basis functions or Gaussian process models – often perform poorly if the function is not sufficiently smooth and the training data are too sparse.

Fortunately, in many problems of interest with high dimensional input spaces, the output often depends on only a few important parameters. Specifically, the variability in the output can be attributed to a subset of the inputs. Surrogates can be adjusted to take advantage of this anisotropic parameter dependence. A common approach – known as global sensitivity analysis [12] – involves a strategy for ranking the input variables and biasing the choice of design points to capture the function’s behavior as the important parameters are varied. In some cases, the ranking procedure can use a priori knowledge from the mathematical model. In other cases, it requires exploration of the output through sampling. In either case, methods based

∗Stanford University, Stanford, California 94305 (paul.constantine@stanford.edu).
†Department of Aeronautics and Astronautics, Massachusetts Institute of Technology, Cambridge, Massachusetts 02139 (qiqi@mit.edu).
on variance-based decompositions [10] or high-dimensional model representations [8] choose a few important parameters from the full set of inputs. For problems encountered in practice, this often results in a *dimension reduction* of the input space; a surrogate can be constructed on a function of fewer variables with significantly less work.

In this paper, we present a generalization of subset selection methods. Namely, we seek a low-dimensional linear subspace of the input parameter space that captures the majority of the output’s variability. The subspace induces a reduced set of coordinates, and surrogate functions can be trained on the reduced coordinates to approximate the output in the full space.

More precisely, for a function of interest \( f = f(s) \) with \( s \in \mathbb{R}^d \), we seek a function \( g = g(s_a) \) with \( s_a \in \mathbb{R}^a \) such that \( f \approx g \) with \( a < d \). The approximate function takes the form \( g(s_a) = f(As_a) \), where \( A \) is a \( d \times a \) matrix representing a linear map from \( \mathbb{R}^a \) to \( \mathbb{R}^d \). Note that the introduction of \( g \) is primarily for notation; each evaluation of \( g \) is ultimately an evaluation of \( f \) at specially chosen input values. However, the dependence of \( g \) on fewer variables makes it more amenable to surrogate approximation.

A similar idea is proposed in [9] in the context of model reduction for inverse problems, but the method for computing the basis vectors that define the subspace employs the residual of a system of equations representing a physical model. Our method applies to more general multivariate functions, and it is particularly efficient if one can easily compute gradients of outputs with respect to inputs. We discuss strategies for the case when only function evaluations are available, including an intriguing idea of using new matrix completion techniques on a partially sampled matrix of finite difference approximations of the gradient.

### 2. Input subspace detection and dimension reduction.

We assume that a given multivariate function \( f(s) \) with \( s \in \mathbb{R}^d \) varies primarily along a few directions in the input space. However, these directions may not be aligned with the natural coordinate system. The goal in this section is to construct a function \( g \) that approximates \( f \) but takes only as many inputs as directions of variability. Our strategy is to first determine the directions along which \( f \) varies most prominently; we rotate our coordinate system according to these directions. We then define \( g \) to depend on the subset of these rotated coordinates that contain the majority of the variability in \( f \).

#### 2.1. Directions of variability.

Let \( \Omega \) be a hyperrectangle defined by the vectors \( s_l \) and \( s_u \),

\[
\Omega = \{ s : s \in \mathbb{R}^d, s_l \leq s \leq s_u \} \tag{2.1}
\]

We assume without loss of generality that \( 0 \in \mathbb{R}^d \) is the center of mass of \( \Omega \). Define a scalar function \( f : \Omega \to \mathbb{R} \) that takes \( d \) inputs. Denote an element of \( \Omega \) by a \( d \)-vector \( s = (s_1, \ldots, s_d)^T \in \Omega \). For the analysis, we assume that \( f \) is analytic in a region containing \( \Omega \). Denote the \( d \)-vector \( \mathbf{j} = \mathbf{j}(s) \) as

\[
\mathbf{j}^T = \nabla f = \begin{bmatrix} \frac{\partial f}{\partial s_1} & \cdots & \frac{\partial f}{\partial s_d} \end{bmatrix}, \tag{2.2}
\]

which is the Jacobian of \( f \). Define the \( d \times d \) matrix \( C \)

\[
C = \int_\Omega \mathbf{j} \mathbf{j}^T \, ds, \tag{2.3}
\]
where we employ a shorthand $ds$ to denote a measure on $\Omega$. Note that $C$ is symmetric and positive semidefinite, which implies it has an eigenvalue decomposition

$$C = V \Lambda V^T, \quad \Lambda = \text{diag} (\lambda_1, \ldots, \lambda_d), \quad \lambda_1 \geq \cdots \geq \lambda_d \geq 0. \quad (2.4)$$

If $v_i$ is the $i$th column of $V$, then

$$\lambda_i = v_i^T Cv_i = v_i^T \left( \int_\Omega j^T ds \right) v_i = \int_\Omega (v_i^T j v_i) ds = \int_\Omega (j^T v_i)^2 ds. \quad (2.5)$$

We examine the Taylor expansion of $f$ at the point $s + h v_i \in \Omega$ around the point $s$,

$$f(s + h v_i) - f(s) = h j(s)^T v_i + \ldots. \quad (2.6)$$

Taking the root-mean-squared of (2.6) and applying (2.5), we get

$$\|f(s + h v_i) - f(s)\|_{L_2} = O(h \sqrt{\lambda_i}), \quad (2.7)$$

where $\| \cdot \|_{L_2}$ is the standard $L_2$ norm for functions defined on $\Omega$. Note that (2.7) implies the following: if $\lambda_i = 0$, then the function $f$ is constant along the direction of $v_i$. We can use this flatness to construct a sampling strategy to approximate $f$ on a low dimensional manifold of $\Omega$.

As an example, consider the function $f(s) = \cos(s_1 + s_2)$ defined on $[-\pi, \pi]^2$; this function is plotted in figure 2.1. The Jacobian of $f$ is

$$\nabla f = \begin{bmatrix} -\sin(s_1 + s_2) & -\sin(s_1 + s_2) \end{bmatrix}. \quad (2.8)$$

The matrix $C$ is then given by

$$C = \int_{-\pi}^{\pi} \int_{-\pi}^{\pi} \begin{bmatrix} \sin^2(s_1 + s_2) & \sin^2(s_1 + s_2) \\ \sin^2(s_1 + s_2) & \sin^2(s_1 + s_2) \end{bmatrix} ds_1 ds_2 = 2\pi^2 \begin{bmatrix} 1 & 1 \\ 1 & 1 \end{bmatrix}. \quad (2.9)$$

The eigenvalue decomposition of $C$ from (2.3) is

$$C = \begin{bmatrix} \sqrt{2}/2 & -\sqrt{2}/2 \\ \sqrt{2}/2 & \sqrt{2}/2 \end{bmatrix} \begin{bmatrix} 4\pi^2 & 0 \\ 0 & 0 \end{bmatrix} \begin{bmatrix} \sqrt{2}/2 & \sqrt{2}/2 \\ -\sqrt{2}/2 & \sqrt{2}/2 \end{bmatrix}. \quad (2.10)$$

Notice that the normalized vector $[\sqrt{2}/2, \sqrt{2}/2]^T$ – the first eigenvector – precisely identifies the direction in the domain along which $f$ varies. Therefore, if we study $f$ along the line defined by $[\sqrt{2}/2, \sqrt{2}/2]^T$, then we can understand the variation in $f$ over the whole domain through a projection.

### 2.1.1. A note on ridge-type functions.

The previous function is an example of a *ridge function*, which appear frequently in statistics [4]. A ridge function takes the form

$$f = f(a^T s) = f(t), \quad a \in \mathbb{R}^d, \ s \in \Omega, \ t = a^T s. \quad (2.11)$$

The Jacobian has a special form in this case:

$$\nabla f = \frac{df}{dt} a^T. \quad (2.12)$$

Then

$$C = v \lambda v^T, \quad \lambda = \frac{||a||^2}{a} \int \left( \frac{df}{dt} \right)^2 dt, \quad ||v|| = 1, \quad (2.13)$$

where the norm $\| \cdot \|$ is the standard 2-norm on $\mathbb{R}^d$. The eigenvector $v$ is a normalized version of $a$ that reveals the direction of variability for $f$. In this case, $C$ has rank one when $df/dt \neq 0$, and the vector $v$ can be computed with one normalized point evaluation of $f(s)$. 
2.2. Dimension Reduction. Assume that the eigenvalue decomposition (2.4) of $C$ can be partitioned as

$$V = [V_a \ V_b], \quad \Lambda = \begin{bmatrix} \Lambda_a & 0 \\ 0 & 0 \end{bmatrix},$$

(2.14)

where $V_a$ has $a$ columns, $V_b$ has $b$ columns, and $a + b = d$. The columns of $V_a$ correspond to directions along which $f$ varies, and the columns of $V_b$ correspond to directions along which $f$ is constant. We can construct a rotated coordinate system since

$$f(s) = f(VV^T s) = f(V_a V_a^T s + V_b V_b^T s) = f(V_a s_a + V_b s_b) \equiv g(s_a, s_b).$$

(2.15)

By construction, the value of the function $g$ will not change as $s_b$ varies. One may be tempted to fix $s_b$ (say, set $s_b = 0$) and treat $g$ as function of the $a$ variables $s_a$. However, there are two issues we must address.

2.2.1. Rotated coordinates. First, what values can $s_a$ take? We can linearly transform the set of points $s \in \Omega$ to get a range for $s_a$. In particular, we define the set

$$\Omega_a = \{s_a : s_a = V_a^T s, \ s \in \Omega\}.$$ 

(2.16)

Since $\Omega$ is convex, $\Omega_a$ is also convex, but this is about all we can say. The coordinates $s_a$ cannot be varied independently within a set of independent intervals like a hyper-rectangle, since the transformed domain $\Omega_a$ will most likely not be a lower dimensional hypercube; imagine taking a photograph of a rotated cube. For this reason, when we construct a surrogate on the reduced coordinates $s_a$, it must be flexible enough to handle general convex domains in multiple dimensions; radial basis functions could be an appropriate choice. We discuss sampling from the space $\Omega_a$ in section 3.4.

2.2.2. The domain of $f$. We must ensure that all function evaluations of $f$ occur at points in the domain $\Omega$; each evaluation of $g$ is ultimately an evaluation of $f$ at specially chosen input values. It is possible that the projection $V_a s_a = V_a V_a^T s$ will not be in $\Omega$, and we do not want to assume anything about $f$ outside its domain.
Fortunately, we can take advantage of the flatness of $f$ to ensure that all evaluations occur within $\Omega$. In short, for any point $V_a s_a$ that falls outside the domain of $f$, we can walk back along the directions in which $f$ is constant until we reach a point in the domain.

More precisely, if $V_a s_a \in \Omega$, then we evaluate $f$ at $V_a s_a$. If $V_a s_a \not\in \Omega$, then we find $z \in \mathbb{R}^b$ such that $V_a s_a + V_b z \in \Omega$. Now we define $g$ as

$$g(s_a) = \begin{cases} f(V_a s_a) & \text{if } V_a s_a \in \Omega, \\ f(V_a s_a + V_b z) & \text{if } V_a s_a \not\in \Omega. \end{cases} \quad (2.17)$$

Note that $z$ is often not uniquely determined, but we only need one for each deviant $s_a$. If $\Omega$ is a hyperrectangle, then a $z$ can be found by solving a suitable linear program; see section 3.4. We have thus achieved our goal of constructing a function $g$ dependent on $a < d$ parameters that behaves like the $d$-variate function $f$.

We demonstrate the rotation and reduction on a slight modification of the previous example. Let $f(s) = \cos(0.3s_1 + 0.7s_2)$ be defined on $[-\pi, \pi]^2$ with gradient

$$\nabla f = \begin{bmatrix} -0.3 \sin(0.3s_1 + 0.7s_2) \\ -0.7 \sin(0.3s_1 + 0.7s_2) \end{bmatrix}. \quad (2.18)$$

Figure 2.2 shows the domain $[-\pi, \pi]^2$ in blue. The projection of the domain onto the line corresponding to the direction of variability of $f$ is shown in red. The red circles correspond to a possible sampling of the reduced (one-dimensional) coordinates. Notice that some of the projected points fall outside $[-\pi, \pi]^2$ when transformed back to the original two-dimensional space. The green circles show the points in $[-\pi, \pi]^2$ that are substituted for the red points outside the domain when evaluating the function $g$.

### 3. Computational aspects

We next consider four computational aspects for the dimension reduction procedure. We close this section with a practical algorithm that summarizes the presentation.

#### 3.1. Low variability versus no variability

In (2.14), we assume that some of the eigenvalues are exactly zero. When this happens, the estimate (2.7) tells us...
that \( f \) is exactly constant along some directions. But what happens when the eigenvalues are small but not zero? The estimate (2.7) addresses an averaged measure of variability along a direction. Like any averaged measure, it does not preclude sharp, local variability. It is possible that one could choose to ignore a direction because its associated eigenvalue is below a specified tolerance but subsequently discover a sharp local feature in \( f \) along this direction.

However, practical considerations like computational budget often dominate the concerns when approximating functions in high dimensions. Any well-motivated strategy to reduce cost is welcome. In this spirit, we treat the magnitudes of the eigenvalues as a ranking on the rotated coordinates. If we desire an approximate bivariate function \( g \) of \( f \), then we choose the directions associated with the two largest eigenvalues.

### 3.2. Approximating the eigenvalues and eigenvectors.

For high dimensional functions found in practice, we expect that there will be a few dominant directions in the sense described above. We usually are not able to compute the exact matrix \( C \) from (2.3), but we can approximate it. Assume for now that we can evaluate the exact Jacobian \( j(s)^T \) given \( s \). Then we can approximate \( C \) with a numerical quadrature rule. For simplicity, we use a Monte Carlo approximation. For \( i = 1, \ldots, k \), let \( s_i \) be samples drawn from \( \Omega \), and compute the \( d \times k \) matrix

\[
J = \begin{bmatrix}
  j(s_1) & \cdots & j(s_k)
\end{bmatrix}.
\]  

Then

\[
C \approx \hat{C} = \frac{\Omega}{k} \sum_{i=1}^{k} j(s_i) j(s_i)^T = \frac{\Omega}{k} JJ^T,
\]

where \( |\Omega| \) is the volume of \( \Omega \). The quality of the approximation can be controlled by the number of samples \( k \). For the Monte Carlo approximation, the variance of the approximation decreases like \( k^{-1/2} \) [7].

Results from eigenvalue perturbation theory show that the error in the approximate eigenvalues is on the order of the error in the matrix elements [6]. More accurate numerical quadrature methods will result in more accurate approximate eigenvalues, but many high order (e.g., interpolatory) multivariate quadrature rules suffer from the same curse of dimensionality that we wish to avoid. For this reason, we rely on Monte Carlo methods.

If \( f \) is constant along some directions, then these directions will be in the null space of \( C \) when \( k \geq d \), i.e., when the number of Jacobian samples is greater than the number of parameters of \( f \). The danger with the approximation \( C \) is potentially overpredicting the dimension of the null space or, equivalently, underpredicting the rank of \( C \). In other words, the Jacobian evaluations at the design sites \( s_i \) may indicate that \( f \) is flat along directions that it actually varies in \( \Omega \).

### 3.3. Approximate Jacobians.

Up to this point, we have assumed that the Jacobian \( j(s)^T \) was available for computation. This is not true in many cases, particularly if \( f \) represents the output of a complex physical simulation. We therefore address the question of approximating the Jacobian from point evaluations of \( f \).

If \( f \) can be evaluated at will, then a finite difference approximation along the original coordinate directions takes \( d + 1 \) evaluations – one at \( s_i \) and one for each perturbation. Thus, approximating \( J \) from (3.1) takes \( k(d + 1) \) function evaluations. The potential benefits of revealing the directions of variability may justify this cost,
particularly if one is faced with a number of evaluations of $f$ that is exponential in $d$

If evaluations of $f$ are very expensive, then we want to obtain the eigenvectors of $\hat{C}$ with as few as possible. We can potentially use fewer than $k(d+1)$ evaluations by employing recently developed methods for matrix completion \[3\] under the assumption that $J$ is row rank deficient, which is equivalent to a rank deficient $\hat{C}$; see \(3.2\). If rank ($J$) = $a$ – corresponding to a function $f$ with $a$ directions of variability – then we can recover $J$ to within the precision of the finite difference approximation by computing a constant times $a(k + d)$ entries of $J$.

Let $D$ be a subset of the pairs of indices $(i,j)$ with $i = 1, \ldots, d$ and $j = 1, \ldots, k$, where $i$ indexes the coordinates $s_i$ and $j$ indexes the design points $s_j$ from \(3.1\). For a $d \times k$ matrix $X$, define $P_D(X)$ to return a vector of the entries of $X$ corresponding to the index pairs in $D$. For a given tolerance $\varepsilon$, the singular value thresholding (SVT) algorithm \[2\] seeks a solution to the convex optimization problem

$$
\begin{align*}
\minimize_{a} & \quad \|X\|_* \\
\text{subject to} & \quad \|P_D(X) - P_D(J)\| \leq \varepsilon,
\end{align*}
$$

where $\| \cdot \|_*$ is the nuclear norm. Note that the finite difference parameter for the approximate Jacobian provides a natural tolerance on the constraints of the convex optimization problem.

In fact, the SVT algorithm returns approximate singular vectors/values for $J$, which saves the trouble of forming $\hat{C}$ with $J$; see \(3.2\). The left singular vectors of $J$ approximate the eigenvectors $V$, and the singular values approximate the square roots of the eigenvalues; see \(2.4\). We can use the output of the SVT method directly to obtain the directions of variability. We demonstrate this approach in the numerical examples in section \(4\).

We will not comment on the cost of the SVT algorithm; we mention it for the case when computing more entries of $J$ through evaluations of $f$ is more expensive than running the SVT algorithm. For example, if $f$ is evaluated with an expensive PDE simulation, and the dimensions of $J$ are in the tens to thousands, then this approach is appropriate.

3.4. Sampling from $\Omega_a$. To sample from the reduced space $\Omega_a$ defined in \(2.16\), we use a simple acceptance/rejection scheme. We first determine an $a$-dimensional hyperrectangle that contains $\Omega_a$ by solving $a$ independent linear programs,

$$
\begin{align*}
\minimize_s & \quad v^T_s t_s \\
\text{subject to} & \quad s_t \leq s \leq s_a,
\end{align*}
$$

where $v_t$ is the $i$th column of $V_a$. Let $s_t^*$ be the minimizer of \(3.4\). Then we define the hyperrectangle $\tilde{\Omega}_a$ as

$$
\tilde{\Omega}_a = \left\{ t = \begin{bmatrix} t_1 \\ \vdots \\ t_a \end{bmatrix} : \begin{bmatrix} V^T_1 s_1^* \\ \vdots \\ V^T_a s_a^* \end{bmatrix} \leq \begin{bmatrix} t_1 \\ \vdots \\ t_a \end{bmatrix} \leq \begin{bmatrix} -V^T_1 s_1^* \\ \vdots \\ -V^T_a s_a^* \end{bmatrix} \right\}. \tag{3.5}
$$

Notice that $\Omega_a \subset \tilde{\Omega}_a$, and we expect that the volume of the enclosing hyperrectangle will be much larger than the volume of $\Omega_a$ in high dimensions.

To draw a sample $s_a$ from $\Omega_a$, we draw $t$ uniformly from $\tilde{\Omega}_a$. If $V_a t \in \Omega$, then we set $s_a = t$. If $V_a t \not\in \Omega$, but there exists a $z$ such that $V_a t + V_a z \in \Omega$, then we also
set \( s_a = t \). To determine if such a point exists, we can attempt to solve the linear program,

\[
\begin{aligned}
\text{minimize} & \quad 0^T s, \\
\text{subject to} & \quad s_a = V_T^T s, \\
& \quad s_l \leq s \leq s_u.
\end{aligned}
\] (3.6)

If a point \( s^* \) is found that satisfies the constraints, then \( z = V_T^T s^* \). If such a \( z \) does not exist, then we reject \( t \). Notice that the objective function in (3.6) is essentially meaningless; it is merely used to set the problem in terms easily entered into a linear program solver.

Each sample from \( \Omega_a \) is used to evaluate \( g(s_a) \) as in (2.17), which we use to construct a surrogate on the low dimensional subspace.

3.5. A practical algorithm. We have now discussed all the pieces in the procedure for approximating \( f \) on the low dimensional manifold.

1. **Compute the directions.** If one can evaluate \( j(s) \), choose points \( s_i \in \Omega \) with \( i = 1, \ldots, k \) and compute

\[
J = \begin{bmatrix} j(s_1) & \cdots & j(s_k) \end{bmatrix}, \quad \frac{|\Omega|}{k} JJ^T = \hat{C} = V \Lambda V^T. \tag{3.7}
\]

If one can only evaluate \( f(s) \), use the procedure from section 3.3 to approximate the eigendecomposition of \( \hat{C} \).

2. **Determine the directions of variability.** Examine the eigenvalues \( \lambda_1, \ldots, \lambda_d \) and choose a truncation \( a < d \) according to their magnitude. (This judgment can be difficult to make algorithmically.) Set \( V_a \) to be the first \( a \) eigenvectors.

3. **Evaluate \( g \) at points in the reduced domain.** Use the acceptance/rejection method from section 3.4 to choose a set of design points \( y_j \in \Omega_a \) for \( j = 1, \ldots, n \). For each design point, compute \( g_j = g(y_j) \) using (2.17). Note that each evaluation may require the computation of \( z_j \) using the approach described in section 3.4.

4. **Approximate \( f \) at a point in \( \Omega \).** For a point \( s \in \Omega \), compute \( s_a = V_T^T s \). Approximate \( g(s_a) \) using an interpolation procedure on the points \( \{y_j\} \) and evaluations \( \{g_j\} \). This approximation occurs on the space \( \Omega_a \) of reduced dimension. Set \( f(s) \) to be the approximation of \( g(s_a) \).

Once the first three steps have been completed, the last step can be repeated as needed. For example, numerical integration or optimization can be performed on \( f(s) \) using the surrogate constructed on the reduced space \( \Omega_a \).

4. **Numerical Examples.** In this numerical exercise, we perform an uncertainty study on an elliptic PDE with a random field model for the coefficients. Such problems are common test cases for methods in uncertainty quantification [1, 5].

4.1. **PDE model, input parameters, and quantity of interest.** Consider the following linear elliptic PDE. Let \( u = u(x, s) \) satisfy

\[
-\nabla \cdot (\alpha \nabla u) = 1 \tag{4.1}
\]

on the spatial domain \( x \in [0, 1]^2 \). We set homogeneous Dirichlet boundary conditions on the left, top, and bottom of the domain; denote this boundary by \( \Gamma_1 \). The right side of the domain — denoted \( \Gamma_2 \) — has a homogeneous Neuman boundary condition.
The log of the coefficients $\alpha = \alpha(x,s)$ of the differential operator are given by a truncated Karhunen-Loeve type expansion

$$\log(\alpha) = \sum_{i=1}^{d} \phi_i \sigma_i s_i,$$  \hfill (4.2)

where the $s_i$ are independent, identically distributed uniform random variables on $[-2, 2]$, and the $\{\phi_i, \sigma_i\}$ are the eigenpairs of the covariance operator

$$C(x, y) = \exp \left\{ - \left( \frac{(x_1 - y_1)^2}{\rho_1} + \frac{(x_2 - y_2)^2}{\rho_2} \right) \right\},$$  \hfill (4.3)

with $\rho_1 = 1$ and $\rho_2 = 0.05$. The small $\rho_2$ models a short correlation length in the vertical coordinate. The decay of the $\sigma_i$ justifies a truncation of $d = 250$, so that the parameter space $\Omega$ for the problem is the 250-dimensional hypercube $[-2, 2]^d$. Three realizations of the log of the coefficients $\alpha$ and their corresponding solutions $u$ are shown in figures 4.1 and 4.2, respectively.

Define the linear function $\tilde{Q} = \tilde{Q}(s)$ of the solution

$$\tilde{Q} = \frac{1}{|\Gamma_2|} \int_{\Gamma_2} u \, dx.$$  \hfill (4.4)

The quantity of interest for the uncertainty study is an approximate density function for $\tilde{Q}$.

4.2. Finite element discretization. Given a value for the input parameters $s$, we discretize the elliptic problem with a standard linear finite element method using MATLAB’s PDE Toolbox. The discretized domain has 34320 triangles and 17361 nodes; the eigenfunctions $\phi_i = \phi_i(x)$ from (4.2) are approximated on this mesh. The matrix equation for the discrete solution $u = u(s)$ at the mesh nodes is

$$Ku = f,$$  \hfill (4.5)

where $K = K(s)$ is symmetric and positive definite for all $s \in \Omega$. We can approximate the linear functional $\tilde{Q}$ as

$$\tilde{Q} \approx c^T u = Q,$$  \hfill (4.6)

where the elements of $c$ are zero except corresponding to nodes on $\Gamma_2$. The nonzero elements are constant and scaled so that they sum to one; note that $c$ does not depend on $s$.

4.3. Adjoint variables for derivatives. Since the quantity of interest can be written as a linear functional of the solution, we can define adjoint variables that we will help us compute the Jacobian of $Q$ with respect to the input parameters $s$. Notice that we can write

$$Q = c^T u = c^T u - y^T (Ku - f),$$  \hfill (4.7)

for any constant vector $y$. Taking the derivative of (4.7) with respect to the input $s_i$, we get

$$\frac{\partial Q}{\partial s_i} = c^T \left( \frac{\partial u}{\partial s_i} \right) - y^T \left( \frac{\partial K}{\partial s_i} u + K \frac{\partial u}{\partial s_i} \right)$$

$$= (c^T - y^T K) \left( \frac{\partial u}{\partial s_i} \right) - y^T \left( \frac{\partial K}{\partial s_i} \right) u.$$
If we choose \( y \) to solve the adjoint equation
\[
K^T y = c, \tag{4.8}
\]
then
\[
\frac{\partial Q}{\partial s_i} = -y^T \left( \frac{\partial K}{\partial s_i} \right) u. \tag{4.9}
\]

Three realizations of the adjoint variables \( y = y(s) \) are shown in figure 4.3.

To approximate the Jacobian \( \nabla Q \) at the point \( s \), we compute the finite element solution with (4.5), solve the adjoint problem (4.8), and compute the components with (4.9). The derivative of \( K \) with respect to \( s_i \) is easy compute from the derivative of \( a(x, s) \) and the same finite element discretization.

4.4. Approximating the subspace. To apply the input reduction with the subspace detection technique, we first sample the Jacobian at random points in \( \Omega \) to construct \( J \) from (3.1). From a reference computation of $10^4$ samples, we examine the singular values to determine an appropriate truncation. The singular values of \( J \) are plotted in Figure 4.4; the decay justifies a truncation after five terms. For reference, we also plot the singular values $\sqrt{\sigma_i}$ from the Karhunen-Loeve expansion (4.2). The more rapid decay of the singular values of \( J \) shows that the particular
output quantity of interest depends primarily on fewer variables than the correlated random field modeling the coefficients of the differential operator.

In the remainder of the numerical exercise, we split the reference $10^4$ samples into five groups of 2000 samples. This is to mimic an initial computational budget of 2000 samples, which we repeat five times to mildly alleviate affects associated with a particularly good or bad choice of 2000 samples. Figures will display the results of each of the five independent experiments.

To check convergence of the projection onto the reduced subspace as more Jacobian samples are added, we compute the left singular vectors of $J$ for $m = 100, 200, 300, \ldots, 2000$. The difference between the subspaces defined by subsequent sets of samples $m_i$ and
Fig. 4.5: The convergence of subspaces defined by taking the first $a = 5$ left singular vectors from five different random samplings of $J$. The index $i$ corresponds to adding more samples to $J$. Figure 4.5a shows the error between the subspace from a reference solution of $10^4$ samples and the increasing the number of samples in $J$; see (4.11). Figure 4.5b shows the relative change in the subspaces as more samples are added; see (4.10).

$m_{i+1}$ is given by

$$E_{i}^{rel} = \|V_{a,m_{i+1}}V_{a,m_{i+1}}^T - V_{a,m_{i}}V_{a,m_{i}}^T\|,$$  \hspace{1cm} (4.10)

where $V_{a,m_{i}}$ are the first $a$ left singular vectors of $J$ approximate with $m_{i}$ samples, and the norm is the matrix 2-norm. In figures 4.5a-4.5b, we plot $E_{i}^{rel}$ along with the error of the projected subspace compared to the reference solution,

$$E_{i}^{abs} = \|V_{a,m_{i}}V_{a,m_{i}}^T - V_{a,ref}V_{a,ref}^T\|,$$  \hspace{1cm} (4.11)

where $V_{a,ref}$ are the first $a$ eigenvectors from the reference computation of $J$.

Each column of $J$ requires a forward solve, an adjoint solve, and 250 computations for the gradient $\nabla Q$. We can try to reduce the number of derivative computations with the SVT algorithm for matrix completion, as described in section 3.3. Using the left singular vectors of $J$ computed with $m = 2000$ samples, we can test the SVT method by uniformly subsampling the entries of $J$. The number of subsampled entries is controlled by $\gamma$ with $0 < \gamma < 1$, which is the proportion of entries revealed in the incomplete matrix. In figure 4.6, we plot the difference between the subspace from the subsampled $J$ and the subspace from the full $J$ for $\gamma = 0.1, 0.2, \ldots, 0.9$,

$$E_\gamma = \|V_{a,\gamma}V_{a,\gamma}^T - V_{a}V_{a}^T\|.$$  \hspace{1cm} (4.12)

For the SVT algorithm, we used the MATLAB implementation from [2] with the following parameters: objective parameter $\text{tau}=100$, stopping criterion $\text{tol}=1e^{-4}$, noise constraint $\text{EPS}=1e^{-6}$, step size $\text{delta}=1$, and maximum iterations $\text{maxiter}=1000$.

4.5. Building the low dimensional surrogate. We first map the 2000 design sites $s_i \in \Omega$ to get an initial set of design sites $y_i = V_a^T s_i \in \Omega_a$; we have the evaluations of $Q$ associated with these points from the initial sampling of the Jacobian. Following
the process outlined in section 3.4 we sample uniformly from the space $\Omega_a$ to find 5000 additional design sites in the lower dimension subspace. The acceptance rate of the acceptance/rejection scheme is roughly 35% (averaged over the five identical experiments), which validates the intuition about the small volume of $\Omega_a$ relative to its enclosing hyperrectangle. It took an average of 19201 linear programs to get the 5000 samples.

For each sample, we evaluate $Q$ from (4.6). This gives us a total of 7000 points in the lower dimensional subspace – 2000 from the original Jacobian evaluations and 5000 from sampling on the reduced subspace $\Omega_a$ – on which to construct a surrogate. We use the kriging toolbox DACE [11] to build a surrogate on the lower dimensional space $\Omega_a$.

Since the evaluation of $Q$ is relatively inexpensive in this example, we compare the surrogate’s prediction of $Q$ with the actual $Q$ on $10^5$ points chosen uniformly at random from $\Omega$. The histograms of the log of the surrogate error are shown in figure 4.7 – one for each of the five experiments.

4.6. **Approximating the density function.** To approximate the density function of $Q$, we draw samples from the full space $\Omega$, use the low dimensional surrogate to approximate the output quantity of interest, and build a histogram of the samples. Specifically, for a point $s \in \Omega$, we compute $s_a = V_s^T s$. Then we use the low dimensional surrogate to approximate $Q$ at $s_a$. Using $10^5$ such evaluations, we obtain reasonably well-converged histograms. In figure 4.8 we plot the histogram of $10^5$ evaluations of $Q$ from the full model alongside the histograms from each surrogate experiment. We see that surrogate approximates the bulk of the histogram reasonably but loses accuracy near the tails. This is expected; the low dimensional subspace is detected by averaged variability. Therefore, we do not expect to capture extremes of
P. G. CONSTANTINE AND Q. WANG

Fig. 4.7: Histograms of the log10 of the error between the surrogate approximation of $Q$ and the true value of $Q$ at $10^5$ randomly sampled points from the full space $\Omega$. Each histogram corresponds to a different random sampling used to construct the surrogate.

$Q$, and this is reflected in a loss of accuracy in the tails.

5. Conclusion. We have presented a method for detecting the primary directions of variability of a function of many variables. We have described how to exploit these directions to construct a surrogate on a low dimensional subspace of the high dimensional input space. We demonstrated this procedure on an uncertainty quantification study with a model problem of an elliptic PDE with variable coefficients that depend on 250 independent input parameters.

REFERENCES

[1] I. Babuška, M. K. Deb, and J. T. Oden, Solution of stochastic partial differential equations using Galerkin finite element techniques, Computer Methods in Applied Mechanics and Engineering, 190 (2001), pp. 6359–6372.

[2] J.-F. Cai, E. J. Candes, and Z. Shen, A singular value thresholding algorithm for matrix completion, SIAM Journal on Optimization, 20 (2010), pp. 1956–1982.

[3] E. Candes and Y. Plan, Matrix completion with noise, Proceedings of the IEEE, 98 (2010), pp. 925 –936.

[4] A. Cohen, I. Daubechies, R. DeVore, G. Kerkyacharian, and D. Picard, Capturing ridge functions in high dimensions from point queries, Constructive Approximation, pp. 1–19. 10.1007/s00365-011-9147-6.

[5] R. Ghanem and P. D. Spanos, Stochastic Finite Elements: A Spectral Approach, Springer-Verlag, New York, 1991.

[6] G. H. Golub and C. F. VanLoan, Matrix Computations, The Johns Hopkins University Press, Baltimore, MD, 3rd ed., 1996.
Fig. 4.8: Histograms of the quantity of interest $Q$ with $10^5$ samples from the input space $\Omega$. The top right histogram 4.8a is the histogram of the full model. The others are for the five experiments from the low dimensional surrogate.

[7] M. H. Kalos and P. A. Whitlock, Monte Carlo Methods, Wiley-VCH, 2nd ed., 2008.
[8] G. Li, S.-W. Wang, and H. Rabitz, Practical approaches to construct RS-HDMR component functions, The Journal of Physical Chemistry A, 106 (2002), pp. 8721–8733.
[9] C. Lieberman, K. Willcox, and O. Ghattas, Parameter and state model reduction for large-scale statistical inverse problems, SIAM Journal on Scientific Computing, 32 (2010), pp. 2523–2542.
[10] R. Liu and A. B. Owen, Estimating mean dimensionality of analysis of variance decompositions, Journal of the American Statistical Association, 101 (2006), pp. 712–721.
[11] S. Lophaven, H. Nielsen, and J. Sondergaard, DACE: A Matlab Kriging toolbox. IMM Technical University of Denmark, 2002. [http://www2.imm.dtu.dk/~hbn/dace/](http://www2.imm.dtu.dk/~hbn/dace/)
[12] A. Saltelli, M. Ratto, T. Andres, F. Campolongo, J. Cariboni, D. Gatelli, M. Saisana, and S. Tarantola, Global Sensitivity Analysis. The Primer, John Wiley & Sons, Ltd, 2008.