SIAM J. SCI. COMPUT.
Vol. 41, No. 5, pp. S106–S128
© 2019 Society for Industrial and Applied Mathematics

Downloaded 11/18/19 to 138.195.214.230. Redistribution subject to SIAM license or copyright; see http://www.siam.org/journals/ojsa.php

GAUSSIAN QUADRATURE AND POLYNOMIAL APPROXIMATION FOR ONE-DIMENSIONAL RIDGE FUNCTIONS*

ANDREW GLAWS† AND PAUL G. CONSTANTINE†

Abstract. Many of the input-parameter-to-output-quantity-of-interest maps that arise in computational science admit a surprising low-dimensional structure, where the outputs vary primarily along a handful of directions in the high-dimensional input space. This type of structure is well modeled by a ridge function, which is a composition of a low-dimensional linear transformation with a nonlinear function. If the goal is to compute statistics of the output—e.g., as in uncertainty quantification or robust design—then one should exploit this low-dimensional structure, when present, to accelerate computations. We develop Gaussian quadrature and the associated polynomial approximation for one-dimensional ridge functions. The key elements of our method are (i) approximating the univariate density of the given linear combination of inputs by repeated convolutions and (ii) a Lanczos–Stieltjes method for constructing orthogonal polynomials and Gaussian quadrature.

Key words. Gaussian quadrature, orthogonal polynomials, ridge functions

AMS subject classifications. 42C05, 33C50, 41A55

DOI. 10.1137/18M1194894

1. Numerical methods for high-dimensional integration.

High-dimensional integration is a common problem in scientific computing arising from, for example, the need to estimate expectations in uncertainty quantification [62, 64] and robust design [1], where physics-based simulation models contain parametric uncertainty. However, integration suffers from the curse of dimensionality [65, 23]; loosely, the amount of information (e.g., integrand evaluations) needed to estimate the integral to within a fixed tolerance grows exponentially with dimension (i.e., the number of independent variables affecting the integrand). Monte Carlo [58] is a popular method for high-dimensional integration, since its $\mathcal{O}(N^{-1/2})$ convergence rate is independent of dimension. However, this rate is also very slow, which precludes the possibility of high accuracy; to obtain k accurate digits, one must sample the integrand $\mathcal{O}(10^{2k})$ times. There are many extensions to simple Monte Carlo that produce relatively higher accuracy (i.e., *variance reduction*) including stratified sampling, control variates, and multilevel methods [30]. Quasi-Monte Carlo methods [6] have a superior dimension-independent convergence rate of $\mathcal{O}(N^{-1})$ for certain classes of functions, though high accuracy is still elusive. Interpolatory sparse grid integration [5] converges rapidly—comparable to one-dimensional interpolatory integration rules [66]—for smooth functions with small mixed partial derivatives, which makes high accuracy possible. Common sparse grid rules contain a *level* parameter where the nodes of the level $\ell - 1$ sparse grid are a subset of the level ℓ sparse grid nodes, which enables practical numerical convergence studies; such rules are called *nested*. Although sparse grids are theoretically optimal in a precise sense [57], the number of nodes needed to advance from level $\ell-1$ to ℓ is still too large for many scientific computing applications with expensive integrands.

*Received by the editors June 18, 2018; accepted for publication (in revised form) March 28, 2019; published electronically October 29, 2019.
https://doi.org/10.1137/18M1194894
Funding: This work was supported by the DARPA DSO Enabling Quantification of Uncertainty in Physical Systems program.
†Department of Computer Science, University of Colorado Boulder, CO (andrew.glaws@colorado.edu, paul.constantine@colorado.edu).

Copyright © by SIAM. Unauthorized reproduction of this article is prohibited.

Downloaded 11/18/19 to 138.195.214.230. Redistribution subject to SIAM license or copyright; see http://www.siam.org/journals/ojsa.php

An alternative strategy is to approximate the integrand with an easy-to-integrate approximation, such as a multivariate polynomial [43]. For any polynomial expressed in an orthogonal (with respect to the integration measure) polynomial basis, the integral of the polynomial is the coefficient associated with the constant term; this fact has contributed to the popularity of so-called *polynomial chaos* methods in uncertainty quantification [48]. From a Monte Carlo perspective, an approximation-based approach is comparable to control variates [58]. The integration error is tied to the integrand approximation error. Unfortunately, polynomial and related approximations are also dimensionally cursed; approximation-based approaches effectively trade one intractable high-dimensional problem for another. But the general idea of identifying exploitable structure in the integrand remains appealing and motivates our approach.

1.1. Ridge structure in scientific computing applications. Many functions that map input parameters to an output quantity of interest found in scientific computing applications admit a surprising one-dimensional structure we call *ridge structure*. A *ridge function* [59] is a function $f : \mathbb{R}^m \to \mathbb{R}$ of the form

$$f(\mathbf{x}) \;=\; g(\mathbf{a}^\top \mathbf{x}), \tag{1.1}$$

where $\mathbf{a} \in \mathbb{R}^m$ is a constant vector called the *ridge direction* and $g : \mathbb{R} \to \mathbb{R}$ is the *ridge profile* [56]. Without loss of generality, we can assume $\mathbf{a}$ has unit 2-norm. Ridge functions are constant along directions in the domain that are orthogonal to $\mathbf{a}$; for $\mathbf{u}$ such that $\mathbf{u}^\top \mathbf{a} = 0$,

$$f(\mathbf{x}+\mathbf{u}) \;=\; g\left(\mathbf{a}^\top(\mathbf{x}+\mathbf{u})\right) \;=\; g(\mathbf{a}^\top \mathbf{x}) \;=\; f(\mathbf{x}). \tag{1.2}$$

Despite their name, ridge functions are not related to *ridge regression* [38, Chapter 3.4], which is a linear approximation scheme with 2-norm regularization on the coefficients. In contrast, ridge functions are basic objects in *projection pursuit regression* [26], which approximates a function with a sum of ridge functions. There is a lot of literature on approximation by sums of ridge functions (see, e.g., Chapters 22–24 of Cheney and Light [7]), partly because one-layer feed-forward neural networks can be written as sums of ridge functions, where the ridge profile is commonly called the *activation function* [39].

There are several methods for assessing whether a given function has ridge structure; these methods start by estimating a ridge direction. The gradient of a differentiable ridge function points in the ridge direction, since $\nabla f(\mathbf{x}) = \mathbf{a}\, g'(\mathbf{a}^\top \mathbf{x})$, where g' is the derivative of the profile. If an application code includes algorithmic differentiation [37] capabilities for sensitivity analysis or optimization, then a single simulation may reveal the ridge direction in a model's input/output map. When gradients are not available, one can use a set of function evaluations or point queries to estimate the ridge direction. For a well-behaved simulation, finite difference approximations of the partial derivatives may suffice to estimate a gradient and hence the ridge direction. Cohen et al. [8] analyze a greedy procedure for estimating the ridge direction $\mathbf{a}$ from point queries when all elements of $\mathbf{a}$ are nonnegative and $\mathbf{a}$ is compressible. Fornasier, Schnass, and Vybiral [25] extend the analysis to the case where $\mathbf{a}$ in (1.1) is a tall matrix with compressible columns; their method uses finite difference approximations of random directional derivatives. Tyagi and Cevher [70] relax the compressibility assumption. These papers contain theoretical recovery guarantees and complexity estimates for their algorithms.

In computational science, Constantine, Dow, and Wang [11] study the matrix $\boldsymbol{C} = \mathbb{E}[\nabla f \, \nabla f^\top]$, whose dominant eigenspaces are *active subspaces*, and noted that $\boldsymbol{C}$

Copyright © by SIAM. Unauthorized reproduction of this article is prohibited.

Downloaded 11/18/19 to 138.195.214.230. Redistribution subject to SIAM license or copyright; see http://www.siam.org/journals/ojsa.php

is rank-one if and only if f is a continuously differentiable ridge function [12]. Thus, when $\boldsymbol{C}$ is rank-one, its first eigenvector is the ridge direction. In practice, if $\boldsymbol{C}$ is nearly rank-one, then the associated f may be well-approximated by a ridge function whose ridge direction is the first eigenvector of $\boldsymbol{C}$. Numerical methods for estimating $\boldsymbol{C}$ may use gradient capabilities when present [16, 41], and several approximations have been developed when gradients are not available [49, 13, 24]. Another approach for estimating a ridge direction involves fitting a ridge function model with a set of data comprised of point queries of the computational model's input/output map [12, 40]. Related techniques use a Bayesian framework to derive posterior distributions on the ridge direction conditioned on data, and these approaches have been used in the related statistical subfield of *computer experiments* [35, 53, 68].

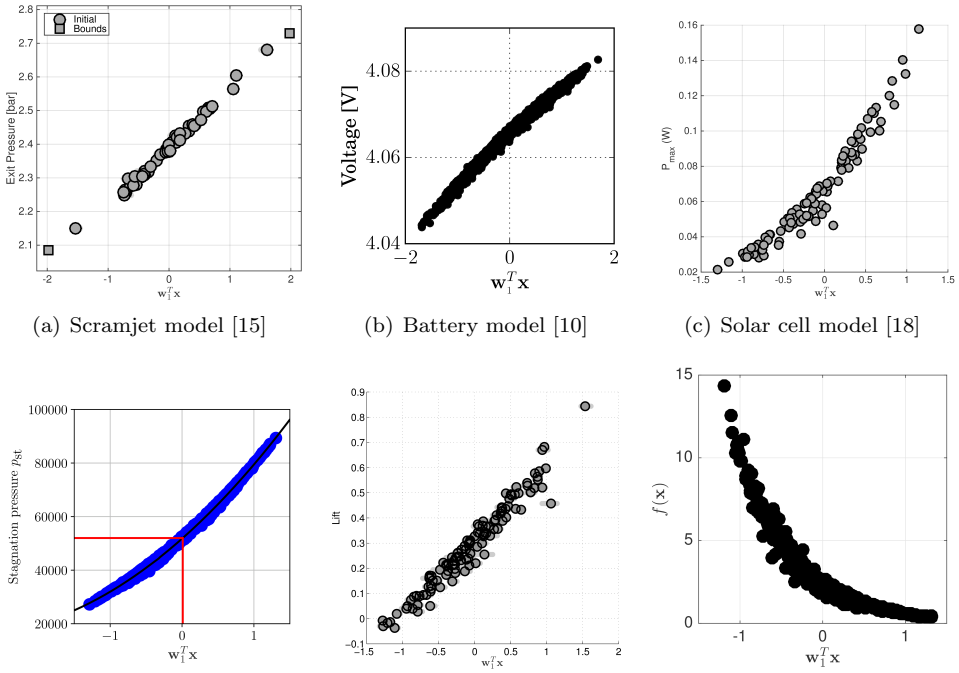
In statistical regression, techniques for *sufficient dimension reduction* (SDR) [50] use given predictor/response pairs to find low-dimensional subspaces of the predictor (i.e., *input*) space that are statistically sufficient to characterize the response (i.e., output). These techniques have been applied to computational science models for sensitivity analysis and dimension reduction [19, 51], where the predictor/response pairs are point queries from the computational model and thus do not require gradients. Glaws, Constantine, and Cook showed how the associated inverse regression methods from SDR should be interpreted as estimating ridge directions in deterministic models [31].

No matter how one estimates a ridge direction $\mathbf{a}$, one can assess the viability of the one-dimensional ridge function model by plotting $f(\mathbf{x})$ versus $\mathbf{a}^\top\mathbf{x}$ using a set of point queries. If the plot reveals (near) functional structure—i.e., that the function f appears to be well represented by a univariate function of $\mathbf{a}^\top\mathbf{x}$—then the particular parameter-to-quantity-of-interest map has ridge structure. In statistical regression, such plots are called *sufficient summary plots* [20]. To avoid confusion with the precise notion of statistical sufficiency, which is not valid in the case where the data are derived from a deterministic function, we prefer to call these plots *shadow plots*, since the name invokes the analogy of the surface's shadow along all but one direction.

Several of these exploratory approaches for estimating ridge directions and generating shadow plots have revealed one-dimensional ridge structure in a range of computational science applications, including wind farm modeling [46], aerospace design [55, 36, 3, 42], hypersonic vehicle modeling [15], turbine manufacturing [61, 2], hydrologic models [45, 29], and energy models [32, 10, 18]. Figure 1 shows shadow plots from several applications demonstrating the near one-dimensional ridge structure. To the best of our knowledge, there is neither a mathematical proof nor a first principles physical argument for why such structure is so prevalent across computational science applications. Nevertheless, the exploratory perspective continues to reveal such exploitable structure in real applications.

1.2. Exploiting ridge structure for numerical integration. The prevalence of one-dimensional ridge structure in applications justifies the question, if a function admits discoverable one-dimensional ridge structure, how can one exploit that structure to efficiently estimate the integral (e.g., the average) of the model's output quantity of interest? In other words, assume that a computational scientist can use one of the methods mentioned above to determine that a given model's input/output map is a ridge function—or is well-approximated by a ridge function—and the method estimates the ridge direction $\mathbf{a}$; how can the scientist exploit this information to efficiently estimate the map's integral?

Copyright © by SIAM. Unauthorized reproduction of this article is prohibited.



(a) Scramjet model [15] (b) Battery model [10] (c) Solar cell model [18]

(d) Reentry vehicle model [21] (e) Transonic airfoil model [9] (f) MHD generator model [32]

FIG. 1. *One-dimensional shadow plots of data derived from computational models across applications. The plots verify the near-one-dimensional structure in the input/output map.*

Such an integration requires approximation of the density induced by the linear combination $\mathbf{a}^\top\mathbf{x}$. We compute this density efficiently using repeated convolutions. This approach is valid provided that the components of $\mathbf{x}$ are independent, as is common in computational science applications. We then introduce a procedure for computing a generalized Gaussian quadrature rule for computing univariate integrals with respect to a known ridge direction in a high-dimensional space. The one-dimensional representation enables high accuracy with relatively few function evaluations. Since Gaussian quadrature is intimately tied to polynomial approximation, we obtain a univariate polynomial approximation of $f(\mathbf{x})$ using a basis of polynomials that are orthogonal with respect to the density of $\mathbf{a}^\top\mathbf{x}$; this approximation can be used as a surrogate or response surface approximation of f. A related approach by Tsilifis uses the inverse empirical distribution to derive a polynomial approximation [69]. However, this approach suffers from the relatively low accuracy of the empirical distribution; our convolution-based approach has no such limitation. Additionally, low-dimensional Gaussian processes can be used with reduced-variance Monte Carlo integration [35], but this approach does not exhibit the same spectral convergence that is possible using the proposed Gaussian quadrature method.

The remainder of this paper proceeds as follows. Section 2 contains important background information, including discussions on ridge functions, Fourier expansions in terms of orthogonal polynomials, and the Lanczos and Stieltjes methods. In section 3, we introduce the new Lanczos–Stieltjes method for one-dimensional ridge approximation and integration. Section 4 provides a heuristic extension of the method for approximate one-dimensional ridge functions, such as those shown in Figure 1.

Copyright © by SIAM. Unauthorized reproduction of this article is prohibited.

Downloaded 11/18/19 to 138.195.214.230. Redistribution subject to SIAM license or copyright; see http://www.siam.org/journals/ojsa.php

Downloaded 11/18/19 to 138.195.214.230. Redistribution subject to SIAM license or copyright; see http://www.siam.org/journals/ojsa.php

2. Background. We represent the map from a computer model's physical inputs[1] to its output quantity of interest as a scalar-valued function of m independent variables,

$$y \;=\; f(\mathbf{x}), \qquad y \in \mathbb{R}, \quad \mathbf{x} \in \mathbb{R}^m, \tag{2.1}$$

and we assume the input space is weighted by a given probability density function $p(\mathbf{x})$, which describes uncertainty in the model inputs. For simplicity, we assume this density is uniform over the m-dimensional hypercube $[-1,1]^m$ such that

$$p(\mathbf{x}) \;=\; \begin{cases} \frac{1}{2^m} & \text{if } ||\mathbf{x}||_\infty \leq 1, \\ 0 & \text{otherwise.} \end{cases} \tag{2.2}$$

The independence is the important part; our approach extends easily to any product-type density function.

2.1. Polynomial approximation. Gaussian quadrature is intimately tied to approximation with orthogonal polynomials; for completeness, we provide a brief background. Assume that f is square-integrable with respect to the input density $p(\mathbf{x})$. Then, f admits a Fourier expansion in orthogonal polynomials with respect to p,

$$y \;=\; f(\mathbf{x}) \;=\; \sum_{|\boldsymbol{\alpha}|=0}^{\infty} f_{\boldsymbol{\alpha}}\, \psi_{\boldsymbol{\alpha}}(\mathbf{x}), \tag{2.3}$$

where equality is denoted in the L^2 sense [27]. The multivariate orthogonal polynomials $\psi_{\boldsymbol{\alpha}}(\mathbf{x})$ are indexed by the multi-index $\boldsymbol{\alpha} \in \mathbb{N}_0^m$ which denotes the degree of the polynomial with respect to each of the components of $\mathbf{x}$. Since $p(\mathbf{x})$ is a product-type uniform density, the multivariate polynomials are

$$\psi_{\boldsymbol{\alpha}}(\mathbf{x}) \;=\; \prod_{i=1}^{m} \psi_{\alpha_i}(x_i), \tag{2.4}$$

where each ψ_{α_i} is the univariate Legendre polynomial of degree α_i. Without loss of generality, we also assume that the $\psi_{\boldsymbol{\alpha}}$ are normalized so that the coefficients in (2.3) are the inner product of f with the appropriate polynomial,

$$f_{\boldsymbol{\alpha}} \;=\; \int f(\mathbf{x})\, \psi_{\boldsymbol{\alpha}}(\mathbf{x})\, p(\mathbf{x})\, \mathrm{d}\mathbf{x}. \tag{2.5}$$

This method of approximation by orthogonal polynomials also appears in the uncertainty quantification literature under the name polynomial chaos [28, 71].

In practice, we compute the *pseudospectral expansion* [14] of f by truncating (2.3) to include only polynomials of total degree d or less and approximating the integral in (2.5) numerically:

$$y \;=\; f(\mathbf{x}) \;\approx\; \sum_{|\boldsymbol{\alpha}|\leq d} \hat{f}_{\boldsymbol{\alpha}}\, \psi_{\boldsymbol{\alpha}}(\mathbf{x}), \quad \text{where} \quad f_{\boldsymbol{\alpha}} \;\approx\; \hat{f}_{\boldsymbol{\alpha}} \;=\; \sum_{j=0}^{M-1} \omega_j\, f(\boldsymbol{\xi}_j)\, \psi_{\boldsymbol{\alpha}}(\boldsymbol{\xi}_j), \tag{2.6}$$

[1] We do not consider numerical parameters such as grid spacing or solver tolerances.

Copyright © by SIAM. Unauthorized reproduction of this article is prohibited.

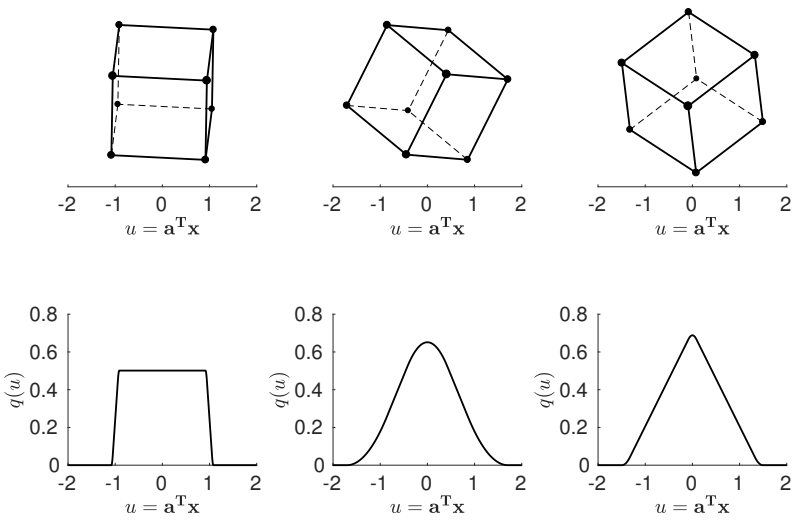


FIG. 2. *Different density functions $q(u)$ induced by different vectors $\mathbf{a} \in \mathbb{R}^3 \setminus \{\mathbf{0}\}$.*

where $(\boldsymbol{\xi}_j, \omega_j)$, $j = 0, \ldots, M-1$, denote the nodes and weights of an M-point numerical integration rule (e.g., tensor product Gauss quadrature) with respect to p.

Pseudospectral polynomial approximations serve as quick-to-evaluate surrogates for the original function. However, as the input dimension m grows, the cost of constructing (2.6) quickly increases—a total degree d polynomial in m dimensions has $\binom{m+d}{d}$ coefficients.

2.2. Polynomial approximation for ridge functions. To construct an orthogonal polynomial expansion of $g(\mathbf{a}^\top \mathbf{x})$ similar to (2.3), we must first understand the transformed input space. Let $u = \mathbf{a}^\top \mathbf{x}$ denote the scalar-valued input of the ridge profile. The linear transform $\mathbf{a}^\top \mathbf{x}$ induces a new density, which we denote $q(u)$. Figure 2 shows different rotations and projections, defined by different vectors $\mathbf{a}$, of the three-dimensional cube $[-1, 1]^3$ and the resulting one-dimensional probability densities $q(u)$.

We address the computation of $q(u)$ in section 3. For now, assume $q(u)$ is known. We write the polynomial expansion of $g(u)$ as

$$y \;=\; g(u) \;=\; \sum_{i=0}^{\infty} g_i \, \phi_i(u), \quad \text{where} \quad g_i \;=\; \int g(u)\, \phi_i(u)\, q(u)\, \mathrm{d}u, \tag{2.7}$$

where the univariate polynomials ϕ_i are orthonormal with respect to q. By truncating the expansion and numerically approximating the coefficients, we obtain the pseudospectral approximation of the ridge profile,

$$y \;=\; g(u) \;\approx\; \sum_{i=0}^{d} \hat{g}_i \, \phi_i(u), \quad \text{where} \quad \hat{g}_i \;=\; \sum_{j=0}^{M-1} \nu_j \, g(\lambda_j)\, \phi_i(\lambda_j), \tag{2.8}$$

where $(\lambda_j \,, \nu_j)$ define a numerical integration rule with respect to q.

Constructing the pseudospectral polynomial expansion of the one-dimensional ridge profile $g(u)$ significantly reduces the number of function evaluations required compared to working in the m-dimensional space. Figure 3 shows an example of

Downloaded 11/18/19 to 138.195.214.230. Redistribution subject to SIAM license or copyright; see http://www.siam.org/journals/ojsa.php

Copyright © by SIAM. Unauthorized reproduction of this article is prohibited.

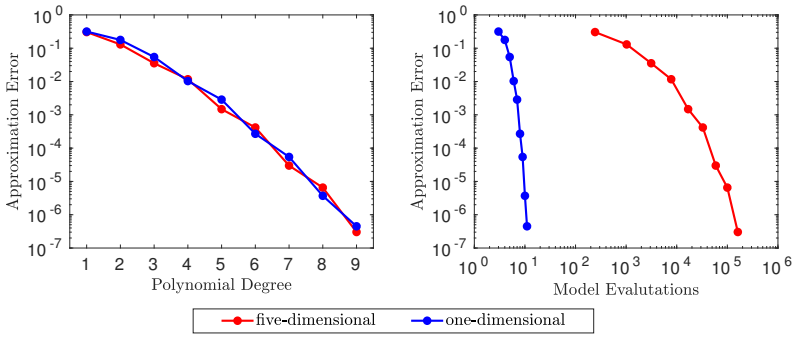


FIG. 3. *A comparison of the cost and accuracy of the five-dimensional (in red) and the one-dimensional (in blue) polynomial approximations for a given function. They perform equally well in terms of polynomial degree, but the one-dimensional approximation uses four orders of magnitude fewer function evaluations.*

approximating a smooth ridge function of five variables using either a multivariate polynomial in all five variables or a univariate polynomial along the ridge direction; unsurprisingly, the univariate approximation uses four orders of magnitude fewer function evaluations. However, (2.8) requires knowledge of the orthonormal polynomials ϕ_i and an integration rule $(\lambda_j\,,\,\nu_j)$ with respect to q, as well as the ability to evaluate the ridge profile, which may not be known in closed form, at the λ_j's. We address this latter issue in section 3. In the next section, we discuss how to obtain the orthonormal polynomials and integration nodes/weights using the Stieltjes and Lanczos iterative methods.

We next review the Stieltjes and Lanczos iterative methods. We discuss the key components of each and show that, under certain conditions, the Lanczos method is a discrete approximation to the Stieltjes procedure. These methods and the relationships between them have been rigorously studied [52, 27, 33]; the discussion in the following section is based on these references.

2.3. The Stieltjes procedure. The Stieltjes procedure—given in Algorithm 2.1—is a method for iteratively constructing a sequence of orthonormal polynomials $\{\phi_0(u), \phi_1(u), \phi_2(u), \dots\}$ with respect to a given density [63]. Step (iv) in Algorithm 2.1 contains the three-term recurrence relationship that must be satisfied by any sequence of orthonormal polynomials.

Algorithm 2.1 Stieltjes procedure [27, section 2.2.3.1].

Given: probability density function $q(u)$
Assumptions: $\phi_{-1}(u) = 0$ and $\tilde{\phi}_0(u) = 1$
For $i = 0, 1, 2, \dots,$

(i) $\beta_i = \int \tilde{\phi}_i(u)^2 \, q(u) \, \mathrm{d}u,$

(ii) $\phi_i(u) = \tilde{\phi}_i(u) \,/\, \beta_i,$

(iii) $\alpha_i = \int u \, \phi_i(u)^2 \, q(u) \, \mathrm{d}u,$

(iv) $\tilde{\phi}_{i+1}(u) = (u - \alpha_i) \, \phi_i(u) - \beta_i \, \phi_{i-1}(u).$

Output: the orthonormal polynomials $\{\phi_0(u), \phi_1(u), \phi_2(u), \dots\}$ and recurrence coefficients α_i, β_i for $i = 0, 1, 2, \dots$

Copyright © by SIAM. Unauthorized reproduction of this article is prohibited.

Downloaded 11/18/19 to 138.195.214.230. Redistribution subject to SIAM license or copyright; see http://www.siam.org/journals/ojsa.php

Downloaded 11/18/19 to 138.195.214.230. Redistribution subject to SIAM license or copyright; see http://www.siam.org/journals/ojsa.php

By stopping the Stieltjes procedure after $d+1$ terms, we can rewrite the three-term recurrence relationship in vector form as

$$u\,\boldsymbol{\phi}(u) \;=\; \boldsymbol{J}\,\boldsymbol{\phi}(u) + \beta_{d+1}\,\phi_{d+1}(u)\mathbf{e}_{d+1}, \tag{2.9}$$

where $\boldsymbol{\phi}(u) = [\,\phi_0(u)\,,\,\phi_1(u)\,,\,\ldots\,,\,\phi_d(u)\,]^\top$, $\mathbf{e}_{d+1} \in \mathbb{R}^{d+1}$ is a vector of zeros with a one in the last entry, and the matrix $\boldsymbol{J} \in \mathbb{R}^{(d+1)\times(d+1)}$—referred to as the *Jacobi matrix*—is a symmetric, tridiagonal matrix of recurrence coefficients,

$$\boldsymbol{J} \;=\; \begin{bmatrix} \alpha_0 & \beta_1 & & & \\ \beta_1 & \alpha_1 & \beta_2 & & \\ & \ddots & \ddots & \ddots & \\ & & \beta_{d-1} & \alpha_{d-1} & \beta_d \\ & & & \beta_d & \alpha_d \end{bmatrix}. \tag{2.10}$$

Let $\boldsymbol{J} \;=\; \boldsymbol{Q}\,\boldsymbol{\Lambda}\,\boldsymbol{Q}^\top$ denote the eigendecomposition of $\boldsymbol{J}$. From (2.9), the $d+1$ eigenvalues of $\boldsymbol{J}$ are the zeros of the $(d+1)$-degree orthonormal polynomial ϕ_{d+1}. The normalized eigenvector associated with the eigenvalue λ_j has the form

$$(\boldsymbol{Q})_j \;=\; \frac{\boldsymbol{\phi}(\lambda_j)}{\sqrt{\boldsymbol{\phi}(\lambda_j)^\top \boldsymbol{\phi}(\lambda_j)}}, \tag{2.11}$$

where $(\cdot)_j$ denotes the jth column of the given matrix.

2.3.1. Gaussian quadrature from the Stieltjes procedure. The $(d+1)$-point Gaussian quadrature rule with respect to q can be obtained from the eigendecomposition of $\boldsymbol{J}$ from (2.10) [34]. The quadrature nodes are the eigenvalues of $\boldsymbol{J}$ and the associated quadrature weights are the square of the first entry of the associated normalized eigenvector,

$$\nu_j \;=\; (\boldsymbol{Q})_{0,j}^2 \;=\; \frac{1}{\boldsymbol{\phi}(\lambda_j)^\top \boldsymbol{\phi}(\lambda_j)}. \tag{2.12}$$

2.4. The Lanczos procedure. The Lanczos algorithm was introduced as an iterative approach to estimating eigenvalues of large symmetric matrices [47]. Given symmetric $\boldsymbol{A} \in \mathbb{R}^{N\times N}$, the Lanczos algorithm constructs the system

$$\boldsymbol{A}\,\boldsymbol{V} = \boldsymbol{V}\,\boldsymbol{T} + \beta_{d+1}\,\mathbf{v}_{d+1}^\top\,\mathbf{e}_{d+1}, \tag{2.13}$$

where $\boldsymbol{T} \in \mathbb{R}^{(d+1)\times(d+1)}$ is a symmetric, tridiagonal matrix of recurrence coefficients, $\boldsymbol{V} \in \mathbb{R}^{N\times(d+1)}$ contains the Lanczos vectors, and $\mathbf{e}_{d+1} \in \mathbb{R}^{d+1}$ is a vector of zeros with a one in the last entry. The eigenvalues of $\boldsymbol{T}$ approximate those of $\boldsymbol{A}$, and $\boldsymbol{V}$ transforms the eigenvectors of $\boldsymbol{T}$ into approximate eigenvectors of $\boldsymbol{A}$. We consider conditions under which the Lanczos algorithm serves as a discrete approximation to the Stieltjes procedure. Algorithm 2.2 contains the Lanczos algorithm. For notational convenience, we put the outputs of Algorithm 2.2 into matrices. Define the matrix $\boldsymbol{V}$ of Lanczos vectors as

$$\boldsymbol{V} \;=\; \begin{bmatrix} | & | & & | \\ \mathbf{v}_0 & \mathbf{v}_1 & \ldots & \mathbf{v}_{d-1} \\ | & | & & | \end{bmatrix}, \tag{2.14}$$

Copyright © by SIAM. Unauthorized reproduction of this article is prohibited.

Algorithm 2.2 Lanczos algorithm [27, section 3.1.7.1].

Given: an $N \times N$ symmetric matrix $\boldsymbol{A}$
Assumptions: $\mathbf{v}_{-1} = \mathbf{0} \in \mathbb{R}^N$ and $\tilde{\mathbf{v}}_0 \in \mathbb{R}^N \setminus \{\mathbf{0}\}$
For $i = 0, 1, \ldots, d-1$,
1. $\beta_i = \sqrt{\tilde{\mathbf{v}}_i^\top \tilde{\mathbf{v}}_i}$,
2. $\mathbf{v}_i = \tilde{\mathbf{v}}_i / \beta_i$,
3. $\alpha_i = \mathbf{v}_i^\top \boldsymbol{A}\, \mathbf{v}_i$,
4. $\tilde{\mathbf{v}}_{i+1} = (\boldsymbol{A} - \alpha_i \boldsymbol{I})\, \mathbf{v}_i - \beta_{i-1}\, \mathbf{v}_{i-1}$.

and the symmetric, tridiagonal Jacobi matrix $\boldsymbol{T}$ as

$$\boldsymbol{T} = \begin{bmatrix} \alpha_0 & \beta_1 & & & \\ \beta_1 & \alpha_1 & \beta_2 & & \\ & \ddots & \ddots & \ddots & \\ & & \beta_{d-2} & \alpha_{d-2} & \beta_{d-1} \\ & & & \beta_{d-1} & \alpha_{d-1} \end{bmatrix}. \tag{2.15}$$

2.4.1. Gaussian quadrature from Lanczos. Let (u_j, v_j), $j = 0, \ldots, N-1$, be the nodes and weights of an N-point numerical integration rule with respect to a given density function $q(u)$. These nodes and weights define a discrete approximation of $q(u)$, which we denote by $q^{(N)}(u)$. Performing Algorithm 2.2 on the diagonal matrix

$$\boldsymbol{A} = \begin{bmatrix} u_0 & & \\ & \ddots & \\ & & u_{N-1} \end{bmatrix} \tag{2.16}$$

with starting vector

$$\tilde{\mathbf{v}}_0 = \begin{bmatrix} \sqrt{v_0} \\ \vdots \\ \sqrt{v_{N-1}} \end{bmatrix} \tag{2.17}$$

is equivalent to performing Algorithm 2.1 on the discrete density function $q^{(N)}(u)$ [17]. The recurrence coefficients in $\boldsymbol{T}$ from (2.15) converge to those in $\boldsymbol{J}$ from (2.10) as N goes to infinity [27]. Therefore, we can use these recurrence coefficients to produce approximations to the orthonormal polynomials $\{\phi_0(u), \phi_1(u), \phi_2(u), \ldots\}$. Additionally, the eigendecomposition of $\boldsymbol{T}$ provides us with an approximate Gaussian quadrature rule with respect to $q(u)$.

3. Integration and approximation for ridge functions. Assume that f is a ridge function,

$$y = f(\mathbf{x}) = g(u), \quad \text{where} \quad u = \mathbf{a}^\top \mathbf{x}. \tag{3.1}$$

We want to build a Gaussian quadrature rule and polynomial approximation with respect to the induced density function $q(u)$.

3.1. Computing the density of a linear combination of independent variables. We compute $q(u)$ using a *convolution of probability densities* [4, Chap. 4].

Downloaded 11/18/19 to 138.195.214.230. Redistribution subject to SIAM license or copyright; see http://www.siam.org/journals/ojsa.php

Copyright © by SIAM. Unauthorized reproduction of this article is prohibited.

Downloaded 11/18/19 to 138.195.214.230. Redistribution subject to SIAM license or copyright; see http://www.siam.org/journals/ojsa.php

Consider two independent random variables x_0 and x_1 with density functions p_0 and p_1, respectively. The density function of $u = x_0 + x_1$ is

$$q(u) \;=\; (p_0 * p_1)(u) \;=\; \int p_0(t)\, p_1(u-t)\, \mathrm{d}t. \tag{3.2}$$

Equation (3.2) is the convolution of p_0 and p_1.

Recall from (2.2) that we assume the input space is weighted by a uniform density over the hypercube, $\mathbf{x} \sim \mathcal{U}([-1,1]^m)$. By independence, we have that $p(\mathbf{x}) = p_0(x_0)\, p_1(x_1)\, \ldots\, p_{m-1}(x_{m-1})$ with each

$$p_i(x_i) \;=\; \begin{cases} \frac{1}{2} & \text{if } |x_i| \le 1, \\ 0 & \text{otherwise.} \end{cases} \tag{3.3}$$

We can write the linear transform $u = \mathbf{a}^\top \mathbf{x} = a_0\, x_0 + \cdots + a_{m-1} x_{m-1}$ and recognize that $a_i\, x_i$ is uniformly distributed in the interval $[-a_i, a_i]$. Thus, we can obtain $q(u)$ by iteratively applying convolutions to each $a_i x_i$. In practice, we approximate the integral in (3.2) using a trapezoidal rule since the integrands in (3.2) are not smooth. However, this integral is one dimensional and does not require us to evaluate the computational model ($f(\mathbf{x})$ from (2.1)), so we can use a high density of points to approximate the convolution.

Algorithm 3.1 details the process of approximating $q(u)$ using iterative convolutions. Step (1) finds the range of u. Equation (3.4) is justified by noting that the maximizer of $\mathbf{a}^\top \mathbf{x}$ over the hypercube is $\operatorname{sign}(\mathbf{a})$, and the minimizer of $\mathbf{a}^\top \mathbf{x}$ is $-\operatorname{sign}(\mathbf{a})$. If $a_i = 0$ for some i, then the corresponding x_i has no influence on the model output and we can remove this variable.

Each element q_i of the output vector $\mathbf{q}$ from Algorithm 3.1 approximates the density q at the point u_i. The sum in (3.9) is a discrete approximation to the convolution (3.2). Modern implementations of discrete convolution, such as the MATLAB `conv`, use the fast Fourier transform that enables scaling to very large N, which in turn controls the error in the q_i's. We stress that no computational model evaluations (i.e., evaluations of f from (2.1)) are needed to compute the q_i's.

3.2. Obtaining the Gaussian quadrature for the convolved density. To obtain the Gaussian quadrature rule for $q(u)$, we apply the Lanczos procedure (section 2.4.1) to the diagonal matrix $\operatorname{diag}([u_0, \ldots, u_{N-1}])$ with starting vector $[q_0^{1/2}, \ldots, q_{N-1}^{1/2}]^\top$ computed with Algorithm 3.1. Effectively, this runs the Stieltjes' procedure (section 2.3.1) using a discrete inner product defined by the trapezoidal rule with N points. Since the matrix is diagonal, all computations use $\mathcal{O}(N)$ operations; thus, we can use very large N to control errors in the quadrature rule calculation. Again, we stress that no model evaluations are needed.

3.3. Evaluating the function at the Gaussian quadrature points. We next address the issue of evaluating the ridge profile at the Gaussian quadrature nodes, $g(\lambda_j)$. Consider the linear transformation $\boldsymbol{\xi}_j = \lambda_j\, \mathbf{a}$, and recognize that

$$f(\boldsymbol{\xi}_j) \;=\; f(\lambda_j\, \mathbf{a}) \;=\; g(\mathbf{a}^\top(\lambda_j\, \mathbf{a})) \;=\; g(\lambda_j\, \mathbf{a}^\top \mathbf{a}) \;=\; g(\lambda_j), \tag{3.10}$$

since $\mathbf{a}$ is assumed to be normalized. Thus, we can evaluate $f(\boldsymbol{\xi}_j)$ in place of $g(\lambda_j)$, provided that $\boldsymbol{\xi}_j \in [-1,1]^m$. To ensure we find a valid point at which to evaluate f, consider the projection of the m-dimensional hypercube down to a one-dimensional

Copyright © by SIAM. Unauthorized reproduction of this article is prohibited.

Downloaded 11/18/19 to 138.195.214.230. Redistribution subject to SIAM license or copyright; see http://www.siam.org/journals/ojsa.php

Algorithm 3.1 Discrete convolution of densities.

Given: the vector $\mathbf{a} \in \mathbb{R}^m \setminus \{\mathbf{0}\}$
Assumptions: $\mathbf{x} \sim \mathcal{U}([-1,1]^m)$

1. Find the inputs of the one-dimensional interval

$$u_\ell \;=\; \mathbf{a}^\top \mathrm{sign}\,(-\mathbf{a})\,, \quad u_r \;=\; \mathbf{a}^\top \mathrm{sign}\,(\mathbf{a})\,, \tag{3.4}$$

and define N (where N is odd) equally spaced points along the interval

$$u_j \;=\; u_\ell + j\,\Delta u, \quad j \;=\; 0, \ldots, N-1, \tag{3.5}$$

where $\Delta u = (u_r - u_\ell)/(N-1)$.

2. Initialize the vector $\mathbf{q} = \begin{bmatrix} q(u_0) & \ldots & q(u_{N-1}) \end{bmatrix}^\top$, where

$$q(u_j) \;=\; \begin{cases} 1/(2\,a_0) & \text{if } |u_j| \le a_0, \\ 0 & \text{otherwise.} \end{cases} \tag{3.6}$$

3. For $i = 1, \ldots, m-1$,
   (i) Define $\mathbf{p} = \begin{bmatrix} p(u_0) & \ldots & p(u_{N-1}) \end{bmatrix}^\top$, where

$$p(u_j) \;=\; \begin{cases} 1/(2\,a_i) & \text{if } |u_j| \le a_i, \\ 0 & \text{otherwise.} \end{cases} \tag{3.7}$$

   (ii) For $j = 0, \ldots, N-1$, define

$$\begin{aligned} k_0 \;&=\; \max\left\{0, j - \frac{N-1}{2}\right\}, \\ k_1 \;&=\; \min\left\{\frac{N-1}{2} + j, N-1\right\}, \end{aligned} \tag{3.8}$$

and compute

$$q_j \;=\; \sum_{k=k_0}^{k_1} q_k\, p_{\frac{N-1}{2}-j+k}. \tag{3.9}$$

NOTE: skip any i for which $a_i = 0$.
Output: $\mathbf{q} = \begin{bmatrix} q_0 & \ldots & q_{N-1} \end{bmatrix}^\top$

domain via $u = \mathbf{a}^\top \mathbf{x}$. In general, the endpoints of this one-dimensional interval are defined by two opposing corners of the hypercube. The endpoints of the interval are

$$u_\ell \;=\; \mathbf{a}^\top \mathrm{sign}\,(-\mathbf{a}) \quad \text{and} \quad u_r \;=\; \mathbf{a}^\top \mathrm{sign}\,(\mathbf{a})\,, \tag{3.11}$$

and the corresponding corners of the hypercube are

$$\mathbf{x}_\ell \;=\; \mathrm{sign}\,(-\mathbf{a}) \quad \text{and} \quad \mathbf{x}_r \;=\; \mathrm{sign}\,(\mathbf{a})\,. \tag{3.12}$$

A line in m-dimensional space that connects these two corners of the hypercube is guaranteed to be contained within the hypercube. By projecting the one-dimensional

Copyright © by SIAM. Unauthorized reproduction of this article is prohibited.

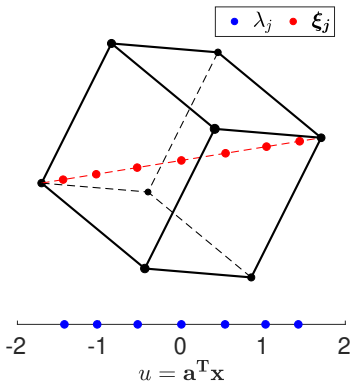


FIG. 4. *The projection of the one-dimensional quadrature nodes λ_j into m-dimensional space.*

quadrature points λ_j onto that line, we ensure that $\boldsymbol{\xi}_j \in [-1,1]^m$. We do this by

$$\boldsymbol{\xi}_j \;=\; (1-\gamma_j)\,\mathbf{x}_\ell + \gamma_j\,\mathbf{x}_r, \tag{3.13}$$

where $\gamma_j = (\lambda_j - u_\ell)/(u_r - u_\ell)$. Figure 4 illustrates this process for a three-dimensional cube.

Algorithm 3.2 summarizes the steps for building a pseudospectral polynomial approximation of a ridge function. The approximate integral of f is the first pseudospectral coefficient:

$$\hat{g}_0 \;=\; \sum_{j=0}^{d} \nu_j\, g(\lambda_j) \;\approx\; \int g(u)\,q(u)\,\mathrm{d}u \;=\; \int f(\mathbf{x})\,p(\mathbf{x})\,\mathrm{d}\mathbf{x}. \tag{3.14}$$

Algorithm 3.2 contains two levels of approximation: (i) the discrete approximation of $q(u)$ using an N-point trapezoidal rule and (ii) the number $d+1$ of Lanczos iterations performed. The number of Lanczos iterations corresponds to the number of terms in the polynomial approximation of g and the number of Gaussian quadrature nodes. The latter is important as this is the number of model evaluations required to compute the pseudospectral coefficients and the estimated integral. In general, we should choose $N \gg d$ since the approximation of q at the N trapezoidal points does not require any function evaluations, which are typically the most expensive step. In the next section, we numerically study this two-level approximation.

3.4. Numerical study. In this section, we numerically study the behavior of Algorithm 3.2 for

$$y \;=\; \sin\left(2\,\pi\,(\mathbf{a}^{\top}\mathbf{x})\right) + \cos\left(\frac{\pi}{2}(\mathbf{a}^{\top}\mathbf{x})\right), \qquad \mathbf{x} \in \mathbb{R}^{25}. \tag{3.15}$$

We assume $\mathbf{x} \sim \mathcal{U}([-1,1]^{25})$. Notice that (3.15) is an exact one-dimensional ridge function.

Figure 5 contains the results from Algorithm 3.2 for $N = 10,000$ and $d = 50$. The first plot shows the one-dimensional ridge profile of (3.15) in black with the polynomial approximation computed using the generalized Gaussian rule from Algorithm 3.2 denoted by the blue dashed line. The inverse cumulative distribution function method using Gauss–Legendre quadrature from Tsilifis [69] is shown in green for comparison. The absolute error of each method at each point is shown in the

Downloaded 11/18/19 to 138.195.214.230. Redistribution subject to SIAM license or copyright; see http://www.siam.org/journals/ojsa.php

Copyright © by SIAM. Unauthorized reproduction of this article is prohibited.

Downloaded 11/18/19 to 138.195.214.230. Redistribution subject to SIAM license or copyright; see http://www.siam.org/journals/ojsa.php

Algorithm 3.2 Univariate polynomial approximation of a ridge function.

Given: function $f : \mathbb{R}^m \to \mathbb{R}$ and unit vector $\mathbf{a} \in \mathbb{R}^m \setminus \{\mathbf{0}\}$ such that

$$y \;=\; f(\mathbf{x}) \;=\; g(\mathbf{a}^\top \mathbf{x})$$

for some unknown $g : \mathbb{R} \to \mathbb{R}$

Assumptions: $\mathbf{x} \sim \mathcal{U}([-1,1]^m)$ and f is square-integrable with respect to the input density $p(\mathbf{x})$

1. Perform Algorithm 3.1 to obtain $\mathbf{q} = \begin{bmatrix} q(u_0) & \dots & q(u_{N-1}) \end{bmatrix}^\top$.
2. Perform Algorithm 2.2 on

$$\boldsymbol{A} \;=\; \begin{bmatrix} u_0 & & \\ & \ddots & \\ & & u_{N-1} \end{bmatrix}, \quad \tilde{\mathbf{v}}_0 \;=\; \begin{bmatrix} \sqrt{q(u_0)} \\ \vdots \\ \sqrt{q(u_{N-1})} \end{bmatrix}, \tag{3.16}$$

   to obtain the Jacobi matrix $\boldsymbol{T}$.
3. Take the eigendecomposition of $\boldsymbol{T}$,

$$\boldsymbol{T} \;=\; \boldsymbol{Q}\,\boldsymbol{\Lambda}\,\boldsymbol{Q}^\top, \tag{3.17}$$

   where the eigenvalues are $\{\lambda_0, \lambda_1, \dots, \lambda_d\}$ and the eigenvectors are normalized. Define $\nu_j = (\boldsymbol{Q})_{0,j}^2$ for $j = 0, \dots, d$.
4. For $j = 0, \dots, d$, compute

$$\boldsymbol{\xi}_j \;=\; (1-\gamma_j)\,\mathbf{x}_\ell + \gamma_j\,\mathbf{x}_r \quad \text{for} \quad \gamma_j = (\lambda_j - u_\ell)\,/\,(u_r - u_\ell), \tag{3.18}$$

   where $\mathbf{x}_\ell = \operatorname{sign}(-\mathbf{a})$, $\mathbf{x}_r = \operatorname{sign}(\mathbf{a})$, $u_\ell = \mathbf{a}^\top \operatorname{sign}(-\mathbf{a})$, and $u_r = \mathbf{a}^\top \operatorname{sign}(\mathbf{a})$.
5. Compute the pseudospectral coefficients

$$\hat{g}_i \;=\; \sum_{j=0}^{d} \nu_j\, f(\boldsymbol{\xi}_j)\, \phi_i(\lambda_j). \tag{3.19}$$

6. Build the pseudospectral expansion

$$y \;=\; g(u) \;\approx\; \sum_{i=0}^{d} \hat{g}_i\, \phi_i(u). \tag{3.20}$$

second plot, and the density $q(u)$ is in the third plot. Visually, both methods appear to perform reasonably well in the center of the domain, where the induced density function is relatively large. However, by examining the absolute errors of each method, we see that Algorithm 3.2 outperforms the pseudospectral approximation constructed using Gauss–Legendre quadrature by several orders of magnitude. Near the endpoints, the errors in each approximation method begin to increase, but $q(u)$ in these regions is many orders of magnitude smaller than in the middle of the domain.

Next, we study the two levels of approximation in Algorithm 3.2: (i) the number N of trapezoidal rule points used to construct the discrete approximation of $q(u)$ and (ii) the number d of Lanczos iterations performed. The latter corresponds to the degree of the polynomial expansion as well as the number of Gaussian quadrature nodes. Figure 6 shows the approximate L^2 norm of the error between f and the

Copyright © by SIAM. Unauthorized reproduction of this article is prohibited.

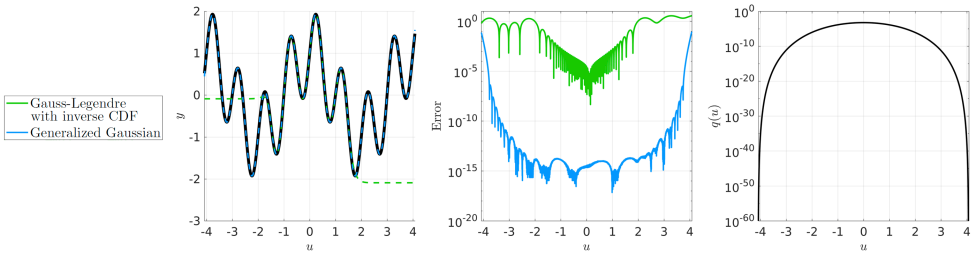


FIG. 5. *The results of Algorithm* 3.2 *applied to* (3.15). *The first plot shows the true ridge profile (in black) and the* $d = 50$ *pseudospectral polynomial approximation (in blue) and the Gauss–Legendre approximation from Tsilifis* [69] *(in green). The second plot contains the absolute error of the approximations, and the third plot shows the density* $q(u)$.

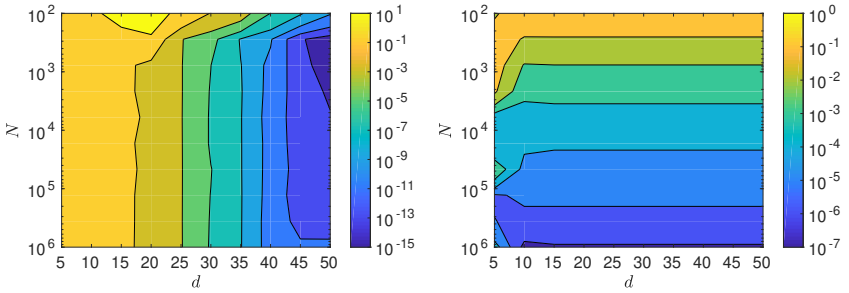


FIG. 6. *Results from studies of the two levels of approximation in Algorithm* 3.2 *applied to* (3.15). *The left plot contains* L^2 *errors in the polynomial approximation for various values of* N *and* d. *The right plot shows errors in the approximate integral of* (3.15).

pseudospectral approximations for varying values of N and d. The error depends strongly on d. This is because a high-degree polynomial is required to fit the highly oscillatory $f(\mathbf{x})$. The rightmost plot contains approximations of the integral of $f(\mathbf{x})$ using the first coefficient in the pseudospectral expansion. Here, we see a strong dependence on N. This is because integration errors in the Gaussian quadrature decay quickly with d. To improve the approximation, the discrete approximation of the density $q(u)$ must be improved.

4. Extension to near-one-dimensional ridge functions. The approach presented in the previous section produces a univariate Gaussian quadrature for an exact ridge function. However, in many if not most applications, the exploratory analysis suggests that the computational model's input/output map can be merely approximated by a one-dimensional ridge function. Evidence for this near ridge structure appears in Figure 1, which shows the one-dimensional shadow plots for six different computational science applications. In each plot, the data show relatively small deviation (relative to the range of the data) from a univariate function.

Next we consider how Algorithm 3.2 can be extended for the case when $f(\mathbf{x})$ is well-approximated by a one-dimensional ridge function,

$$y = f(\mathbf{x}) \approx g(u), \quad \text{where} \quad u = \mathbf{a}^\top \mathbf{x}. \tag{4.1}$$

The best L^2 approximation of f by g is the expected value of the output conditioned

Copyright © by SIAM. Unauthorized reproduction of this article is prohibited.

Downloaded 11/18/19 to 138.195.214.230. Redistribution subject to SIAM license or copyright; see http://www.siam.org/journals/ojsa.php

Downloaded 11/18/19 to 138.195.214.230. Redistribution subject to SIAM license or copyright; see http://www.siam.org/journals/ojsa.php

on $u = \mathbf{a}^\top \mathbf{x}$ [59, Chap. 8]. That is,

$$g(u) = \mathbb{E}\left[f(\mathbf{x}) \,|\, \mathbf{a}^\top \mathbf{x} = u\right]. \tag{4.2}$$

By the tower property of conditional expectations [4], we can write the integral of f as

$$\int f(\mathbf{x})\, p(\mathbf{x})\, \mathrm{d}\mathbf{x} \;=\; \int g(u)\, q(u)\, \mathrm{d}u. \tag{4.3}$$

Thus, if we can evaluate the conditional expectation $g(u)$, then we can construct a univariate polynomial approximation and Gaussian quadrature for near ridge functions. We can measure the degree to which f is a near-one-dimensional ridge function using the conditional variance $\mathrm{Var}\left[f(\mathbf{x}) \,|\, \mathbf{a}^\top \mathbf{x} = u\right]$. This conditional variance is the error for the best one-dimensional ridge approximation of $f(\mathbf{x})$.

The difficulty in applying the methodology from section 3 to (4.1) is how to compute $g(\lambda_j)$. Recall from (3.10) that, in the case of the exact ridge function, we can transform λ_j into a corresponding input $\boldsymbol{\xi}_j$ in the full-dimensional space and evaluate f at this point, provided that $\boldsymbol{\xi}_j$ is in the domain of f. In the near ridge case, we want to approximate the average of $f(\mathbf{x})$ over all $\mathbf{x} \in [-1,1]^m$ such that $\lambda_j = \mathbf{a}^\top \mathbf{x}$. We write the sample approximation of this conditional expectation as

$$g(\lambda_j) \;\approx\; \hat{g}(\lambda_j) \;=\; \frac{1}{M_j} \sum_{i=0}^{M_j - 1} f(\boldsymbol{\xi}_{j,i}), \tag{4.4}$$

where the M_j input values $\boldsymbol{\xi}_{j,i} \in [-1,1]^m$ are sampled uniformly conditioned on $\mathbf{a}^\top \boldsymbol{\xi}_{j,i} = \lambda_j$. To compute (4.4), we use a hit-and-run sampling algorithm [54]. Start by choosing a random (unit) direction $\mathbf{w} \in \mathbb{R}^m \setminus \{\mathbf{0}\}$ that is orthogonal to the ridge direction $\mathbf{a}$. We then pick a step size $t \in [-\sqrt{m}, \sqrt{m}]$, where this range is used to ensure that the step size covers the maximum possible range of the rotated $[-1,1]^m$ hypercube. To obtain the $(i+1)$st conditional sample, we step from the previously drawn sample, $\boldsymbol{\xi}_{j,i+1} = \boldsymbol{\xi}_{j,i} + t\,\mathbf{w}$, provided that $\boldsymbol{\xi}_{j,i+1} \in [-1,1]^m$. If this is not the case, then we choose a new random step size until a valid conditional sample is obtained.

The second issue in approximating a near-one-dimensional ridge function is that the sample approximation of the conditional expectation in (4.4) results in noisy estimates of $g(\lambda_j)$. For this reason, constructing an interpolating polynomial, as described in section 3, is not the best approach given a restricted computational budget. We recommend truncating the pseudospectral polynomial expansion to fewer than d terms (where d is the number of Gaussian quadrature points) to avoid overfitting. For each Gaussian quadrature node λ_j, we have $\boldsymbol{\xi}_{j,i}$ for $i = 0, \ldots, M_j - 1$. Estimate the standard error in each approximation of the conditional expectation, $s_j = \hat{\sigma}_j / \sqrt{M_j}$, where $\hat{\sigma}_j$ is the standard deviation of the $f(\boldsymbol{\xi}_{j,i})$, $i = 0, \ldots, M_j - 1$. When $g(u)$ is smooth, we expect decay in the coefficients g_i from (2.7) for sufficiently large i [67]. The pseudospectral coefficients, $\hat{g}_i$ from (2.8), approximate the true coefficients. We suggest truncating the expansion at a degree $\tilde{d} \leq d$ polynomial, where $|\hat{g}_i| < \sum_{j=0}^{d} s_j/(d+1)$ for all $i > \tilde{d}$. This heuristic removes terms whose contribution to the expansion is smaller than the noise in the sample approximations $\hat{g}(\lambda_j)$. Algorithm 4.1 formalizes the process for building a pseudospectral approximation of a near-one-dimensional ridge function.

4.1. Error analysis. We briefly explore the sources of error in the approximation and integration methods for near-one-dimensional ridge functions. For this dis-

Copyright © by SIAM. Unauthorized reproduction of this article is prohibited.

Downloaded 11/18/19 to 138.195.214.230. Redistribution subject to SIAM license or copyright; see http://www.siam.org/journals/ojsa.php

Algorithm 4.1 Univariate polynomial approximation of a near ridge function.

Given: function $f : \mathbb{R}^m \to \mathbb{R}$ and unit vector $\mathbf{a} \in \mathbb{R}^m \setminus \{\mathbf{0}\}$ such that

$$y \;=\; f(\mathbf{x}) \;\approx\; g(\mathbf{a}^\top \mathbf{x})$$

for some unknown $g : \mathbb{R} \to \mathbb{R}$

Assumptions: $\mathbf{x} \sim \mathcal{U}([-1,1]^m)$ and f is square-integrable with respect to the input density $p(\mathbf{x})$

1. Perform steps 1–3 of Algorithm 3.2.
2. For $j = 0, \ldots, d$,
   (i) Compute

   $$\boldsymbol{\xi}_{j,0} \;=\; (1-\gamma_j)\,\mathbf{x}_\ell + \gamma_j\,\mathbf{x}_r, \quad \text{for} \quad \gamma_j = (\lambda_j - u_\ell)\,/\,(u_r - u_\ell), \tag{4.5}$$

   where $\mathbf{x}_\ell = \operatorname{sign}(-\mathbf{a})$, $\mathbf{x}_r = \operatorname{sign}(\mathbf{a})$, $u_\ell = \mathbf{a}^\top \operatorname{sign}(-\mathbf{a})$, and $u_r = \mathbf{a}^\top \operatorname{sign}(\mathbf{a})$.
   (ii) Evaluate $f(\boldsymbol{\xi}_{j,0})$.
   (iii) For $i = 1, \ldots, M_j - 1$,
      (a) Choose a random vector $\mathbf{w} \in \mathbb{R}^m \setminus \{\mathbf{0}\}$ with $\mathbf{w}^\top \mathbf{a} = 0$ and $||\mathbf{w}||_2 = 1$.
      (b) Choose a random value $t \in [-\sqrt{m}, \sqrt{m}]$.
      (c) If $||\boldsymbol{\xi}_{j,i} + t\,\mathbf{w}||_\infty <= 1$, then set $\boldsymbol{\xi}_{j,i+1} = \boldsymbol{\xi}_{j,i} + t\,\mathbf{w}$. Otherwise, repeat step (b).
      (d) Evaluate $f(\boldsymbol{\xi}_{j,i+1})$.
   (iv) Approximate the conditional expectation at the quadrature point,

   $$g(\lambda_j) \;\approx\; \hat{g}(\lambda_j) \;=\; \frac{1}{M_j} \sum_{i=0}^{M_j - 1} f(\boldsymbol{\xi}_{j,i}) \tag{4.6}$$

   and the standard error $s_j \;=\; \hat{\sigma}_j / \sqrt{M_j}$, where $\hat{\sigma}_j$ is the standard deviation of the $f(\boldsymbol{\xi}_{j,i})$ for $i = 0, \ldots, M_j - 1$.
3. Compute the pseudospectral coefficients

   $$\hat{g}_i \;=\; \sum_{j=0}^{d} \nu_j\, \hat{g}(\lambda_j)\, \phi_i(\lambda_j). \tag{4.7}$$

4. Define $\tilde{d}$ to be the largest index such that $|\hat{g}_i| < \sum_{j=0}^{d} s_j/(d+1)$ for all $i > \tilde{d}$.
5. Build the pseudospectral expansion

   $$y \;=\; g(u) \;\approx\; \sum_{i=0}^{\tilde{d}} \hat{g}_i\, \phi_i(u). \tag{4.8}$$

cussion, let $p_d(u)$ denote the d-degree pseudospectral polynomial approximation of the ridge profile $g(u)$ constructed by Algorithm 4.1. Define the m-dimensional analogs of p_d and g by

$$p_{d,\mathbf{x}}(\mathbf{x}) \;=\; p_d(\mathbf{a}^\top \mathbf{x}) \quad \text{and} \quad g_{\mathbf{x}}(\mathbf{x}) \;=\; g(\mathbf{a}^\top \mathbf{x}). \tag{4.9}$$

This enables us to study the L^2 errors without changing the behavior of the one-dimensional functions p_d and g.

Copyright © by SIAM. Unauthorized reproduction of this article is prohibited.

Downloaded 11/18/19 to 138.195.214.230. Redistribution subject to SIAM license or copyright; see http://www.siam.org/journals/ojsa.php

Using the triangle inequality, we can decompose the error in the one-dimensional approximation of f as

$$\|f - p_{d,\mathbf{x}}\|_{L^2} \;\leq\; \|f - g_{\mathbf{x}}\|_{L^2} + \|g_{\mathbf{x}} - p_{d,\mathbf{x}}\|_{L^2}\,. \tag{4.10}$$

The two terms on the right-hand side of (4.10) represent the degree to which $f(\mathbf{x})$ is well-approximated by a one-dimensional ridge function and the degree to which we approximate the ridge function by a polynomial, respectively. For exact ridge functions (see section 3), $\|f - g_{\mathbf{x}}\|_{L^2} = 0$ and the error only depends on the ability to approximate the ridge profile by a polynomial. For an accessible treatment of univariate polynomial approximation, see Trefethen's *Approximation Theory and Approximation Practice* [67]. For near-one-dimensional ridge functions, the ridge approximation error is nonzero. This implies that the error in the one-dimensional polynomial approximation of $f(\mathbf{x})$ cannot be reduced below this threshold. We show this behavior numerically in the next sections.

The integration error is tied to the approximation error. Using Hölder's inequality,

$$\|f - g_{\mathbf{x}}\|_{L^1} \;\leq\; \|f - g_{\mathbf{x}}\|_{L^2}\,. \tag{4.11}$$

Then,

$$\|f - g_{\mathbf{x}}\|_{L^1} \;=\; \int |f - g_{\mathbf{x}}|\; p(\mathbf{x})\,\mathrm{d}\mathbf{x} \;\geq\; \left|\int (f - g_{\mathbf{x}})\, p(\mathbf{x})\,\mathrm{d}\mathbf{x}\right| \;=\; |\mathbb{E}\,[f] - \mathbb{E}\,[g_{\mathbf{x}}]|\,. \tag{4.12}$$

Thus,

$$|\mathbb{E}\,[f] - \mathbb{E}\,[g_{\mathbf{x}}]| \;\leq\; \|f - g_{\mathbf{x}}\|_{L^2}\,. \tag{4.13}$$

4.2. Numerical studies: Example 1. In this section, we numerically study the behavior of Algorithm 4.1 for approximating near-one-dimensional ridge functions. We consider the function

$$y \;=\; \sin\left(\frac{\pi}{5}\,(\mathbf{a}^\top\mathbf{x})\right) + \frac{1}{5}\,\cos\left(\frac{4\,\pi}{5}(\mathbf{a}^\top\mathbf{x})\right) + \frac{1}{40}\mathbf{x}^\top\boldsymbol{B}\,\mathbf{1}, \qquad \mathbf{x}\in\mathbb{R}^{\mathbf{25}}, \tag{4.14}$$

where $\mathbf{a}\in\mathbb{R}^{25}\setminus\{\mathbf{0}\}$ defines the ridgelike structure, $\boldsymbol{B}\in\mathbb{R}^{25\times 24}$ contains an orthonormal basis for the subspace orthogonal to $\mathbf{a}$, and $\mathbf{1}\in\mathbb{R}^{24}$ is a vector of ones. We assume $\mathbf{x}\sim\mathcal{U}([-1,1]^{25})$.

Figure 7 shows the results of using the extension of Algorithm 3.2 to (4.14). The plot on the left is a shadow plot of evaluations of (4.14) against $u = \mathbf{a}^\top\mathbf{x}$. The spread in the plot is due to variations in the 24 directions orthogonal to $\mathbf{a}$. The red line is the polynomial approximation of the ridge profile—$g(u) = \mathbb{E}\left[\, f(\mathbf{x}) \,|\, \mathbf{a}^\top\mathbf{x} = u\right]$. This is computed using $d = 11$ and $M = 50$ total function evaluations distributed among the 12 Gaussian quadrature nodes. The polynomial expansion is truncated at $\tilde{d} = 6$ to avoid overfitting the noise in the approximation of $g(\lambda_j)$. Note that fitting a polynomial of total degree 6 in 25 dimensions would require at least $M = \binom{25+6}{6} = 736,281$ to have a well-posed fitting problem.

4.3. Numerical studies: Example 2. Next, we consider a physically motivated problem in magnetohydrodynamics: the Hartmann problem [22]. The Hartmann problem models the flow of an ionized fluid in the presence of a perpendicular magnetic field along an infinitely long channel (see Figure 8). The magnetic field

Copyright © by SIAM. Unauthorized reproduction of this article is prohibited.

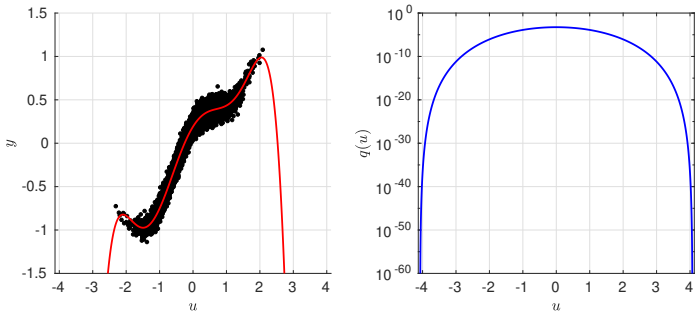


FIG. 7. *The results of the extension to Algorithm* 3.2 *applied to the approximate ridge function* (4.14). *The left plot contains a show plot of* $f(\mathbf{x})$ *along the* $u = \mathbf{a}^\top \mathbf{x}$ *axis with the* $d = 11$ *polynomial approximation on top of it. The right plot shows the density* $q(u)$.

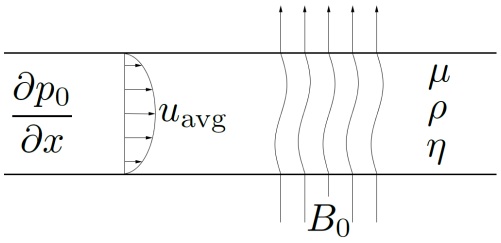


FIG. 8. *An illustration of the Hartmann problem, which models the flow of an ionized fluid in the presence of a perpendicular magnetic field along an infinitely long channel.*

TABLE 1
Descriptions and ranges of the five variable inputs to (4.15). *Note that the range of values is given in terms of the log of each input.*

Symbol	Description	Range of log(*input*)
μ	fluid viscosity	$[\log(0.05), \log(0.2)]$
ρ	fluid density	$[\log(1), \log(5)]$
$\partial p_0/\partial x$	applied pressure gradient	$[\log(0.5), \log(3)]$
η	magnetic resistivity	$[\log(0.5), \log(3)]$
B_0	applied magnetic field	$[\log(0.25), \log(1)]$

acts as a resistive force on the flow while the fluid induces a horizontal component in the magnetic field. For this work, we consider the average flow velocity across the channel, denoted by u_{avg}, as the output of interest. This quantity can be written as

$$u_{\text{avg}}(\mathbf{x}) = -\frac{\partial p_0}{\partial x}\frac{\eta}{B_0^2}\left(1 - \frac{\ell B_0}{\sqrt{\eta\mu}}\coth\left(\frac{\sqrt{\eta\mu}}{B_0\ell}\right)\right), \tag{4.15}$$

where the five input variables are described in Table 1. The constant values ℓ and μ_0 are the width of the channel and the magnetic permeability of free space (i.e., a universal constant), respectively. Recent work showed that (4.15) exhibits approximate one-dimensional ridge structure with respect to the log-transformed inputs [32]. Thus, we consider (4.15) with respect to the inputs

$$\mathbf{x} = \begin{bmatrix}\log(\mu) & \log(\rho) & \log(\partial p_0/\partial x) & \log(\eta) & \log(B_0)\end{bmatrix}^\top. \tag{4.16}$$

We assume a uniform density function over the range of values given in Table 1.

Copyright © by SIAM. Unauthorized reproduction of this article is prohibited.

Downloaded 11/18/19 to 138.195.214.230. Redistribution subject to SIAM license or copyright; see http://www.siam.org/journals/ojsa.php

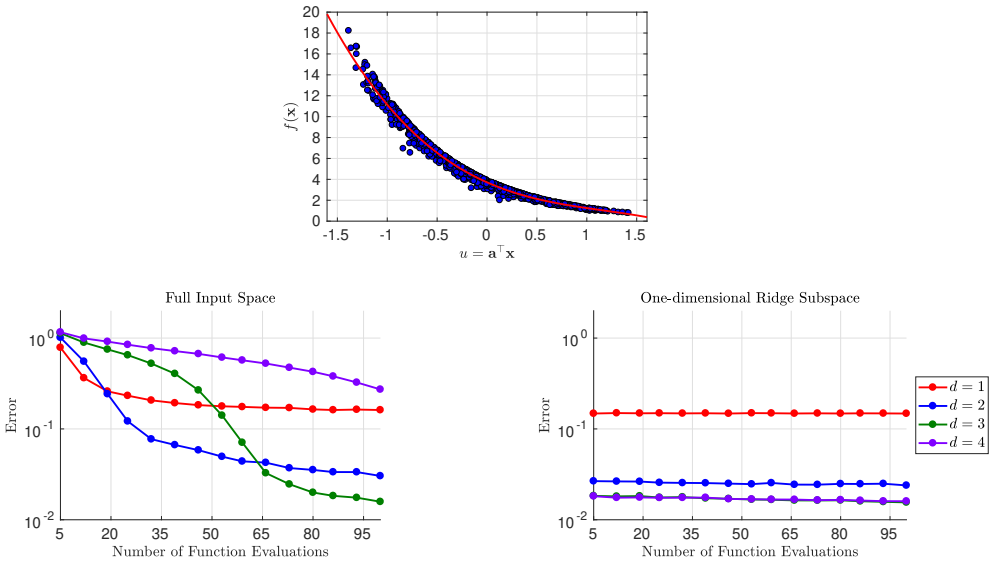


FIG. 9. *The results of applying the extension of Algorithm* 3.2 *to* (4.15). *The top plot is a shadow plot of* u_{avg} *against* $u = \mathbf{a}^\top \mathbf{x}$. *The bottom plots show* L^2 *errors of polynomial approximations constructed on the full five-dimensional input space (on the left) and the one-dimensional ridge subspace (on the right).*

Figure 9 contains the results of applying Algorithm 4.1 to (4.15). The top plot is a shadow plot of u_{avg} against $u = \mathbf{a}^\top \mathbf{x}$ overlaid with a polynomial approximation of the ridge profile. This approximation was constructed using $d = 4$ with a total computational budget of 100 function evaluations (20 for each of the 5 quadrature nodes). The bottom plots show the approximated relative L^2 errors in the polynomial approximations constructed on the full five-dimensional input space (on the left) and the one-dimensional ridge subspace (on the right). On the full input space, we use uniformly sampled points from the $[-1, 1]^5$ hypercube and construct the least-squares polynomial approximation using Legendre polynomials with L^2 regularization. These approximations perform poorly when the computational budget is limited and restrict our optimal choice of polynomial degree. This issue grows exponentially as the dimension of the given function increases. The one-dimensional approximation constructed using Algorithm 4.1 achieves its optimal performance with very few function evaluations. For the restricted computational budget studied, the one-dimensional approximation outperforms its five-dimensional counterpart. Given a larger computational budget, the full polynomial approximation will be more accurate. This is because the one-dimensional approximation is limited by the accuracy of approximating $u_{\text{avg}}(\mathbf{x})$ by a ridge function (recall (4.10)). We estimate the degree to which $u_{\text{avg}}(\mathbf{x})$ is well-approximated by a ridge function as

$$\|u_{\text{avg}} - g_{\mathbf{x}}\|_{L^2} \approx 1.29 \times 10^{-2}, \tag{4.17}$$

which is approximately where the L^2 error in the one-dimensional ridge approximation begins to level out. The value of (4.17) is approximated by taking 1000 uniformly sampled points from $[-1, 1]^5$ and transforming them into $u_i = \mathbf{a}^\top \mathbf{x}_i$. At each u_i, we obtain 100 randomly sampled points orthogonal to $\mathbf{a}$. Such a study is infeasible for

Downloaded 11/18/19 to 138.195.214.230. Redistribution subject to SIAM license or copyright; see http://www.siam.org/journals/ojsa.php

Copyright © by SIAM. Unauthorized reproduction of this article is prohibited.

Downloaded 11/18/19 to 138.195.214.230. Redistribution subject to SIAM license or copyright; see http://www.siam.org/journals/ojsa.php

an expensive computational model, but it is useful in explaining the behavior of the one-dimensional ridge approximations introduced in this paper.

5. Summary and extensions. We introduce a novel algorithm for estimating the integral and constructing a polynomial approximation of a one-dimensional ridge function based on Lanczos' method. In general, building a polynomial surrogate of an m-dimensional function suffers from the curse of dimensionality—an exponential increase in computational costs resulting from increases in m. We also introduce an approach to extending this algorithm to functions that are well-approximated by a one-dimensional ridge function.

We numerically study the new algorithm on several test problems, including exact and approximate ridge functions. We show that exploiting low-dimensional structure can result in exponential savings while maintaining accuracy. Additionally, we study the two-level approximation behavior of the algorithm: the first level is a discrete approximation of the induced density function $q(u)$, and the second level is the number of Lanczos iterations, which corresponds to the degree of the polynomial approximation of the ridge function as well as the number of Gaussian quadrature points. In studying the extension of the algorithm to near-one-dimensional ridge functions, we show that we can quickly achieve the ridge approximation error using very few function evaluations.

In section 1.1, we mentioned that projection pursuit regression [26] uses a model for the conditional mean of the regression that is a sum of ridge functions, where each ridge profile in the sum is a smoothing spline. Classically, a maximum likelihood approach is used to estimate the directions and spline parameters. Once those parameters are estimated, our approach can be used to accurately estimate the integral of the projection pursuit regression model, since

$$\int \left(\sum_i g_i(\mathbf{a}_i^\top \mathbf{x}) \right) p(\mathbf{x})\, \mathrm{d}\mathbf{x} \;=\; \sum_i \left(\int g_i(\mathbf{a}_i^\top \mathbf{x})\, p(\mathbf{x})\, \mathrm{d}\mathbf{x} \right). \tag{5.1}$$

Essentially, one could repeat our process for each pair of $\mathbf{a}_i$ and g_i, and then add the contributions.

Finally, we mention the difficulty in extending our approach to more than one dimension. Some functions that arise in computational science may be well-approximated by a *generalized* ridge function $g(\boldsymbol{A}^\top \mathbf{x})$, where $\boldsymbol{A} \in \mathbb{R}^{m \times n}$ with $m > n$ and $g : \mathbb{R}^n \to \mathbb{R}$. Unfortunately, the vector $\boldsymbol{A}^\top \mathbf{x}$ does not in general contain independent components even when $\mathbf{x}$'s components are independent. Therefore, straightforward tensor product extensions to univariate Gaussian quadrature are not appropriate. In principle, one could use linear programming extensions for Gaussian quadrature on convex domains [60, 44]. We leave such extensions for future work.

REFERENCES

[1] M. Allen and K. Maute, *Reliability-based design optimization of aeroelastic structures*, Struct. Multidiscip. Optim., 27 (2004), pp. 228–242, https://doi.org/10.1007/s00158-004-0384-1.

[2] J. A. Beck, J. M. Brown, A. A. Kaszynski, and E. B. Carper, *Active subspace development of integrally bladed disk dynamic properties due to manufacturing variations*, in ASME Turbo Expo: Power for Land, Sea, and Air, Volume 7A: Structures and Dynamics, paper GT2018-76800, ASME, New York, 2018, V07AT32A011, https://doi.org/10.1115/GT2018-76800.

Copyright © by SIAM. Unauthorized reproduction of this article is prohibited.

[3] S. H. Berguin and D. N. Mavris, *Dimensionality reduction using principal component analysis applied to the gradient*, AIAA Journal, 53 (2014), pp. 1078–1090, https://doi.org/10.2514/1.J053372.
[4] P. Billingsley, *Probability and Measure*, Wiley, New York, 1986, https://www.wiley.com/en-us/Probability+and+Measure%2C+Anniversary+Edition-p-9781118341919.
[5] H.-J. Bungartz and M. Griebel, *Sparse grids*, Acta Numer., 13 (2004), pp. 147–269, https://doi.org/10.1017/S0962492904000182.
[6] R. E. Caflisch, *Monte Carlo and quasi-Monte Carlo methods*, Acta Numer., 7 (1998), pp. 1–49, https://doi.org/10.1017/S0962492900002804.
[7] W. Cheney and W. Light, *A Course in Approximation Theory*, American Mathematical Society, Providence, RI, 2000.
[8] A. Cohen, I. Daubechies, R. DeVore, G. Kerkyacharian, and D. Picard, *Capturing ridge functions in high dimensions from point queries*, Constr. Approx., 35 (2012), pp. 225–243, https://doi.org/10.1007/s00365-011-9147-6.
[9] P. G. Constantine, *Active Subspaces: Emerging Ideas for Dimension Reduction in Parameter Studies*, SIAM Spotlights, SIAM, Philadelphia, 2015, http://doi.org/10.1137/1.9781611973860.
[10] P. G. Constantine and A. Doostan, *Time-dependent global sensitivity analysis with active subspaces for a lithium ion battery model*, Stat. Anal. Data Min., 10 (2017), pp. 243–262, https://doi.org/10.1002/sam.11347.
[11] P. G. Constantine, E. Dow, and Q. Wang, *Active subspace methods in theory and practice: Applications to kriging surfaces*, SIAM J. Sci. Comput., 36 (2014), pp. A1500–A1524, https://doi.org/10.1137/130916138.
[12] P. G. Constantine, A. Eftekhari, J. Hokanson, and R. Ward, *A near-stationary subspace for ridge approximation*, Comput. Methods Appl. Mech. Engrg., 326 (2017), pp. 402–421, https://doi.org/10.1016/j.cma.2017.07.038.
[13] P. G. Constantine, A. Eftekhari, and M. B. Wakin, *Computing active subspaces efficiently with gradient sketching*, in IEEE 6th International Workshop on Computational Advances in Multi-Sensor Adaptive Processing (CAMSAP), Cancun, IEEE, Piscataway, NJ, 2015, pp. 353–356, https://doi.org/10.1109/CAMSAP.2015.7383809.
[14] P. G. Constantine, M. S. Eldred, and E. T. Phipps, *Sparse pseudospectral approximation method*, Comput. Methods Appl. Mech. Engrg., 229 (2012), pp. 1–12, https://doi.org/10.1016/j.cma.2012.03.019.
[15] P. G. Constantine, M. Emory, J. Larsson, and G. Iaccarino, *Exploiting active subspaces to quantify uncertainty in the numerical simulation of the HyShot II scramjet*, J. Comput. Phys., 302 (2015), pp. 1–20, https://doi.org/10.1016/j.jcp.2015.09.001.
[16] P. Constantine and D. Gleich, *Computing Active Subspaces with Monte Carlo*, preprint, https://arxiv.org/abs/1408.0545 (2015).
[17] P. G. Constantine and E. T. Phipps, *A Lanczos method for approximating composite functions*, Appl. Math. Comput., 218 (2012), pp. 11751–11762, https://doi.org/10.1016/j.amc.2012.05.009.
[18] P. G. Constantine, B. Zaharatos, and M. Campanelli, *Discovering an active subspace in a single-diode solar cell model*, Stat. Anal. Data Min., 8 (2015), pp. 264–273, https://doi.org/10.1002/sam.11281.
[19] R. D. Cook, *Using dimension-reduction subspaces to identify important inputs in models of physical systems*, in Proceedings of the Section on Physical and Engineering Sciences, American Statistical Association, Alexandria, VA, 1994, pp. 18–25, http://users.stat.umn.edu/~rdcook/SDR/ASA94.pdf.
[20] R. D. Cook, *Regression Graphics: Ideas for Studying Regression through Graphics*, Wiley, New York, 1998, http://doi.org/10.1002/9780470316931.
[21] A. Cortesi, P. Constantine, T. Magin, and P. M. Congedo, *Forward and Backward Uncertainty Quantification with Active Subspaces: Application to Hypersonic Flows Around a Cylinder*, preprint, https://hal.inria.fr/hal-01592591/ (2017).
[22] T. G. Cowling and R. B. Lindsay, *Magnetohydrodynamics*, Physics Today, 10 (1957), p. 40, http://doi.org/10.1063/1.3060498.
[23] D. L. Donoho, *High-dimensional data analysis: The curses and blessings of dimensionality*, in AMS Conference on Math Challenges of the 21st Century, AMS, Providence, RI, 2000, http://citeseerx.ist.psu.edu/viewdoc/summary?doi=10.1.1.329.3392.
[24] A. Eftekhari, M. B. Wakin, P. Li, P. G. Constantine, and R. A. Ward, *Learning the second-moment matrix of a smooth function from point samples*, in 51st Asilomar Conference on Signals, Systems, and Computers, Asilomar, CA, IEEE, Piscataway, NJ, 2017, pp. 671–675, https://doi.org/10.1109/ACSSC.2017.8335427.

Downloaded 11/18/19 to 138.195.214.230. Redistribution subject to SIAM license or copyright; see http://www.siam.org/journals/ojsa.php

Copyright © by SIAM. Unauthorized reproduction of this article is prohibited.

[25] M. Fornasier, K. Schnass, and J. Vybiral, *Learning functions of few arbitrary linear parameters in high dimensions*, Found. Comput. Math., 12 (2012), pp. 229–262, http://doi.org/10.1007/s10208-012-9115-y.

[26] J. H. Friedman and W. Stuetzle, *Projection pursuit regression*, J. Amer. Statist. Assoc., 76 (1981), pp. 817–823, https://doi.org/10.1080/01621459.1981.10477729.

[27] W. Gautschi, *Orthogonal Polynomials*, Oxford University Press, Oxford, 2004, https://global.oup.com/academic/product/orthogonal-polynomials-9780198506720.

[28] R. Ghanem and P. Spanos, *Stochastic Finite Elements: A Spectral Approach*, Springer, New York, 1991, https://www.springer.com/us/book/9781461277958.

[29] J. M. Gilbert, J. L. Jefferson, P. G. Constantine, and R. M. Maxwell, *Global spatial sensitivity of runoff to subsurface permeability using the active subspace method*, Adv. Water Res., 92 (2016), pp. 30–42.

[30] M. B. Giles, *Multilevel Monte Carlo methods*, Acta Numer., 24 (2015), pp. 259–328, https://doi.org/10.1017/S096249291500001X.

[31] A. Glaws, P. G. Constantine, and R. D. Cook, *Inverse regression for ridge recovery: A data-driven approach for parameter space dimension reduction in computational science*, SIAM J. Math. Data Sci., to appear.

[32] A. Glaws, P. G. Constantine, J. Shadid, and T. M. Wildey, *Dimension reduction in MHD power generation models: Dimensional analysis and active subspaces*, Stat. Anal. Data Min., 10 (2017), pp. 312–325, https://doi.org/10.1002/sam.11355.

[33] G. H. Golub and G. Meurant, *Matrices, Moments, and Quadrature with Applications*, Princeton University Press, Princeton, NJ, 2010, http://press.princeton.edu/titles/9104.html.

[34] G. H. Golub and J. H. Welsch, *Calculation of Gauss quadrature rules*, Math. Comp., 23 (1969), pp. 221–230, https://doi.org/10.1090/S0025-5718-69-99647-1.

[35] R. B. Gramacy and H. Lian, *Gaussian process single-index models as* emulators for computer experiments*, Technometrics, 54 (2012), pp. 30–41.

[36] Z. J. Grey and P. G. Constantine, *Active subspaces of airfoil shape parameterizations*, AIAA J., 56 (2018), pp. 2003–2017, https://doi.org/10.2514/1.J056054.

[37] A. Griewank and A. Walther, *Evaluating Derivatives: Principles and Techniques of Algorithmic Differentiation*, 2nd ed., Other Titles Appl. Math., SIAM, Philadelphia, 2008, https://doi.org/10.1137/1.9780898717761.

[38] T. Hastie, R. Tibshirani, and J. Friedman, *The Elements of Statistical Learning: Data Mining, Inference, and Prediction*, 2nd ed., Springer, New York, 2009.

[39] C. F. Higham and D. J. Higham, *Deep learning: An introduction for applied mathematicians*, SIAM Rev., to appear.

[40] J. M. Hokanson and P. G. Constantine, *Data-driven polynomial ridge approximation using variable projection*, SIAM J. Sci. Comput., 40 (2018), pp. A1566–A1589, https://doi.org/10.1137/17M1117690.

[41] J. T. Holodnak, I. C. F. Ipsen, and R. C. Smith, *A probabilistic subspace bound with application to active subspaces*, SIAM J. Matrix Anal. Appl., 39 (2018), pp. 1208–1220, https://doi.org/10.1137/17M1141503.

[42] X. Hu, G. T. Parks, X. Chen, and P. Seshadri, *Discovering a one-dimensional active subspace to quantify multidisciplinary uncertainty in satellite system design*, Adv. Space Res., 57 (2016), pp. 1268–1279, https://doi.org/10.1016/j.asr.2015.11.001.

[43] J. M. Hyman and M. Tian, *Accurate integration of high dimensional functions using polynomial detrending*, Technical report, Tulane University, New Orleans, LA, 2014.

[44] J. D. Jakeman and A. Narayan, *Generation and application of multivariate polynomial quadrature rules*, Comput. Methods Appl. Mech. Engrg., 338 (2018), pp. 134–161, https://doi.org/10.1016/j.cma.2018.04.009.

[45] J. L. Jefferson, J. M. Gilbert, P. G. Constantine, and R. M. Maxwell, *Active subspaces for sensitivity analysis and dimension reduction of an integrated hydrologic model*, Comput. Geosci., 83 (2016), pp. 127–138, http://doi.org/10.1016/j.cageo.2015.07.001.

[46] R. King, J. Quick, C. Adcock, and K. Dykes, *Active subspaces for wind plant surrogate modeling*, in 2018 Wind Energy Symposium, Kissimmee, FL, AIAA, Reston, VA, 2018, https://doi.org/10.2514/6.2018-2019.

[47] C. Lanczos, *An iteration method for the solution of the eigenvalue problem of linear differential and integral operators*, J. Res. Nat. Bureau Standards, 45 (1950), pp. 255–282.

[48] O. P. Le Maître and O. M. Knio, *Spectral Methods for Uncertainty Quantification*, Springer, New York, 2010, https://doi.org/10.1007/978-90-481-3520-2.

[49] A. Lewis, R. Smith, and B. Williams, *Gradient free active subspace construction using Morris screening elementary effects*, Comput. Math. Appl., 72 (2016), pp. 1603–1615, https://doi.org/10.1016/j.camwa.2016.07.022.

Downloaded 11/18/19 to 138.195.214.230. Redistribution subject to SIAM license or copyright; see http://www.siam.org/journals/ojsa.php

Copyright © by SIAM. Unauthorized reproduction of this article is prohibited.

[50] B. Li, *Sufficient Dimension Reduction: Methods and Applications in R*, CRC Press, Boca Raton, FL, 2018.
[51] W. Li, G. Lin, and B. Li, *Inverse regression-based uncertainty quantification algorithms for high-dimensional models: Theory and practice*, J. Comput. Phys., 321 (2016), pp. 259–278, https://doi.org/10.1016/j.jcp.2016.05.040.
[52] J. Liesen and Z. Strakos, *Krylov Subspace Methods: Principles and Analysis*, Oxford University Press, Oxford, 2013, https://global.oup.com/academic/product/krylov-subspace-methods-9780199655410.
[53] X. Liu and S. Guillas, *Dimension reduction for Gaussian process emulation: An application to the influence of bathymetry on tsunami heights*, SIAM/ASA J. Uncertain. Quantif., 5 (2017), pp. 787–812, https://doi.org/10.1137/16M1090648.
[54] L. Lovász and M. Simonovits, *Random walks in a convex body and an improved volume algorithm*, Random Structures Algorithms, 4 (1993), pp. 359–412, https://doi.org/10.1002/rsa.3240040402.
[55] T. W. Lukaczyk, P. Constantine, F. Palacios, and J. J. Alonso, *Active subspaces for shape optimization*, in 10th AIAA Multidisciplinary Design Optimization Conference, National Harbor, AIAA, Reston, VA, 2014, pp. 1–18, https://doi.org/10.2514/6.2014-1171.
[56] S. Mayer, T. Ullrich, and J. Vybíral, *Entropy and sampling numbers of classes of ridge functions*, Constr. Approx., 42 (2015), pp. 231–264, http://doi.org/10.1007/s00365-014-9267-x.
[57] E. Novak and K. Ritter, *High dimensional integration of smooth functions over cubes*, Numer. Math., 75 (1996), pp. 79–97, https://doi.org/10.1007/s002110050231.
[58] A. B. Owen, *Monte Carlo Theory, Methods and Examples*, http://statweb.stanford.edu/~owen/mc/, (2013).
[59] A. Pinkus, *Ridge Functions*, Cambridge University Press, Cambridge, 2015, https://doi.org/10.1017/CBO9781316408124.
[60] E. K. Ryu and S. P. Boyd, *Extensions of Gauss quadrature via linear programming*, Found. Comput. Math., 15 (2015), pp. 953–971, https://doi.org/10.1007/s10208-014-9197-9.
[61] P. Seshadri, S. Shahpar, P. Constantine, G. Parks, and M. Adams, *Turbomachinery active subspace performance maps*, J. Turbomach., 140 (2018), 041003, https://doi.org/10.1115/1.4038839.
[62] R. C. Smith, *Uncertainty Quantification: Theory, Implementation, and Applications*, Comput. Sci. Eng., SIAM, Philadelphia, 2013, http://bookstore.siam.org/cs12/.
[63] T. J. Stieltjes, *Quelques recherches sur la théorie des quadratures dites mécaniques*, Ann. Sci. Éc. Norm. Supér. (3), 1 (1884), pp. 409–426, http://eudml.org/doc/80911.
[64] T. J. Sullivan, *Introduction to Uncertainty Quantification*, Springer, New York, 2015, https://doi.org/10.1007/978-3-319-23395-6.
[65] J. F. Traub and A. G. Werschulz, *Complexity and Information*, Cambridge University Press, Cambridge, 1998.
[66] L. N. Trefethen, *Is Gauss quadrature better than Clenshaw–Curtis?*, SIAM Rev., 50 (2008), pp. 67–87, https://doi.org/10.1137/060659831.
[67] L. N. Trefethen, *Approximation Theory and Approximation Practice*, Appl. Math. 128, SIAM, Philadelphia, 2013.
[68] R. Tripathy, I. Bilionis, and M. Gonzalez, *Gaussian processes with built-in dimensionality reduction: Applications to high-dimensional uncertainty propagation*, J. Comput. Phys., 321 (2016), pp. 191–223, https://doi.org/10.1016/j.jcp.2016.05.039.
[69] P. A. Tsilifis, *Gradient-informed basis adaptation for Legendre chaos expansions*, J. Verification, Validation, Uncertain. Quantif., 3 (2018), p. 011005, https://doi.org/10.1115/1.4040802.
[70] H. Tyagi and V. Cevher, *Learning non-parametric basis independent models from point queries via low-rank methods*, Appl. Comput. Harmon. Anal., 37 (2014), pp. 389–412, https://doi.org/10.1016/j.acha.2014.01.002.
[71] D. Xiu and G. E. Karniadakis, *The Wiener–Askey polynomial chaos for stochastic differential equations*, SIAM J. Sci. Comput., 24 (2002), pp. 619–644, https://doi.org/10.1137/S1064827501387826.

Downloaded 11/18/19 to 138.195.214.230. Redistribution subject to SIAM license or copyright; see http://www.siam.org/journals/ojsa.php

Copyright © by SIAM. Unauthorized reproduction of this article is prohibited.