Active Subspaces
Josef Málek and Zdeněk Strakoš, *Preconditioning and the Conjugate Gradient Method in the Context of Solving PDEs*

Paul G. Constantine, *Active Subspaces: Emerging Ideas for Dimension Reduction in Parameter Studies*
Active Subspaces

Emerging Ideas for Dimension Reduction in Parameter Studies

Paul G. Constantine
Colorado School of Mines
Golden, Colorado
Contents

Preface vii

1 Quick Start 1
1.1 Gradients or no gradients? .......................... 3
1.2 Evaluating the dimension of the active subspace 6
1.3 Sufficient summary plots .......................... 7
1.4 An example with a parameterized PDE .......... 10

2 Parameterized Models in Physics and Engineering 13
2.1 What is random? .................................. 14
2.2 The parameter studies .............................. 15
2.3 Too many parameters! ............................. 15
2.4 Dimension reduction: Subsets and subspaces .... 16

3 Discover the Active Subspace 21
3.1 Parameterized simulations and \( f(x) \) .......... 21
3.2 Defining the active subspace ..................... 22
3.3 Computing the active subspace ................... 25
3.4 A practical recipe ................................ 35

4 Exploit the Active Subspace 45
4.1 Dimension reduction and mappings ............. 45
4.2 Response surfaces ................................ 48
4.3 Integration ...................................... 59
4.4 Optimization .................................... 62
4.5 Inversion ...................................... 65

5 Active Subspaces in Action 71
5.1 HyShot II scramjet ................................. 71
5.2 Photovoltaic solar cell ............................ 78
5.3 Airfoil shape optimization ....................... 81

6 Summary and Future Directions 87
6.1 Multiple outputs .................................. 87
6.2 Coupled systems .................................. 88
6.3 Anticipating active subspaces .................... 88

Bibliography 89

Index 99
Preface

Parameter studies are everywhere in computational science. Complex engineering simulations must run several times with different inputs to effectively study the relationships between inputs and outputs. Studies like optimization, uncertainty quantification, and sensitivity analysis produce sophisticated characterizations of the input/output map. But thorough parameter studies are more difficult when each simulation is expensive and the number of parameters is large. In practice, the engineer may try to limit a study to the most important parameters, which effectively reduces the dimension of the parameter study.

Active subspaces offer a more general approach to reduce the study’s dimension. They identify a set of important directions in the input space. If the engineer discovers a model’s low-dimensional active subspace, she can exploit the reduced dimension to enable otherwise infeasible parameter studies for expensive simulations with many input parameters. This book develops active subspaces for dimension reduction in parameter studies.

My journey with active subspaces started at Stanford’s Center for Turbulence Research Summer Program in 2010 with Gianluca Iaccarino (Stanford), Alireza Doostan (CU Boulder), and Qiqi Wang (MIT). We were studying surrogate models for calibrating the six inputs of an expensive scramjet simulation with Markov chain Monte Carlo. After struggling for two weeks to get the chain to converge, Qiqi came to the morning meeting touting some “magic parameters.” He had found three orthogonal directions in the six-dimensional parameter space such that small input perturbations along these directions barely changed the pressure at prescribed sensor locations. The remaining three orthogonal directions gave three linear combinations of the six parameters; perturbing the latter linear combinations significantly changed the pressure, so they were more important for the calibration. He had found these directions by studying the pressure’s gradient with respect to the input parameters. Lacking a technically descriptive name, Qiqi called the three important linear combinations magic. Given the scramjet runs we had available, he could build a regression surface in three variables with a higher degree polynomial than in six variables. We all thought this was a really cool idea.

The work was published in the Summer Program’s proceedings and later as an AIAA conference paper [35]. But then it sat on the back burner for a while. In late 2011, I started bugging Qiqi that we should write a journal-worthy version of this input dimension reduction idea with more precise statements and formal development. Since so many uncertainty quantification techniques suffered from the dreaded curse of dimensionality, I thought reducing the dimension of a model’s inputs could have a big impact. It took a while to get the formulation right, and we went back and forth with the reviewers. Amid that exchange, I found Trent Russi’s 2010 Ph.D. thesis [108], in which he proposed a similar idea for reducing a model’s input dimension with subspaces derived from random samples of the gradient; he called these subspaces active subspaces. We thought that was a catchy and reasonably descriptive name, so we adopted it.
The paper was finally accepted and appeared as [28]. Around the same time, Qiqi and I connected with Yousef Marzouk (MIT) and Tan Bui (UT Austin), who were working on techniques for improving Markov chain Monte Carlo for Bayesian inverse problems—including some subspace-based dimension reduction ideas. We put together a proposal for the DOE’s Advanced Scientific Computing Research Applied Mathematics program, and we were awarded a three-year grant to develop active subspaces for data-intensive inverse problems. That grant has supported my effort to write this monograph.

Audience. Computational science is inherently interdisciplinary. I hope that this text offers something useful and interesting to researchers across fields. Dimension reduction must be practical and well-motivated—with easy-to-implement algorithms and easy-to-interpret results—to impact real applications. But the techniques must also be theoretically well-founded with rigorous performance guarantees for cases that satisfy simplifying assumptions. When simplified cases lack the challenges of real applications, engineers accuse the analysts of working on toy problems. When an engineering method has no rigorous performance guarantees, the analysts deride the method as heuristic—to which the engineer may respond that it works well in practice.

These squabbles are unlikely to subside in the near future, and this book will not resolve them. However, I have tried to balance both objectives. I have tried to develop the material so that it is amenable to analysis. And I have tried to demonstrate the value of active subspaces in engineering applications. I hope that both groups will benefit. There is still plenty to do on both fronts—analysis and engineering. Computational science graduate students may find a new perspective on their research with these techniques and subsequently advance the state of the art.

Outline. The first chapter provides a quick start with active subspaces for engineers paralyzed by their high-dimensional parameter studies. It offers some easy-to-implement procedures for discovering whether the given model admits an active subspace. If these tests show evidence of an active subspace, then the practitioner should be well-motivated to read on. The first chapter is reasonably self-contained. As such, I repeat some of the material in later chapters for further development.

The second chapter puts active subspaces into the broader context of algorithm research in uncertainty quantification. It includes important references to related statistics research on sufficient dimension reduction that blossomed more than 20 years ago.

The third and fourth chapters develop the technical content, including defining the active subspace, proposing and analyzing a method to discover the active subspace, and discussing strategies to exploit the active subspace for high-dimensional parameter studies.

The fifth chapter demonstrates the utility of active subspaces in three engineering applications: (i) studying the safe operating regime of a hypersonic scramjet, (ii) characterizing the relationship between model inputs and maximum power in a photovoltaic solar cell model, and (iii) optimizing the shape of an airfoil. Each section has just enough detail to put the application in the mathematical framework for active subspaces.

The quickest way to impact applications is to provide easy-to-use software. I have not included any language-specific implementations in the text—mostly because of the rapid pace of innovation in software and computing. However, I maintain the website activesubspaces.org, and I provide scripts and utilities there for working with active subspaces.

Acknowledgments. Throughout the text, I use the first person plural we1 to reflect that the work in this book has been a collaborative effort. Although the choice and arrangement of words and symbols are my own, the intellectual content contains

1In the first chapter, I use the second person singular you to make directions easier to read.
significant contributions from my esteemed colleagues. My own interests cannot possibly cover the breadth of expertise needed for interdisciplinary computational science research—especially the domain expertise in the applications in Chapter 5. The final work would not have been possible without my colleagues’ efforts.

I am very grateful for Qiqi Wang and Eric Dow at MIT. Much of the technical development in Chapters 3 and 4 comes from our paper [28]. David Gleich (Purdue) and I worked together to analyze the random sampling method for estimating active subspaces presented in Chapter 3, and his help with the presentation and several tricky turns in the proofs was invaluable [30]. Youssef Marzouk (MIT) and Tan Bui (UT Austin) helped refine my ideas on Bayesian inverse problems in Chapter 4. The scramjet application from Chapter 5 was a primary objective at Stanford’s NNSA-funded Predictive Science Academic Alliance Program center. The specific work with active subspaces was performed jointly with Michael Emory (Stanford), Johan Larsson (UMD College Park), and Gianluca Iaccarino (Stanford) [29]. Their comments from the engineering side helped ensure that active subspaces could be valuable in real applications. The photovoltaics application in Chapter 5 consists of joint work with Brian Zaharatos (Colorado School of Mines) and Mark Campanelli (NREL) [37]. The airfoil shape optimization in Chapter 5 consists of joint work with Trent Lukaczyk, Francisco Palacios, and Juan Alonso at Stanford’s Aerospace Design Laboratory [87]. I am especially thankful for Trent’s help in running the simulations, preparing the figures, and interpreting the results in section 5.3. Lastly, I am grateful for the helpful suggestions and support from Ralph Smith (NCSU) and the SIAM Spotlights anonymous reviewers. Their comments helped me to clarify the presentation.

I have been extremely fortunate to receive funding from the U.S. Department of Energy. As a postdoc at Stanford, I was funded by the National Nuclear Security Administration under Award NA28614 through the Predictive Science Academic Alliance Program. As an assistant professor at Colorado School of Mines, my efforts have been supported by the Office of Science Advanced Scientific Computing Research Applied Mathematics program under Award DE-SC-0011077.
Chapter 1
Quick Start

We assume that you are here with a computer simulation of a complicated physical model that includes several input parameters. You want to perform some sort of parameter study—such as optimization, uncertainty quantification, or sensitivity analysis—with the simulation’s predictions as a function of its inputs. But the methods you know for such studies are not practical with all of your parameters because of your limited computational budget. In other words, the methods need too much computation in high dimensions—more computation than you have available.

If you knew that some of the inputs would not affect the results of the parameter study, then you could ignore them. But we assume that you do not know this beforehand—at least not with certainty. One strategy to enable your parameter study would be to use some of the computational budget to discover which input parameters are important and which you can safely ignore. If you can ignore some inputs, then we say that you have reduced the dimension of the parameter study. In the best case, you reduce the dimension to the point that a thorough parameter study becomes feasible within your budget.

Active subspaces offer one appealing approach for this type of dimension reduction. However, instead of identifying a subset of the inputs as important, active subspaces identify a set of important directions in the space of all inputs. Each direction is a set of weights that define a linear combination of the inputs. If the simulation’s prediction does not change as the inputs move along a particular direction, then we can safely ignore that direction in the parameter study. For example, Figure 1.1 plots the function $\exp(0.7x_1 + 0.3x_2)$. The arrows indicate the direction $[0.7, 0.3]$ along which this function varies the most and the orthogonal direction $[-0.3, 0.7]$ along which it is constant. This generalizes the idea of identifying a subset of important parameters, since a particular function could have important directions aligned with the input coordinates.

The rest of this chapter offers some easy-to-implement procedures for determining whether your simulation is eligible for dimension reduction with active subspaces. We assume you have the following pieces available:

1. a simulation model with $m$ well-defined inputs and a scalar quantity of interest,
2. a range for each of the independent input parameters,
3. resources for running the simulation multiple times.

Some comments on these quantities are in order. First, the quantity of interest should be a smooth function of the input parameters. Generic discontinuities wreak havoc in the
random sampling-based algorithms we use. Second, you may have some choice in defining the ranges of the input parameters. Make sure that your model is well-defined and numerically well-behaved for all parameter values within the ranges. To ensure this, you may need to adjust meshes or solver tolerances for different values of the inputs. Beware of asymptotes in your quantity of interest; unbounded growth can also cause problems in sampling-based approaches. Table 1.1 sets up working notation to describe the procedures, and we begin with a few preliminary preparations.

1. **Normalized inputs.** We need to normalize the input parameters to be centered at zero with equal ranges. Normalizing removes units and ensures that parameters with relatively large values do not disproportionately affect the analysis. Consider a vector \( \mathbf{x} \) with \( m \) components, each between \(-1\) and \(1\). If \( \mathbf{x}_l \) and \( \mathbf{x}_u \) are \( m \)-vectors containing the lower and upper bounds of the simulation’s inputs, respectively, then the scaled and shifted vector

\[
\frac{1}{2} \left( \text{diag}(\mathbf{x}_u - \mathbf{x}_l) \mathbf{x} + (\mathbf{x}_u + \mathbf{x}_l) \right)
\]

contains the natural inputs of the simulation.
1.1. Gradients or no gradients?

2. **Sampling density.** The check for dimension reduction is based on randomly sampling the normalized inputs and running the model for each sample; random sampling is a powerful tool for working in high-dimensional spaces. If your model is not naturally endowed with such a density (most are not), then you must choose one. We suggest choosing something easy to work with, such as a uniform density over the hypercube \([-1, 1]^m\). We denote this density by \(\rho\). The results depend on the choice of \(\rho\), but that dependence may be weak. If you are worried by such dependence, you should repeat the analysis with different \(\rho\)'s to see how the results change.

3. **The largest dimension.** Think ahead to how you plan to use the low-dimensional model. For example, if you ultimately want to build response surfaces, you may not be willing to work in more than three or four dimensions. Let \(k\) be one greater than the largest dimension that you can handle in subsequent use. In the worst case, you can choose \(k = m + 1\), which is one more than the number of input parameters. However, you will save substantial computational effort if you choose \(k\) less than \(m\).

4. **An oversampling factor.** Rules of thumb in random sampling algorithms include an oversampling factor. For example, it might be possible to fit a linear model with only \(m + 1\) evaluations of the quantity of interest. But it is safer to fit the model with evaluations between \(2m\) and \(10m\) evaluations. Choose an oversampling factor \(\alpha\) to be between 2 and 10.

With \(\rho\), \(k\), and \(\alpha\) defined, we proceed to the computation.

### 1.1 - Gradients or no gradients?

There are several ideas based on subspaces for reducing the dimension of the input space, many of which were developed in the context of regression models in statistics [38]. The active subspace in particular is derived from the gradient \(\nabla f(x)\), and discovering an active subspace requires the capability to evaluate the gradient—or at least an approximate gradient—at any \(x\) in the space of input parameters. Fortunately, more and more simulation codes have gradient capabilities thanks to technologies such as adjoint solvers and algorithmic differentiation, although these are often absent in legacy codes and simulations that include multiple, coupled components.

If your simulation has gradient capabilities, start with Algorithm 1.1, which requires \(M = \alpha k \log(m)\) evaluations of the gradient. (See Chapter 3 for how we derived this number.) This algorithm produces a set of eigenvectors and eigenvalues. We use the eigenvalues to determine the dimension of the active subspace, and the corresponding eigenvectors define the active subspace. If the number \(m\) of input parameters is in the tens to thousands, then this eigenvalue decomposition is quick to compute on a modern laptop.

If you do not have gradients, but your simulation is amenable to finite difference approximations of the gradient—i.e., numerical noise in your quantity of interest is small enough to use finite differences—then you can use Algorithm 1.1 with finite differences. First-order finite differences cost \(m + 1\) simulation runs per gradient evaluation—one for the point in the input space where the approximate gradient is evaluated and one per perturbed parameter. The total cost of Algorithm 1.1 with first-order finite differences is \(N = \alpha k(m + 1) \log(m)\) total simulations.
Chapter 1. Quick Start

**Algorithm 1.1.** Active subspace estimation with gradients.

1. Draw $M = \alpha k \log(m)$ independent samples $\{x_i\}$ according to the sampling density $\rho$.
2. For each sample $x_i$, compute the gradient $\nabla_x f_i = \nabla f(x_i)$ and the quantity of interest $q_i = f(x_i)$.
3. Compute the matrix $\hat{C}$ and its eigenvalue decomposition,
   \[ \hat{C} = \frac{1}{M} \sum_{i=1}^{M} \nabla_x f_i \nabla_x f_i^T = \hat{W} \hat{\Lambda} \hat{W}^T, \]  
   where $\hat{W}$ is the matrix of eigenvectors, and $\hat{\Lambda} = \text{diag}(\hat{\lambda}_1, \ldots, \hat{\lambda}_m)$ is the diagonal matrix of eigenvalues ordered in decreasing order.

If you do not have access to gradients, and finite differences are infeasible, you can build a model of $f(x)$ to approximate gradients. For example, if you fit a polynomial model to $f(x)$, then computing gradients of the polynomial model is straightforward. But fitting global multivariate polynomial models of degree greater than one requires many simulation runs. Kernel-based response surfaces, e.g., radial basis functions or Gaussian processes, suffer from the same scalability issue. Fortunately, we are primarily interested in local behavior when approximating gradients. We propose the following algorithm based on local linear models to estimate the eigenpairs from Algorithm 1.1 when gradients are not available. Each local linear model is fit with a subset of the predictions from a set of randomly sampled runs.

**Algorithm 1.2.** Active subspace estimation with local linear models.

1. Choose $N \geq \alpha m$, $M = \alpha k \log(m)$, and an integer $p$ such that $m + 1 \leq p \leq N$.
2. Draw $N$ independent samples $\{x_j\}$ according to the density $\rho$, and compute $q_j = f(x_j)$ for each sample.
3. Draw $M$ independent samples $\{x'_i\}$ according to $\rho$.
4. For each $x'_i$, find the $p$ points from the set $\{x_j\}$ nearest to $x'_i$; denote this set by $\mathcal{X}_i$.
   Let $\mathcal{Q}_i$ be the subset of $\{q_j\}$ that corresponds to the points in $\mathcal{X}_i$.
5. Use least-squares to fit the coefficients $c_i$ and $b_i$ of a local linear regression model,
   \[ q_j \approx c_i + b_i^T x_j, \quad x_j \in \mathcal{X}_i, q_j \in \mathcal{Q}_i. \]  
   \[ (1.3) \]
6. Compute the matrix $\hat{C}$ and its eigenvalue decomposition,
   \[ \hat{C} = \frac{1}{M} \sum_{i=1}^{M} b_i b_i^T = \hat{W} \hat{\Lambda} \hat{W}^T. \]  
   \[ (1.4) \]

---

2 First-order finite differences are partial derivatives of local linear interpolants.
1.1. Gradients or no gradients?

There is a lot to say about Algorithm 1.2. Note that it needs two additional parameters: (i) the number \( N \) of initial runs that are used to build the local linear models, and (ii) the parameter \( p \) that determines the number of runs used to fit each local linear model. We are actively analyzing Algorithm 1.2. We hope to derive quantitative statements about the quality of the approximated \( \hat{W} \) and \( \hat{\Lambda} \) along with more precise guidance for the choices of \( N \) and \( p \). For now we are guided by intuition, a few observations, and our experience that this method has revealed active subspaces in real applications.

First, in step 2 we choose the points \( \{ x_j \} \) independently at random according to \( \rho \). There may be better choices for \( \{ x_j \} \)—e.g., quasi–Monte Carlo points [100]—that properly cover the space of inputs when \( m < 10 \). In general, the points \( \{ x_j \} \) need to resolve the essential features of \( f(x) \) for the local linear models to accurately approximate the gradients. Second, the algorithm assumes that each least-squares problem in step 5 is well-posed. If the matrix used to fit the linear model is rank deficient, then the fit needs either regularization or a larger value for \( p \); including more neighbors when fitting the linear model may make the least-squares problem well-posed.

If \( p = m + 1 \), then each local linear model interpolates the predictions in \( \mathcal{D} \). If \( p = N \), then there is only one global linear model, and \( \hat{\mathcal{C}} \) is rank one. The case when \( p = N \) is sufficiently interesting to warrant its own algorithm, which reduces to fitting a global linear model and computing the normalized gradient. The global linear model is interesting for two reasons. First, it’s so cheap! It only requires enough runs to fit a linear model. Second, we have used this approach to discover a one-dimensional active subspace in several real applications; see section 5.1 for an example.

**Algorithm 1.3.** Active subspace estimation with a global linear model.

1. Draw \( N = \alpha m \) independent samples \( \{ x_j \} \) from the sampling density \( \rho \).
2. For each sample \( x_j \), run the simulation and compute the quantity of interest, \( q_j = f(x_j) \).
3. Use least-squares to compute the coefficients \( c \) and \( b \) of the linear model
   \[
   q_j \approx c + b^T x_j, \quad j = 1, \ldots, N. \tag{1.5}
   \]
4. Compute the normalized gradient of the linear model
   \[
   \hat{w} = b/\|b\|. \tag{1.6}
   \]

Algorithm 1.3 first fits a linear model of the quantity of interest as a function of the simulation’s input parameters using \( N = \alpha m \) simulation runs. The direction \( \hat{w} \) in (1.6) is the normalized gradient of the fitted linear model. If the map from inputs to quantity of interest is actually a linear function, then \( \hat{w} \) is equivalent to the first eigenvector from the gradient-based Algorithm 1.1; however, in general they are different. The global linear model can produce only a single direction.

Figure 1.2 compares the cost of each algorithm in terms of simulation runs as \( m \) increases. The comparison implicitly assumes that evaluating the gradient is about as expensive as running the simulation. The cost \( \alpha m \) of the Algorithm 1.3, denoted “Linear” in the legend, is a lower bound for Algorithm 1.2. Next we describe how to use the quantities computed by the algorithms to decide if the model is eligible for dimension reduction.
1.2 • Evaluating the dimension of the active subspace

We use the first $k$ eigenvalues $\hat{\lambda}_1, \ldots, \hat{\lambda}_k$ from Algorithm 1.1 to check for an active subspace. Note that we listed only the first $k$ eigenvalues instead of all $m$. The number $M = \alpha k \log(m)$ gives enough samples of the gradient to ensure that the first $k$ eigenvalues are sufficiently accurate; see Chapter 3. Plot these eigenvalues—preferably on a log scale—and look for gaps. A gap in the eigenvalues indicates a separation between active and inactive subspaces. Additionally, the computed eigenvectors are more accurate when there is a gap in the eigenvalues. Choose the dimension $n$ of the active subspace to be the number of eigenvalues preceding the gap. For example, if $k = 6$ and there is a gap between eigenvalues 3 and 4, then choose $n = 3$.

If there is no gap in the eigenvalues, then you do not have an active subspace up to dimension $k$. The map between your simulation’s inputs and quantity of interest varies significantly along all of the directions that the first $k$ eigenvectors represent. You might repeat the process with a larger $k$ to look for gaps in later eigenvalues. Or you might increase $\alpha$ and sample more gradients to increase the accuracy of the eigenvalues. However, the slow convergence of the random sampling means that additional samples may not be of much help. In either case, if you decide to return for more samples of the gradient, you can reuse the ones you already computed since they are all independent samples.

Looking for gaps differs from other methods that use eigenvalues to produce a low-dimensional representation, such as the proper orthogonal decomposition for model reduction in dynamical systems or principal component analysis for high-dimensional data sets. For these analyses, one typically chooses the dimension of the low-dimensional approximation based on the magnitude of the eigenvalues—e.g., so that the sum of the retained eigenvalues exceeds some proportion of the sum of all the eigenvalues. We are less concerned with such energy-like criteria and more concerned with accurately approximating the subspace. The accuracy of the computed subspace depends heavily on the gap between associated eigenvalues; see Chapter 3 for the supporting analysis.

Gaps in the eigenvalues from Algorithm 1.1 with finite differences or Algorithm 1.2 with local linear models can similarly identify an active subspace. However, it is possible
1.3 Sufficient summary plots

Sufficient summary plots were developed by Cook in the context of regression graphics [38]. They are powerful visualization tools for identifying low-dimensional structure in a quantity that depends on several input variables. At first glance, they are scatter plots. But closer inspection reveals great innovation in the horizontal axis. We introduce these plots using the function \( f(x_1, x_2) = \exp(0.7x_1 + 0.3x_2) \) shown in Figure 1.1.

Most graphical software packages allow one to produce a surface plot of a function that depends on two inputs. The surface plots can be dragged and rotated to allow viewing the function from different angles. Imagine first dragging the plot so that the \( x-z \) plane is parallel to the plane of the computer monitor and the \( z \)-axis is aligned straight up and down. Next rotate the plot about the \( z \)-axis, which produces views of the surface from varying angles in the \( x-y \) plane. If one can find a view such that the surface collapses to a univariate plot, then the function \( f \) is constant along the direction into and out of the monitor. Such a collapse suggests that the two input variables can be condensed into a single variable, and \( f \) can be treated like a function of one variable instead of two. Voilà! Dimension reduction!

Figure 1.3(a) shows such a rotated view of \( \exp(0.7x_1 + 0.3x_2) \). This is equivalent to a plot of \( \exp(t) \) against the variable \( t = 0.7x_1 + 0.3x_2 \). In other words, it is a plot of the quantity of interest against a particular linear combination of the input variables.

The sufficient summary plot generalizes the idea of plotting the quantity of interest \( q \) against a linear combination of the input variables \( \eta^T x \), where \( \eta \) is an \( m \)-vector. We denote the sufficient summary plot by \( \text{SSP}(q, \eta^T x) \). If \( \eta \) is a canonical basis vector (a vector of zeros with a one in a single entry), then the sufficient summary plot becomes a simple scatter plot, such as those described in section 1.2.3 of Saltelli et al. [110]. The question is how to choose \( \eta \) such that a univariate trend emerges from the high-dimensional surface—that is, if one is present. Alternatively, can we find \( \eta_1 \) and \( \eta_2 \) to produce a surface of the quantity of interest as a function of the linear combinations \( \eta_1^T x \) and \( \eta_2^T x \)? We denote the latter surface by \( \text{SSP}(q, \eta_1^T x, \eta_2^T x) \).

The first two eigenvectors from \( \hat{W} \) in (1.4) are natural choices for \( \eta_1 \) and \( \eta_2 \) in the sufficient summary plot. (Beware: If the inputs are not normalized to be centered at zero with equal ranges, then the sufficient summary plot may fail to reveal the dimension reduction.) More precisely, the one- and two-dimensional sufficient summary plots using the eigenvectors defining the active subspace are denoted \( \text{SSP}(q, \hat{w}_1^T x) \) and \( \text{SSP}(q, \hat{w}_1^T x, \hat{w}_2^T x) \), respectively. If a large gap exists between the first and second eigenvalues, then there is a good chance that a strong univariate trend is present in \( \text{SSP}(q, \hat{w}_1^T x) \); similarly, a gap between the second and third eigenvalues suggests a strong bivariate trend. It is possible
Chapter 1. Quick Start

0.5 1
0
1.5
2
2.5
x
2
x
1
exp(0.7x
1
+ 0.3x
2
)
(a) Rotated surface plot.

η
T
x
j
f
j
(b) Sufficient summary plot.

Figure 1.3. (b) is the sufficient summary plot using the direction \( \hat{w} \) produced by Algorithm 1.3 with the map \( f(x_1, x_2) = \exp(0.7x_1 + 0.3x_2) \). (a) is a rotated surface plot of the same function.

Roughly speaking, if there are high-frequency oscillations in the function and if the gradients resolve the oscillations, then the eigenvectors from Algorithm 1.1 may choose directions related to the oscillations instead of larger trends. The corresponding sufficient summary plot may not reveal reduced dimensionality, even if it is present with another choice of \( \eta \). In this case, it may be better to choose the direction \( \hat{w} \) from Algorithm 1.3 or the eigenvectors from Algorithm 1.2 (with \( p > m + 1 \)) than the eigenvectors from Algorithm 1.1, which uses the gradients. Since the linear models are fit with least-squares, they are naturally robust to noise; their surfaces are smooth approximations to the true function.

We presented the linear models as workarounds for when gradients are not available. However, if the quantity of interest is noisy—say, due to a finite number of fixed point iterations in the simulation’s solver—then the least-squares-fit linear models provide complementary choices for \( \eta_1 \) and \( \eta_2 \). We can mimic such behavior with the function \( \sin(4\pi x_1) + 4x_2 \), where the sine term creates oscillations along \( x_1 \). Figure 1.4 shows the sufficient summary plots using Algorithms 1.1 and 1.3 with 208 samples of the gradient and the function, respectively. The sufficient summary plot using the direction (close to \( [0,1] \)) produced by the linear model appears to identify a more useful dimension reduction than the one using the first eigenvector (close to \( [1,0] \)) from the gradient-based algorithm.

There are cases when the directions computed by all algorithm variants coincide. The function \( \exp(0.7x_1 + 0.3x_2) \) is such a case. Figure 1.3(b) shows the sufficient summary plot for this function using the direction \( \eta = \hat{w} \) computed with the linear model in Algorithm 1.3. This plot would be essentially the same if we had used the first eigenvector from the gradient-based approach (Algorithm 1.1) or the local linear model approach (Algorithm 1.2).

The global linear model has one pitfall worth mentioning. If the function is not monotonic—in other words, if the sign of some partial derivative changes over the input parameter space—then the linear model may fail to identify the dimension reduction space. We demonstrate this with the function \( \frac{1}{2}(0.7x_1 + 0.3x_2)^2 \). Figure 1.5 compares the
1.3. Sufficient summary plots

(a) Sufficient summary plot, Alg. 1.1.
(b) Sufficient summary plot, Alg. 1.3.

Figure 1.4. (a) is the sufficient summary plot using the first eigenvector produced by Algorithm 1.1 for the function \( f(x_1, x_2) = \sin(4\pi x_1) + 4x_2 \). (b) is the sufficient summary plot using the direction \( \hat{w} \) computed with Algorithm 1.3. The directions produced by the least-squares linear models are robust to high-frequency oscillations.

sufficient summary plots using the gradient-based eigenvector and the normalized gradient of the linear model; the plots use 208 samples of the gradient and the function, respectively. We have not encountered such cases in practice. We think this is because many—if not most—engineering quantities of interest are monotonic with respect to the inputs. Intuition-driven statements from engineers, such as more of input \( X \) yields more/less of output \( Q \) within the regime of interest, support this claim. In such cases, the linear model can often quickly and cheaply identify a one-dimensional subspace that is sufficient for describing the behavior of the quantity of interest. We present the results of Algorithm 1.3 applied to a simulation of a hypersonic scramjet in Chapter 5.

(a) Sufficient summary plot, Alg. 1.1.
(b) Sufficient summary plot, Alg. 1.3.

Figure 1.5. (a) is the sufficient summary plot using the first eigenvector produced by Algorithm 1.1 for the function \( f(x_1, x_2) = \frac{1}{2}(0.7x_1 + 0.3x_2)^2 \). (b) is the sufficient summary plot using the direction \( \hat{w} \) computed with Algorithm 1.3. The linear model fails to identify the dimension reduction space when the function is not monotonic with respect to its inputs.
1.4 • An example with a parameterized PDE

We compare these approaches on Poisson’s equation in two spatial dimensions with constant forcing and spatially varying coefficients parameterized by 100 Gaussian random variables. Similar models appear in single-phase subsurface flow, where the parameterized coefficients model uncertainty in subsurface permeability. More details of this specific test problem are given in section 3.4.2. The point of this example is to show the algorithms’ results on something closer to an engineering application than simple test functions; see Chapter 5 for real engineering models.

We begin with the following three required pieces taken from the beginning of this chapter:

- The smooth scalar quantity of interest $q$ is the average of the PDE solution on the right boundary.
- The $m = 100$ parameters that characterize the PDE coefficients are unbounded, but the model is well-defined for all parameter values.
- The simulation uses a finite element method to estimate the solution and the quantity of interest given values for the parameters. The computing platform is MATLAB 2014a on a MacBook Air with a dual-core processor and 8 GB of RAM.

We do not have a budget per se. The finite element simulation runs fast enough to permit us to run all algorithm variants. The sampling density $\rho$ is a standard Gaussian density on $\mathbb{R}^{100}$; it is already normalized. We choose $k = 6$ for the largest dimension of interest and $\alpha = 3$ for an oversampling factor. Gradients are available via an adjoint solver.

We apply Algorithm 1.1 using the gradient computed with the adjoint solution. We use $M = 83$ evaluations of the gradient. Figure 1.6 shows the first $k = 6$ eigenvalues and the first eigenvector’s components. Note the large gap between the first and second eigenvalues. Figure 1.6 also shows the eigenvalues and first eigenvector’s components from Algorithm 1.2 with $N = 2000$ and $p = 180$. The eigenvalues have a similar gap between
the first and the second, and the first eigenvector’s components match the ones computed using gradients in Algorithm 1.1. We apply Algorithm 1.3 with $N = 300$; the components of its computed vector $\hat{\mathbf{w}}$ match the first eigenvector’s components from Algorithms 1.1 and 1.2.

Figure 1.7 displays the sufficient summary plots from each of the three algorithm variants. All sufficient summary plots show a strong univariate trend in the function of 100 parameters, which is consistent with the eigenvalue analysis. Figure 1.7(a) plots the 83 evaluations $q_i$ against $\hat{\mathbf{w}}^T \mathbf{x}_i$ from the gradient-based algorithm. Figure 1.7(b) shows 100 of the $N = 2000$ samples used in Algorithm 1.2. Figure 1.7(c) shows 100 of the 300 samples used to fit the global linear model in Algorithm 1.3.

![Figure 1.7](image)

**Figure 1.7.** (a) Sufficient summary plots from Algorithm 1.1 using 83 gradient samples. (b) Algorithm 1.2 using $N = 2000$ initial samples and $p = 180$. (c) Algorithm 1.3 using 300 samples. All plots indicate that a univariate model is a reasonable choice for this function. Image (a) is modified from [28].

For this PDE model, the results from all algorithms are consistent, but this is not the case every time. You should choose the algorithm that is most appropriate for your particular application. The primary factors that influence this choice are (i) how many parameters your model has, (ii) how many times you can run the simulation, (iii) whether or not you have gradients, and (iv) whether or not there is noise in your quantity of interest.
Chapter 5 presents several real applications where we discuss how these considerations affect our choice of algorithm to search for dimension reduction.

One final note: The results of any random sampling algorithm change based on the particular set of samples that are drawn. In Chapter 3 we develop a bootstrap procedure for studying the variability in the estimated eigenvalues and active subspace. We can then include bootstrap intervals in the plots. The bootstrap is the most appropriate choice for studying variability because it is easy to implement and does not require further calls to the simulation code.
Bibliography


