Statistics and Computing
https://doi.org/10.1007/s11222-018-9816-4

Gauss–Christoffel quadrature for inverse regression: applications to computer experiments

Andrew Glaws[1] · **Paul G. Constantine**[1]

Received: 26 October 2017 / Accepted: 29 May 2018

Abstract
Sufficient dimension reduction (SDR) provides a framework for reducing the predictor space dimension in statistical regression problems. We consider SDR in the context of dimension reduction for deterministic functions of several variables such as those arising in computer experiments. In this context, SDR can reveal low-dimensional ridge structure in functions. Two algorithms for SDR—sliced inverse regression (SIR) and sliced average variance estimation (SAVE)—approximate matrices of integrals using a sliced mapping of the response. We interpret this sliced approach as a Riemann sum approximation of the particular integrals arising in each algorithm. We employ the well-known tools from numerical analysis—namely, multivariate numerical integration and orthogonal polynomials—to produce new algorithms that improve upon the Riemann sum-based numerical integration in SIR and SAVE. We call the new algorithms *Lanczos–Stieltjes inverse regression* (LSIR) and *Lanczos–Stieltjes average variance estimation* (LSAVE) due to their connection with Stieltjes' method—and Lanczos' related discretization—for generating a sequence of polynomials that are orthogonal with respect to a given measure. We show that this approach approximates the desired integrals, and we study the behavior of LSIR and LSAVE with two numerical examples. The quadrature-based LSIR and LSAVE eliminate the first-order algebraic convergence rate bottleneck resulting from the Riemann sum approximation, thus enabling high-order numerical approximations of the integrals when appropriate. Moreover, LSIR and LSAVE perform as well as the best-case SIR and SAVE implementations (e.g., adaptive partitioning of the response space) when low-order numerical integration methods (e.g., simple Monte Carlo) are used.

Keywords Sufficient dimension reduction · Sliced inverse regression · Sliced average variance estimation · Orthogonal polynomials

1 Introduction and background

Increases in computational power have enabled the simulation of complex physical processes by models that more accurately reflect reality. Scientific studies that employ such models are often referred to as *computer experiments* (Sacks et al. 1989; Koehler and Owen 1996; Santner et al. 2003). For algorithm development in this context, the computer simulation is represented by a deterministic function that maps the simulation inputs to outputs of interest. These functions rarely have closed-form expressions and are often expensive to evaluate. To enable computer experiments with expensive models, we may construct a cheaper surrogate—e.g., a response surface approximation (Myers and Montgomery 1995; Jones 2001)—that can be evaluated quickly. However, building a surrogate suffers from *the curse of dimensionality* (Traub and Werschulz 1998; Donoho 2000); loosely speaking, the number of model evaluations required to achieve a prescribed accuracy grows exponentially with the input space dimension.

One approach to combat the curse of dimensionality is to reduce the dimension of the input space. For example,

Glaws' work is supported by the Ben L. Fryrear Ph.D. Fellowship in Computational Science at the Colorado School of Mines and the Department of Defense, Defense Advanced Research Projects Agency's program Enabling Quantification of Uncertainty in Physical Systems. Constantine's work is supported by the US Department of Energy Office of Science, Office of Advanced Scientific Computing Research, Applied Mathematics program under Award Number DE-SC-0011077.

✉ Andrew Glaws
andrew.glaws@colorado.edu

Paul G. Constantine
paul.constantine@colorado.edu

[1] Department of Computer Science, University of Colorado Boulder, Boulder, CO, USA

Published online: 07 June 2018

when the inputs are correlated, then principal component analysis (Jolliffe 2002) can identify a set of fewer uncorrelated variables that linearly map to the correlated simulation inputs. When the inputs are independent, one may use global sensitivity analysis (Saltelli et al. 2008) to identify a subset of inputs whose variability does not significantly change the outputs; these inputs can be fixed at nominal values, thus reducing the input dimension. Active subspaces (Constantine 2015) generalize the coordinate-based global sensitivity analysis to important directions in the input space; perturbing inputs along the orthogonal complement of the active subspace changes the outputs relatively little, on average.

In the related context of statistical regression, sufficient dimension reduction (SDR) provides a theoretical framework for subspace-based dimension reduction in the predictor space relative to the regression response (Cook 1998; Ma and Zhu 2013); SDR approaches have also been used as heuristics for dimension reduction in deterministic functions arising in computer experiments (Cook 1994; Li et al. 2016; Glaws et al. 2018). There are several methods for SDR including ordinary least squares (OLS) (Li and Duan 1989), principal Hessian directions (pHd) (Li 1992), among others (Cook and Ni 2005; Li and Wang 2007; Cook and Forzani 2009). In this paper, we consider two well-known methods for SDR—sliced inverse regression (SIR) (Li 1991) and sliced average variance estimation (SAVE) (Cook and Weisberg 1991). Although SIR and SAVE are two of the earlier methods developed for SDR, they remain popular today due to their effectiveness and ease of implementation. Moreover, many alternative methods for SDR are variations of SIR and SAVE in some way.

SIR and SAVE each approximate a specific matrix of expectations by *slicing* the response space based on a given data set of predictor/response pairs. If we consider applying SIR and SAVE for dimension reduction in deterministic functions, these expectations become Lebesgue integrals over the function's range (i.e., the space of simulation outputs). Assuming the output distributions are sufficiently smooth, the integrals can be written as Riemann integrals against the push-forward density induced by the function and the distribution on the input space. Then the slicing from SIR and SAVE can be interpreted as a Riemann sum approximation of these integrals (Davis and Rabinowitz 1984). The approximation accuracy of SIR and SAVE depends on the number of terms in the Riemann sum—i.e., the number of slices. The Riemann sum approximation acts as an accuracy bottleneck for these algorithms; obtaining additional data or using a better design on the input space cannot maximally improve the approximation. We introduce new algorithms—Lanczos–Stieltjes inverse regression (LSIR) and Lanczos–Stieltjes average variance estimation (LSAVE)—that improve the accuracy and convergence rates compared to slicing by employing high-order Gauss–Christoffel quadrature to approximate the integrals with respect to the output. The new algorithms eliminate the bottleneck due to Riemann sums and place the burden of accuracy on the input space design. Thus, LSIR and LSAVE enable the use of higher-order numerical integration rules on the input space such as sparse grids or tensor product quadrature when such approaches are appropriate.

This work contains two main contributions: First, by characterizing the matrices of expectations from SIR and SAVE as integrals in the context of dimension reduction for deterministic functions, we interpret the slice-based methods as Riemann sum approximations of these integral. We recognize that this approach produces an accuracy bottleneck on the approximation of the matrices of expectations underlying SIR and SAVE. Second, we develop high-order quadrature methods for these integrals that remove this accuracy bottleneck imposed by the Riemann sums on the output space. When the input space dimension is sufficiently small and the function of interest is sufficiently smooth so that high-order numerical integration is justified on the input space, our new LSIR and LSAVE methods produce exponentially converging estimates of the matrices of expectations used for subspace-based dimension reduction—compared to the maximal first-order algebraic convergence rate resulting from the slicing. When low-order, dimension-independent numerical integration methods (e.g., simple Monte Carlo) are more appropriate due to a high input space dimension, the LSIR and LSAVE methods perform as well as the best slice-based approaches (e.g., adaptive partitioning of the output space).

The remainder of this paper is structured as follows: In Sect. 2, we review SDR theory and the SIR and SAVE algorithms. We interpret these methods in the context of computer experiments in Sect. 3. In Sect. 4, we review key elements of classical numerical analysis—specifically Gauss–Christoffel quadrature and orthogonal polynomials—that we employ in the new Lanczos–Stieltjes algorithms. We introduce and discuss the new algorithms in Sect. 5. Finally, in Sect. 6, we numerically study various aspects of the newly proposed algorithms on several test problems and compare the results to those from the traditional SIR and SAVE algorithms.

Throughout this paper we use the following notational conventions unless otherwise noted. Bold uppercase letters represent matrices, while bold lowercase letters are vectors. The elements of vectors are denoted using parentheses and subscript indices. For example, $(\boldsymbol{A})_{i,j}$ refers to the ijth element of the matrix $\boldsymbol{A}$ and $(\mathbf{v})_i$ refers to the ith element of the vector $\mathbf{v}$. Also, we use zero-based indexing throughout this paper. We do this because matrix and vector elements will often be associated with polynomials of increasing degree, starting with degree zero (i.e., constant) polynomials. We make use of a variety of probability measures, which we denote by π with the appropriate subscripts to indicate their domain.

2 Inverse regression methods

Sufficient dimension reduction (SDR) enables dimension reduction in regression problems. In this section, we provide a brief overview of SDR and two methods for its implementation. More detailed discussions of SDR can be found in Cook (1998) and Ma and Zhu (2013).

The regression problem considers a given set of predictor/response pairs, denoted by

$$\{[\,\mathbf{x}_i^\top\,,\ y_i\,]\},\quad i=0,\ldots,N-1, \tag{1}$$

where $\mathbf{x}_i \in \mathbb{R}^m$ is the vector-valued predictor and $y_i \in \mathbb{R}$ is the scalar-valued response. These pairs are assumed to be drawn according to an unknown joint distribution $\pi_{\mathbf{x},y}$. The goal of regression is to approximate statistical properties (e.g., the CDF, expectation, variance) of the conditional random variable $y|\mathbf{x}$ using the predictor/response pairs. However, such characterization of $y|\mathbf{x}$ becomes difficult when the dimension of the predictor space m becomes large.

SDR seeks out a low-dimensional linear transform of the predictors under which the statistical behavior of $y|\mathbf{x}$ is unchanged. Consider $\boldsymbol{A} \in \mathbb{R}^{m\times n}$ with $n < m$. If

$$y \perp\!\!\!\perp \mathbf{x}\,|\,\boldsymbol{A}^\top\mathbf{x}, \tag{2}$$

then, $y|\mathbf{x} \sim y|\boldsymbol{A}^\top\mathbf{x}$ (Cook 1998, Ch. 6). Equation (2)—referred to as *conditional independence*—states that if the reduced predictors $\boldsymbol{A}^\top\mathbf{x}$ are known, then the response and the original predictors are independent. Under conditional independence, $y|\mathbf{x}$ and $y|\boldsymbol{A}^\top\mathbf{x}$ have the same distribution. Furthermore, any $\boldsymbol{A}$ that satisfies (2) is a basis for sufficient dimension reduction. Methods for SDR attempt to compute $\boldsymbol{A}$ for a given regression problem using the predictor/response pairs $\{[\,\mathbf{x}_i^\top\,,\ y_i\,]\}$, $i=0,\ldots,N-1$.

We consider two methods for SDR—sliced inverse regression (SIR) (Li 1991) and sliced average variance estimation (SAVE) (Cook and Weisberg 1991). These methods fall into a class of techniques known as *inverse regression methods*, named as such because they search for dimension reduction in $y|\mathbf{x}$ by studying the *inverse regression* $\mathbf{x}|y$. The inverse regression is an m-dimensional conditional random variable parameterized by the scalar-valued response. This replaces the single m-dimensional problem with m one-dimensional problems. A detailed discussion on inverse regression can be found in Adragni and Cook (2009). We focus on the SIR and SAVE methods.

Before describing the SIR and SAVE algorithms, we introduce the assumption of *standardized predictors*,

$$\mathbb{E}[\mathbf{x}] = \mathbf{0} \quad\text{and}\quad \mathbb{C}\mathrm{ov}[\mathbf{x}] = \boldsymbol{I}. \tag{3}$$

This assumption simplifies discussion and does not restrict the SDR problem (Cook 1998, Ch. 11). We assume (3) holds for the remainder of this section.

The SIR and SAVE algorithms approximate population matrices constructed from statistical characteristics of $\mathbf{x}|y$. The matrix underlying the SIR algorithm is

$$\boldsymbol{C}_{\mathrm{IR}} = \mathbb{C}\mathrm{ov}\left[\mathbb{E}[\mathbf{x}|y]\right]. \tag{4}$$

We denote this matrix by $\boldsymbol{C}_{\mathrm{IR}}$ because it does not yet include slicing (i.e., the "S" in SIR). We introduce the slicing later in this section. The matrix $\boldsymbol{C}_{\mathrm{IR}}$ is the covariance of the *inverse regression function* $\mathbb{E}[\mathbf{x}|y]$. This function draws out a curve through m-dimensional space that is parameterized by the scalar-valued response y. Under certain conditions (Li 1991), the column space of $\boldsymbol{C}_{\mathrm{IR}}$ can be used to approximate $\boldsymbol{A}$ from (2). This matrix is known to perform well for regressions exhibiting strong linear trends, but can struggle on problems where the response is symmetric about the origin.

SAVE is based on the population matrix

$$\boldsymbol{C}_{\mathrm{AVE}} = \mathbb{E}\left[(\boldsymbol{I}-\mathbb{C}\mathrm{ov}[\mathbf{x}|y])^2\right]. \tag{5}$$

Similar to (4), we drop the "S" in the notation as this matrix does not include slicing. We see the motivation for this matrix by recognizing that, under (3),

$$\begin{aligned}\mathbb{E}[\mathbb{C}\mathrm{ov}[\mathbf{x}|y]] &= \mathbb{C}\mathrm{ov}[\mathbf{x}] - \mathbb{C}\mathrm{ov}[\mathbb{E}[\mathbf{x}|y]]\\ &= \boldsymbol{I} - \mathbb{C}\mathrm{ov}[\mathbb{E}[\mathbf{x}|y]],\end{aligned} \tag{6}$$

which implies that $\mathbb{C}\mathrm{ov}[\mathbb{E}[\mathbf{x}|y]] = \mathbb{E}[\boldsymbol{I} - \mathbb{C}\mathrm{ov}[\mathbf{x}|y]]$. The square of $\boldsymbol{I} - \mathbb{C}\mathrm{ov}[\mathbf{x}|y]$ is used to ensure that the eigenvalues of the resulting matrix are nonnegative (Cook and Weisberg 1991). By using higher moments of the inverse regression, this method overcomes the issues with symmetry in SIR. However, in practice accurately estimating these higher moments can be difficult, especially when the number of predictor/response pairs is small.

The regression problem begins with predictor/response pairs $\{[\,\mathbf{x}_i^\top\,,\ y_i\,]\}$, $i=0,\ldots,N-1$. We want to use these data to approximate $\boldsymbol{C}_{\mathrm{IR}}$ and $\boldsymbol{C}_{\mathrm{AVE}}$. This is difficult due to the conditional expectations in these matrices. The SIR and SAVE algorithms use a slicing approach to enable numerical approximation of (4) and (5). Let J_r for $r=0,\ldots,R-1$ denote R intervals that partition the range of response values into slices. Define the sliced mapping of the response

$$h(y) = r \quad\text{where}\quad y \in J_r. \tag{7}$$

The SIR and SAVE algorithms consider dimension reduction relative to the conditional random variable $h(y)|\mathbf{x}$ instead of

$y|\mathbf{x}$. That is, these methods approximate the matrices

$$\begin{aligned} \boldsymbol{C}_{\text{SIR}} &= \mathbb{C}\text{ov}\left[\mathbb{E}\left[\mathbf{x}|h(y)\right]\right], \\ \boldsymbol{C}_{\text{SAVE}} &= \mathbb{E}\left[\left(\boldsymbol{I} - \mathbb{C}\text{ov}\left[\mathbf{x}|h(y)\right]\right)^2\right], \end{aligned} \tag{8}$$

respectively, where the notation $\boldsymbol{C}_{\text{SIR}}$ and $\boldsymbol{C}_{\text{SAVE}}$ indicate that they are defined with respect to the sliced response $h(y)$.

Algorithm 1 provides an outline for estimating $\boldsymbol{C}_{\text{SIR}}$ by computing a sample estimate $\hat{\boldsymbol{C}}_{\text{SIR}}$ from the given predictor/response pairs; Algorithm 2 shows the same for SAVE. In practice, the eigendecomposition of the sample estimates $\hat{\boldsymbol{C}}_{\text{SIR}}$ and $\hat{\boldsymbol{C}}_{\text{SAVE}}$ leads to an approximation of the basis $\boldsymbol{A}$ from (2). We focus not on the eigenspaces, but on how we can interpret the outputs from the SIR and SAVE algorithms as approximations of $\boldsymbol{C}_{\text{IR}}$ and $\boldsymbol{C}_{\text{AVE}}$ from (4) and (5), respectively.

Algorithm 1 Sliced inverse regression (SIR)

Given: Predictor/response pairs $\{[\,\mathbf{x}_i^\top,\ y_i\,]\}$, $i = 0, \ldots, N-1$ drawn according to $\pi_{\mathbf{x},y}$.
Assumptions: The predictors are standardized such that (3) is satisfied.

1. Define a sliced partition of the observed response space, J_r for $r = 0, \ldots, R-1$. Let $\mathcal{I}_r \subset \{0, \ldots, N-1\}$ be indices i for which $y_i \in J_r$ and $N_r = |\mathcal{I}_r|$.
2. For $r = 0, \ldots, R-1$, compute the in-slice sample expectation

$$\hat{\boldsymbol{\mu}}(r) = \frac{1}{N_r} \sum_{i \in \mathcal{I}_r} \mathbf{x}_i. \tag{9}$$

3. Compute the sample matrix

$$\hat{\boldsymbol{C}}_{\text{SIR}} = \frac{1}{N} \sum_{r=0}^{R-1} N_r\, \hat{\boldsymbol{\mu}}(r)\, \hat{\boldsymbol{\mu}}(r)^\top. \tag{10}$$

Output: The matrix $\hat{\boldsymbol{C}}_{\text{SIR}}$.

The sliced mapping h enables the computation of the sample estimates

$$\hat{\boldsymbol{\mu}}(r) \approx \mathbb{E}\left[\mathbf{x}|h(y)\right] \quad \text{and} \quad \hat{\boldsymbol{\Sigma}}(r) \approx \mathbb{C}\text{ov}\left[\mathbf{x}|h(y)\right] \tag{14}$$

by binning the response data. The approximations in (14) depend on the number of predictor/response pairs N. The sample estimates of $\boldsymbol{C}_{\text{SIR}}$ and $\boldsymbol{C}_{\text{SAVE}}$ have been shown to be $N^{-1/2}$ consistent (Li 1991; Cook 2000). Accuracy of $\boldsymbol{C}_{\text{SIR}}$ and $\boldsymbol{C}_{\text{SAVE}}$ in approximating $\boldsymbol{C}_{\text{IR}}$ and $\boldsymbol{C}_{\text{AVE}}$, respectively, depends on the slicing applied to the response space. Defining the slices J_r, $r = 0, \ldots, R-1$, such that they each contain approximately equal numbers of samples can improve the approximation by balancing accuracy in the sample estimates (14) across all slices (Li 1991). By increasing the number R of slices, we increase accuracy due to slicing, but may reduce accuracy of (14) by decreasing the number of samples

Algorithm 2 Sliced average variance estimation (SAVE)

Given: Predictor/response pairs $\{[\,\mathbf{x}_i^\top,\ y_i\,]\}$, $i = 0, \ldots, N-1$ drawn according to $\pi_{\mathbf{x},y}$.
Assumptions: The predictors are standardized such that (3) is satisfied.

1. Define a sliced partition of the observed response space, J_r for $r = 0, \ldots, R-1$. Let $\mathcal{I}_r \subset \{0, \ldots, N-1\}$ be indices i for which $y_i \in J_r$ and $N_r = |\mathcal{I}_r|$.
2. For $r = 0, \ldots, R-1$,
   (i) Compute the in-slice sample expectation

$$\hat{\boldsymbol{\mu}}(r) = \frac{1}{N_r} \sum_{i \in \mathcal{I}_r} \mathbf{x}_i. \tag{11}$$

   (ii) Compute the in-slice sample covariance

$$\hat{\boldsymbol{\Sigma}}(r) = \frac{1}{N_r - 1} \sum_{i \in \mathcal{I}_r} \left(\mathbf{x}_i - \hat{\boldsymbol{\mu}}(r)\right)\left(\mathbf{x}_i - \hat{\boldsymbol{\mu}}(r)\right)^\top. \tag{12}$$

3. Compute the sample matrix

$$\hat{\boldsymbol{C}}_{\text{SAVE}} = \frac{1}{N} \sum_{r=0}^{R-1} N_r \left(\boldsymbol{I} - \hat{\boldsymbol{\Sigma}}(r)\right)^2. \tag{13}$$

Output: The matrix $\hat{\boldsymbol{C}}_{\text{SAVE}}$.

within each slice (Cook 1998, Ch. 6). We discuss the idea of convergence in terms of the number of slices in more detail in Sect. 3. Generally, the slicing approach in Algorithms 1 and 2 is justified by properties of SDR that relate dimension reduction of $h(y)|\mathbf{x}$ to dimension reduction of $y|\mathbf{x}$ (Cook 1998, Ch. 6).

3 SIR and SAVE for deterministic functions

In this section, we consider the application of SIR and SAVE to deterministic functions, such as those arising in computer experiments. Computer experiments employ models of physical phenomena, which we represent as deterministic functions that map m simulation inputs to a scalar-valued output (Sacks et al. 1989; Koehler and Owen 1996; Santner et al. 2003). We write this as

$$y = f(\mathbf{x}), \quad \mathbf{x} \in \mathcal{X} \subseteq \mathbb{R}^m, \quad y \in \mathcal{F} \subseteq \mathbb{R}, \tag{15}$$

where $f : \mathcal{X} \to \mathcal{F}$ is measurable and the random variables $\mathbf{x}$, y are the function inputs and output, respectively. Let $\pi_{\mathbf{x}}$ denote the probability measure induced by $\mathbf{x}$ over $\mathcal{X}$. Unlike the joint distribution $\pi_{\mathbf{x},y}$, this measure is assumed to be known. Furthermore, we may choose the values of $\mathbf{x}$ at which to evaluate (15) according to $\pi_{\mathbf{x}}$. Define the probability measure induced by y over $\mathcal{F}$ by π_y. This measure is fully determined by $\pi_{\mathbf{x}}$ and f, but its form is assumed to be unknown. We can estimate π_y using point evaluations of f.

Remark 1 In the context of deterministic functions, SDR may be viewed as a method for ridge recovery (Glaws et al. 2018). That is, it uses point queries of $f(\mathbf{x})$ to search for $\boldsymbol{A} \in \mathbb{R}^{m\times n}$ with $n < m$ assuming that

$$y = f(\mathbf{x}) \;=\; g(\boldsymbol{A}^\top \mathbf{x}) \tag{16}$$

for some $g : \mathbb{R}^n \to \mathbb{R}$. When (16) holds for some g and $\boldsymbol{A}$, $f(\mathbf{x})$ is called a *ridge function* (Pinkus 2015).

In Sect. 5, we introduce new algorithms for estimating $\boldsymbol{C}_{\text{IR}}$ and $\boldsymbol{C}_{\text{AVE}}$. First, we must provide context for applying these methods to deterministic functions. We start by introducing an assumption on the input space $\mathcal{X}$.

Assumption 1 The inputs have finite fourth moments and are standardized such that

$$\int \mathbf{x}\, \mathrm{d}\pi_{\mathbf{x}}(\mathbf{x}) \;=\; \mathbf{0} \quad \text{and} \quad \int \mathbf{x}\,\mathbf{x}^\top \mathrm{d}\pi_{\mathbf{x}}(\mathbf{x}) \;=\; \boldsymbol{I}. \tag{17}$$

Assumption 1 is similar to the assumption of standardized predictors in (3). However, it also assumes finite fourth moments of the predictors. This assumption is important for the new algorithms introduced in Sect. 5. We assume Assumption 1 holds throughout.

Recall that SIR and SAVE are based on the inverse regression $\mathbf{x}|y$. For deterministic functions, this is the inverse image of the function f,

$$f^{-1}(y) \;=\; \{\mathbf{x} \in \mathcal{X} : f(\mathbf{x}) \;=\; y\}, \tag{18}$$

and we define the conditional measure $\pi_{\mathbf{x}|y}$ as the restriction of $\pi_{\mathbf{x}}$ to f^{-1} through disintegration (Chang and Pollard 1997). We denote this measure by $\pi_{\mathbf{x}|y}$ to emphasize its connection to the inverse regression $\mathbf{x}|y$. Since the inverse regression methods for deterministic functions operate on the inverse image $f^{-1}(y)$, these methods do not require that the function f be uniquely invertible.

The matrices $\boldsymbol{C}_{\text{IR}}$ and $\boldsymbol{C}_{\text{AVE}}$ contain the conditional expectation and the conditional covariance of the inverse regression (see (4) and (5), respectively). These statistical quantities can be expressed as integrals with respect to the conditional measure $\pi_{\mathbf{x}|y}$,

$$\begin{aligned} \boldsymbol{\mu}(y) \;&=\; \int \mathbf{x}\, \mathrm{d}\pi_{\mathbf{x}|y}(\mathbf{x}), \\ \boldsymbol{\Sigma}(y) \;&=\; \int (\mathbf{x} - \boldsymbol{\mu}(y))\,(\mathbf{x} - \boldsymbol{\mu}(y))^\top \,\mathrm{d}\pi_{\mathbf{x}|y}(\mathbf{x}). \end{aligned} \tag{19}$$

We denote the conditional expectation and covariance by $\boldsymbol{\mu}(y)$ and $\boldsymbol{\Sigma}(y)$, respectively, to emphasize that these quantities are functions of the output. Using (19), we can express the $\boldsymbol{C}_{\text{IR}}$ and $\boldsymbol{C}_{\text{AVE}}$ matrices as integrals over the output space $\mathcal{F}$,

$$\begin{aligned} \boldsymbol{C}_{\text{IR}} \;&=\; \int \boldsymbol{\mu}(y)\,\boldsymbol{\mu}(y)^\top \,\mathrm{d}\pi_y(y), \\ \boldsymbol{C}_{\text{AVE}} \;&=\; \int (\boldsymbol{I} - \boldsymbol{\Sigma}(y))^2 \,\mathrm{d}\pi_y(y). \end{aligned} \tag{20}$$

The integrals in (19) are difficult to approximate due to the complex structure of $\pi_{\mathbf{x}|y}$. Algorithms 1 and 2 slice the range of output values to enable this approximation. Let $h(y)$ denote the slicing function from (7) and define the probability mass function

$$\omega(r) \;=\; \int_{J_r} \mathrm{d}\pi_y(y), \qquad r \;=\; 0, \ldots, R-1, \tag{21}$$

where J_r is the rth interval over the range of outputs. We can then express the slice-based matrices $\boldsymbol{C}_{\text{SIR}}$ and $\boldsymbol{C}_{\text{SAVE}}$ as sums over the R slices,

$$\begin{aligned} \boldsymbol{C}_{\text{SIR}} \;&=\; \sum_{r=0}^{R-1} \omega(r)\,\boldsymbol{\mu}(r)\,\boldsymbol{\mu}(r)^\top, \\ \boldsymbol{C}_{\text{SAVE}} \;&=\; \sum_{r=0}^{R-1} \omega(r)\;(\boldsymbol{I} - \boldsymbol{\Sigma}(r))^2, \end{aligned} \tag{22}$$

where $\boldsymbol{\mu}(r)$ and $\boldsymbol{\Sigma}(r)$ are as in (19), but applied to the sliced output $h(y) = r$.

This provides a proper interpretation for the application of Algorithms 1 and 2 to (15). We can construct a set of input/output pairs $\{[\,\mathbf{x}_i^\top,\; y_i\,]\}$, $i \;=\; 0, \ldots, N-1$, by randomly sampling the input space $\mathcal{X}$ according to $\pi_{\mathbf{x}}$ and evaluating $y_i = f(\mathbf{x}_i)$. SIR and SAVE then produce numerical approximations $\hat{\boldsymbol{C}}_{\text{SIR}}$ and $\hat{\boldsymbol{C}}_{\text{SAVE}}$ to $\boldsymbol{C}_{\text{SIR}}$ and $\boldsymbol{C}_{\text{SAVE}}$, respectively.

This exercise of expressing the various elements of SIR and SAVE as integrals also emphasizes the two levels of approximation occurring in these algorithms: (i) approximation in terms of the number of samples N and (ii) approximation in terms of the number of slices R. That is,

$$\begin{aligned} \hat{\boldsymbol{C}}_{\text{SIR}} \;&\approx\; \boldsymbol{C}_{\text{SIR}} \;\approx\; \boldsymbol{C}_{\text{IR}}, \\ \hat{\boldsymbol{C}}_{\text{SAVE}} \;&\approx\; \boldsymbol{C}_{\text{SAVE}} \;\approx\; \boldsymbol{C}_{\text{AVE}}. \end{aligned} \tag{23}$$

As mentioned in Sect. 2, approximation due to sampling (i.e., the leftmost approximations in (23)) has been shown to be $N^{-1/2}$ consistent (Li 1991; Cook 2000). From the integral perspective, the slicing approach can be interpreted as a Riemann sum approximation of the integrals in (20). Riemann sum approximations estimate integrals by a finite sum. These approximations converge like R^{-1} for continuous functions

on compact domains, where R denotes the number of terms in the Riemann sum (Davis and Rabinowitz 1984, Ch. 2).

In the next section, we introduce several numerical tools and link them to the various elements of the SIR and SAVE algorithms discussed so far. Ultimately, we use these tools, including orthogonal polynomials and numerical quadrature, in the proposed algorithms to enable approximation of $\boldsymbol{C}_{\text{IR}}$ and $\boldsymbol{C}_{\text{AVE}}$ without using Riemann sums.

4 Orthogonal polynomials and Gauss–Christoffel quadrature

Orthogonal polynomials and numerical quadrature are fundamental to numerical analysis and have been studied extensively; detailed discussions are available in Liesen and Strakoš (2013), Gautschi (2004), Meurant (2006), Golub and Meurant (2010). Our discussion is based on these references, reviewing key concepts necessary to develop the new algorithms for approximating the matrices in (20). We begin with the Stieltjes procedure for constructing orthonormal polynomials with respect to a given measure. We then relate this procedure to Gauss–Christoffel quadrature, polynomial expansions, and the Lanczos algorithm.

4.1 The Stieltjes procedure

The Stieltjes procedure recursively constructs a sequence of polynomials that are orthonormal with respect to a given measure (Liesen and Strakoš 2013, Ch. 3). Let π denote a given probability measure with sample space $\mathbb{R}$, and let $\phi, \psi : \mathbb{R} \to \mathbb{R}$ be two scalar-valued functions that are square-integrable with respect to π. The continuous inner product relative to π is

$$(\phi, \psi)_\pi = \int \phi(y)\, \psi(y)\, \mathrm{d}\pi(y), \tag{24}$$

and the induced norm is $||\phi||_\pi = \sqrt{(\phi, \phi)_\pi}$. A sequence of polynomials $\{\phi_0, \phi_1, \phi_2, \dots\}$ is orthonormal with respect to π if

$$(\phi_i, \phi_j)_\pi = \delta_{i,j}, \quad i, j = 0, 1, 2, \dots, \tag{25}$$

where $\delta_{i,j}$ is the Kronecker delta. Algorithm 3 contains a method for constructing a sequence of orthonormal polynomials $\{\phi_0, \phi_1, \phi_2, \dots\}$ relative to π. This algorithm is known as the Stieltjes procedure and was first introduced in Stieltjes (1884).

The last step of Algorithm 3 defines the three-term recurrence relationship for orthonormal polynomials,

$$\beta_{i+1}\phi_{i+1}(y) = (y - \alpha_i)\, \phi_i(y) - \beta_i \phi_{i-1}(y), \tag{26}$$

Algorithm 3 Stieltjes procedure (Gautschi 2004, Section 2.2.3.1)

Given: The probability measure π.
Assumptions: Let $\phi_{-1}(y) = 0$ and $\tilde{\phi}_0(y) = 1$.

1. For $i = 0, 1, 2, \dots$
   (i) $\beta_i = ||\tilde{\phi}_i(y)||_\pi$
   (ii) $\phi_i(y) = \tilde{\phi}_i(y) \,/\, \beta_i$
   (iii) $\alpha_i = (y\, \phi_i(y),\, \phi_i(y))_\pi$
   (iv) $\tilde{\phi}_{i+1}(y) = (y - \alpha_i)\, \phi_i(y) - \beta_i\, \phi_{i-1}(y)$

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

for $i = 0, 1, 2, \dots$. Any sequence of polynomials that satisfies (26) is orthonormal with respect to the given measure. If we consider the first k terms, then we can rearrange it to obtain

$$y\, \phi_i(y) = \beta_i \phi_{i-1}(y) + \alpha_i \phi_i(y) + \beta_{i+1}\phi_{i+1}(y), \tag{27}$$

for $i = 0, 1, 2, \dots, k-1$. Let

$$\boldsymbol{\phi}(y) = [\, \phi_0(y)\,,\ \phi_1(y)\,,\ \dots\,,\ \phi_{k-1}(y)\,]^\top. \tag{28}$$

We can then write (27) in matrix form as

$$y\, \boldsymbol{\phi}(y) = \boldsymbol{J}\, \boldsymbol{\phi}(y) + \beta_k\, \phi_k(y)\, \mathbf{e}_k, \tag{29}$$

where $\mathbf{e}_k \in \mathbb{R}^k$ is the vector of zeros with a one in the kth entry and $\boldsymbol{J} \in \mathbb{R}^{k \times k}$ is

$$\boldsymbol{J} = \begin{bmatrix} \alpha_0 & \beta_1 & & & \\ \beta_1 & \alpha_1 & \beta_2 & & \\ & \ddots & \ddots & \ddots & \\ & & \beta_{k-2} & \alpha_{k-2} & \beta_{k-1} \\ & & & \beta_{k-1} & \alpha_{k-1} \end{bmatrix}, \tag{30}$$

where α_i, β_i are the recurrence coefficients from Algorithm 3. This matrix—known as the Jacobi matrix—is symmetric and tridiagonal. Let the eigendecomposition of $\boldsymbol{J}$ be

$$\boldsymbol{J} = \boldsymbol{Q}\boldsymbol{\Lambda}\boldsymbol{Q}^\top. \tag{31}$$

From (29), the eigenvalues of $\boldsymbol{J}$, denoted by λ_i, $i = 0, \dots, k-1$, are the zeros of the degree-k polynomial $\phi_k(y)$. Furthermore, the eigenvector associated with λ_i is $\boldsymbol{\phi}(\lambda_i)$. We assume the eigenvectors of $\boldsymbol{J}$ are normalized such that $\boldsymbol{Q}$ is an orthogonal matrix with entries

$$(\boldsymbol{Q})_{i,j} = \frac{\phi_i(\lambda_j)}{||\boldsymbol{\phi}(\lambda_j)||_2}, \quad i, j = 0, \dots, k-1, \tag{32}$$

where $||\cdot||_2$ is the vector 2-norm.

We end this section with brief a note about Fourier expansion of functions in terms of orthonormal polynomials. If a given function $g(y)$ is square-integrable with respect to π, then g admits a mean-squared convergent Fourier series in terms of the orthonormal polynomials,

$$g(y) \;=\; \sum_{i=0}^{\infty} g_i\, \phi_i(y), \tag{33}$$

for y in the support of π and where equality is in the $L_2(\pi)$ sense. By orthogonality of the polynomials, the Fourier coefficients g_i are

$$g_i \;=\; (g, \phi_i)_\pi\,. \tag{34}$$

This polynomial approximation plays an important role in the algorithms introduced in Sect. 5.

4.2 Gauss–Christoffel quadrature

We next discuss the Gauss–Christoffel quadrature for numerical integration and show its connection to orthonormal polynomials (Liesen and Strakoš 2013, Ch. 3). Given a measure π and an integrable function $g(y)$, a k-point quadrature rule approximates the integral of g with respect to π by a weighted sum of g evaluated at k input values,

$$\int g(y)\,\mathrm{d}\pi(y) \;=\; \sum_{i=0}^{k-1} \omega_i\, g(\lambda_i) + r_k. \tag{35}$$

The λ_i's are the quadrature nodes, and the ω_i's are the associated quadrature weights. The k-point quadrature approximation error is contained in the residual term r_k. We can minimize $|r_k|$ by choosing the quadrature nodes and weights appropriately. The nodes and weights of the Gauss–Christoffel quadrature maximize the polynomial *degree of exactness*, which refers to the highest degree polynomial that the quadrature rule exactly integrates (i.e., $r_k = 0$). The k-point Gauss–Christoffel quadrature rule has polynomial degree of exactness $2k-1$. Furthermore, the Gauss–Christoffel quadrature has been shown to converge exponentially at a rate $\mathcal{O}(\rho^{-k})$ for integrals defined on a compact domain when the integrand is analytic (Trefethen 2013, Ch. 19). The base $\rho > 1$ relates to the size of the function's domain of analytical continuability. For functions with $p-1$ continuous derivatives on a compact domain, the Gauss–Christoffel quadrature converges like $\mathcal{O}(k^{-(2p+1)})$.

The Gauss–Christoffel quadrature nodes and weights depend on the given measure π. They can be obtained through the eigendecomposition of $\boldsymbol{J}$ (Golub and Welsch 1969). Recall from (30) that $\boldsymbol{J}$ is the matrix of recurrence coefficients resulting from k steps of the Stieltjes procedure. The eigenvalues of $\boldsymbol{J}$ are the zeros of the k-degree orthonormal polynomial $\phi_k(y)$. These zeros are the nodes of the k-point Gauss–Christoffel quadrature rule with respect to π (i.e., $\phi_k(\lambda_i) = 0$ for $i = 0, \ldots, k-1$). The associated weights are the squares of the first entry of each normalized eigenvector,

$$\omega_i \;=\; (\boldsymbol{Q})^2_{0,i} \;=\; \frac{1}{||\boldsymbol{\phi}(\lambda_i)||_2^2}, \quad i = 0, \ldots, k-1. \tag{36}$$

The Stieltjes procedure employs the inner product from (24) to define the recurrence coefficients α_i, β_i. In the next section, we consider the Lanczos algorithm and explore conditions under which it be may considered a discrete analog to the Stieltjes procedure in Sect. 4.1. Before we make this connection, we define the discrete inner product. Consider an N-point numerical integration rule with respect to π with nodes λ_i and weights ω_i for $i = 0, \ldots, N-1$. For example, these could be the Gauss–Christoffel quadrature nodes and weights, or we could use Monte Carlo as a randomized numerical integration method, where the λ_i's are drawn randomly according to π and $\omega_i = 1/N$ for all i. The discrete inner product is the numerical approximation of the continuous inner product,

$$(\phi, \psi)_{\pi^{(N)}} \;=\; \sum_{i=0}^{N-1} \omega_i\, \phi(\lambda_i)\, \psi(\lambda_i) \;\approx\; (\phi, \psi)_\pi\,, \tag{37}$$

provided that the weights ω_i are all positive. The "N" in $\pi^{(N)}$ denotes the number of points in the numerical integration rule used. The discrete norm is $||\phi||_{\pi^{(N)}} = \sqrt{(\phi, \phi)_{\pi^{(N)}}}$.

The weights ω_i are guaranteed to be positive for Gauss–Christoffel quadrature and Monte Carlo, regardless of dimension. This is not the case for other numerical integration methods such as sparse grids. However, sparse grid integration can still be used to define a discrete inner product as long as care is taken to combine function evaluations appropriately, see Constantine et al. (2012) for details. For simplicity, we only consider discrete inner products defined by either Gauss–Christoffel quadrature or Monte Carlo in this paper.

The pseudospectral expansion approximates the Fourier expansion from (33) by truncating the series after k terms and approximating the Fourier coefficients in (34) using the discrete inner product (Constantine et al. 2012). We write this series for a given square-integrable function $g(y)$ as

$$g(y) \;\approx\; \hat{g}(y) \;=\; \sum_{i=0}^{k-1} \hat{g}_i\, \phi_i(y), \tag{38}$$

where the pseudospectral coefficients are

$$\hat{g}_i = (g, \phi_i)_\pi^{(N)}. \tag{39}$$

Note that the approximation of $g(y)$ by $\hat{g}(y)$ depends on two factors: (i) the approximation accuracy of the coefficients included in the pseudospectral expansion $\hat{g}_i, i = 0, \ldots, k-1$ and (ii) the magnitude of the coefficients in the omitted terms $g_i, i = k, k+1, \ldots$. We can improve (i) by using a higher-order integration rule or by increasing N. We improve (ii) by including more terms in the truncated series—that is, increasing k. The pseudospectral approximation and this two-level convergence play an important role in the new algorithms proposed in Sect. 5.

4.3 The Lanczos algorithm

The Lanczos algorithm was originally introduced as an iterative scheme for approximating eigenvalues and eigenvectors of linear differential operators (Lanczos 1950). Given a symmetric $N \times N$ matrix $\boldsymbol{A}$, it constructs a symmetric, tridiagonal $k \times k$ matrix $\boldsymbol{T}$ whose eigenvalues approximate those of $\boldsymbol{A}$. Additionally, it produces an $N \times k$ matrix, denoted by $\boldsymbol{V} = [\mathbf{v}_0, \mathbf{v}_1, \ldots, \mathbf{v}_{k-1}]$, where the $\mathbf{v}_i$'s are the Lanczos vectors. These vectors transform the eigenvectors of $\boldsymbol{T}$ into approximate eigenvectors of $\boldsymbol{A}$. Algorithm 4 contains the steps of the Lanczos algorithm. Note that the inner products and norms in Algorithm 4 are given by

$$(\mathbf{w}, \mathbf{u}) = \mathbf{w}^\top \mathbf{u}, \qquad ||\mathbf{w}|| = \sqrt{(\mathbf{w}, \mathbf{w})}, \tag{40}$$

for vectors $\mathbf{w}, \mathbf{u} \in \mathbb{R}^N$. These relate to the discrete inner products introduced in Sect. 4.2, and we make precise connections later in this section.

After k iterations, the Lanczos algorithm yields the relationship

$$\boldsymbol{A}\boldsymbol{V} = \boldsymbol{V}\boldsymbol{T} + \beta_k \mathbf{v}_k \mathbf{e}_k^\top, \tag{43}$$

where $\mathbf{e}_k \in \mathbb{R}^k$ is the vector of zeros with a one in the kth entry and $\boldsymbol{V}$ and $\boldsymbol{T}$ are as in (41) and (42), respectively. The matrix $\boldsymbol{T}$, similar to $\boldsymbol{J}$ in (30), is the Jacobi matrix (Gautschi 2004). The relationship between the matrices $\boldsymbol{J}$ and $\boldsymbol{T}$ has been studied extensively (Gautschi 2002; Forsythe 1957; Boor and Golub 1978).

We are interested in the use of the Lanczos algorithm as a discrete approximation to the Stieltjes procedure. Recall that Algorithm 3 (the Stieltjes procedure) assumes a given measure π. Let λ_i, ω_i for $i = 0, \ldots, N-1$ be the nodes and weights for some N-point numerical integration rule with respect to π. This integration rule defines a discrete approximation of π, which we denote by $\pi^{(N)}$ (see Fig. 1). Inner products with respect to $\pi^{(N)}$ take the form of the discrete inner product in (37). If we perform the Lanczos algorithm with

Algorithm 4 Lanczos algorithm (Gautschi 2004, Section 3.1.7.1)

Given: An $N \times N$ symmetric matrix $\boldsymbol{A}$.
Assumptions: Let $\mathbf{v}_{-1} = \mathbf{0} \in \mathbb{R}^N$ and $\tilde{\mathbf{v}}_0$ be an arbitrary nonzero vector of length N.

1. For $i = 0, 1 \ldots, k-1$,
   (i) $\beta_i = ||\tilde{\mathbf{v}}_i||$
   (ii) $\mathbf{v}_i = \tilde{\mathbf{v}}_i / \beta_i$
   (iii) $\alpha_i = (\boldsymbol{A}\mathbf{v}_i, \mathbf{v}_i)$
   (iv) $\tilde{\mathbf{v}}_{i+1} = (\boldsymbol{A} - \alpha_i \boldsymbol{I})\mathbf{v}_i - \beta_{i-1}\mathbf{v}_{i-1}$
2. Define

$$\boldsymbol{V} = \begin{bmatrix} | & | & & | \\ \mathbf{v}_0 & \mathbf{v}_1 & \ldots & \mathbf{v}_{k-1} \\ | & | & & | \end{bmatrix} \tag{41}$$

and

$$\boldsymbol{T} = \begin{bmatrix} \alpha_0 & \beta_1 & & & \\ \beta_1 & \alpha_1 & \beta_2 & & \\ & \ddots & \ddots & \ddots & \\ & & \beta_{k-2} & \alpha_{k-2} & \beta_{k-1} \\ & & & \beta_{k-1} & \alpha_{k-1} \end{bmatrix}. \tag{42}$$

Output: The matrix of Lanczos vectors $\boldsymbol{V}$ and the matrix of recurrence coefficients $\boldsymbol{T}$.

$$\boldsymbol{A} = \begin{bmatrix} \lambda_0 & & \\ & \ddots & \\ & & \lambda_{N-1} \end{bmatrix}, \qquad \tilde{\mathbf{v}}_0 = \begin{bmatrix} \sqrt{\omega_0} \\ \vdots \\ \sqrt{\omega_{N-1}} \end{bmatrix}, \tag{44}$$

then the result is equivalent to running the Stieltjes procedure using the discrete inner product with respect to $\pi^{(N)}$. Increasing N improves the approximation of $\pi^{(N)}$ to π. It can be shown that the recurrence coefficients in the resulting Jacobi matrix will converge to the recurrence coefficients related to the Stieltjes procedure with respect to π as N increases (Gautschi 2004, Section 2.2). In the next section, we show how this relationship between Stieltjes and Lanczos can be used to approximate composite functions, which is essential to understand the underpinnings of LSIR and LSAVE in Sect. 5.

4.4 Composite function approximation

The connection between Algorithms 3 and 4 can be exploited for polynomial approximation of composite functions (Constantine and Phipps 2012). Consider a function of the form

$$h(x) = g(f(x)), \qquad x \in \mathcal{X} \subseteq \mathbb{R}, \tag{45}$$

where

$$\begin{aligned} f &: \mathcal{X} \to \mathcal{F} \subseteq \mathbb{R}, \\ g &: \mathcal{F} \to \mathcal{G} \subseteq \mathbb{R}. \end{aligned} \tag{46}$$

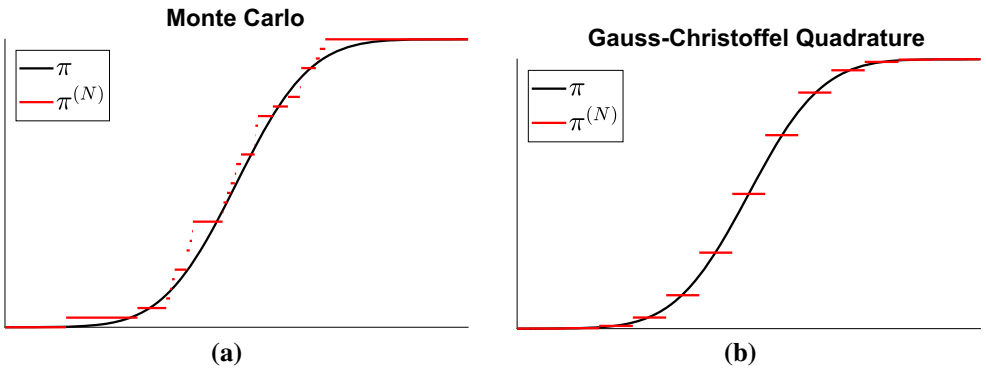


Fig. 1 Distribution functions associated with the probability measure π and the discrete approximation $\pi^{(N)}$. **a** constructs $\pi^{(N)}$ using a Monte Carlo integration rule with respect to π, while **b** uses Gauss–Christoffel quadrature to construct $\pi^{(N)}$

Assume the input space $\mathcal{X}$ is weighted with a given probability measure π_x. This measure and f induce a measure on $\mathcal{F}$ that we denote by π_y. Note that the methodology described in this section can be extended to multivariate inputs (i.e., $\mathcal{X} \subseteq \mathbb{R}^m$) through tensor product constructions and to multivariate outputs (i.e., $\mathcal{G} \subseteq \mathbb{R}^n$) by considering each output individually. We need both of these extensions in Sect. 5; however, we consider the scalar case here for clarity.

The goal is to construct a pseudospectral expansion of g using orthonormal polynomials with respect to π_y and a Gauss–Christoffel quadrature rule defined over $\mathcal{F}$. In Sect. 4.1, we examined the Stieltjes procedure which constructs a sequence of orthonormal polynomials with respect to a measure—π_y in this context. In Sect. 4.2, we showed this algorithm also produces the nodes and weights of the Gauss–Christoffel quadrature rule with respect to π_y. In Sect. 4.3, we saw how the Lanczos algorithm can be used to produce similar results for a discrete approximation to the measure π_y. All of this suggests a methodology for constructing a pseudospectral approximation of g. However, we constructed the discrete approximation of π_y in Sect. 4.3 using a numerical integration rule. We cannot do this here since π_y is unknown. We can construct an N-point numerical integration rule on $\mathcal{X}$ since π_x is known. Let x_i, v_i for $i = 0, \ldots, N-1$ denote the nodes and weights for our integration rule of choice with respect to π_x. This rule defines a discrete approximation of π_x, which we write as $\pi_x^{(N)}$. We approximate π_y by the discrete measure $\pi_y^{(N)}$ by evaluating $f_i = f(x_i)$ for $i = 0, \ldots, N-1$. We then perform the Lanczos algorithm on

$$\boldsymbol{A} = \begin{bmatrix} f_0 & & \\ & \ddots & \\ & & f_{N-1} \end{bmatrix}, \qquad \tilde{\mathbf{v}}_0 = \begin{bmatrix} \sqrt{v_0} \\ \sqrt{v_1} \\ \vdots \\ \sqrt{v_{N-1}} \end{bmatrix} \tag{47}$$

to obtain the system $\boldsymbol{A}\boldsymbol{V} = \boldsymbol{V}\boldsymbol{T} + \beta_k \mathbf{v}_k \mathbf{e}_k^\top$.

Let the eigendecomposition of the resulting Jacobi matrix be

$$\boldsymbol{T} = \boldsymbol{Q}\boldsymbol{\Lambda}\boldsymbol{Q}^\top. \tag{48}$$

The eigenvalues of $\boldsymbol{T}$ define the k-point Gauss–Christoffel quadrature nodes relative to $\pi_y^{(N)}$. We denote these quadrature nodes by $\lambda_i^{(N)}$, $i = 0, \ldots, k-1$, where the superscript indicates that these nodes are relative to the discrete measure $\pi_y^{(N)}$. From (32), the normalized eigenvectors of $\boldsymbol{T}$ have the form

$$(\boldsymbol{Q})_i = \frac{\boldsymbol{\phi}^{(N)}(\lambda_i^{(N)})}{||\boldsymbol{\phi}^{(N)}(\lambda_i^{(N)})||_2}, \qquad i = 0, \ldots, k-1, \tag{49}$$

where $\boldsymbol{\phi}^{(N)}(y) = [\,\phi_0^{(N)}(y)\,,\ \phi_1^{(N)}(y)\,,\ \ldots\,,\ \phi_{k-1}^{(N)}(y)\,]^\top$ and $\phi_i^{(N)}(y)$ denotes the ith orthonormal polynomial relative to $\pi_y^{(N)}$. The quadrature weight associated with $\lambda_i^{(N)}$ is given by the square of the first element of the ith normalized eigenvector,

$$\omega_i^{(N)} = (\boldsymbol{Q})_{0,i}^2, \qquad i = 0, \ldots, k-1. \tag{50}$$

From Sect. 4.3 we have convergence of the quadrature nodes $\lambda_i^{(N)}$ and weights $\omega_i^{(N)}$ to λ_i and ω_i, respectively, as N increases. In this sense, we consider these quantities to be approximations of the quadrature nodes and weights relative to π_y,

$$\lambda_i^{(N)} \approx \lambda_i \quad \text{and} \quad \omega_i^{(N)} \approx \omega_i. \tag{51}$$

In Sect. 6, we numerically study this approximation.

An alternative perspective on this k-point Gauss–Christoffel quadrature rule is that of a second discrete measure $\pi_y^{(N,k)}$ that approximates the measure $\pi_y^{(N)}$. That is,

$$\pi_y^{(N,k)} \approx \pi_y^{(N)} \approx \pi_y. \tag{52}$$

By taking more Lanczos iterations, we improve the leftmost approximation, and for $k = N$, we have $\pi_y^{(N,N)} = \pi_y^{(N)}$ (in exact arithmetic assuming that $f_0, \ldots, f_{N-1}$ from (47) are distinct). By increasing N (the number of points in the numerical integration rule with respect to π_x), we improve the rightmost approximation. This may be viewed as convergence of the discrete Lanczos algorithm to the continuous Stieltjes procedure. This mirrors the two-level approximation from (23). Both contain approximation over $\mathcal{X}$ by a chosen integration rule and approximation over $\mathcal{F}$ by an integration rule resulting from the different algorithms. The key difference is in the quality of those integration rules over $\mathcal{F}$—Gauss–Christoffel quadrature in (52) versus Riemann sums in (23).

Constantine and Phipps (2012) show that the Lanczos vectors resulting from performing the Lanczos algorithm on (47) also contain useful information. The Lanczos vectors are of the form

$$(\boldsymbol{V})_{i,j} \approx \sqrt{v_i}\,\phi_j(f_i), \qquad \begin{array}{l} i = 0, \ldots, N-1, \\ j = 0, \ldots, k-1, \end{array} \tag{53}$$

where $f_i = f(x_i)$, v_i is the weight associated with the node x_i, and ϕ_j is the jth-degree orthonormal polynomial with respect to π_y. The approximation in (53) is due to the approximation of π_y by $\pi_y^{(N)}$ and is in the same vein as the approximation in (51). We also numerically study this approximation in Sect. 6.

The approximation method in Constantine and Phipps (2012) suggests evaluating g at the $k \ll N$ quadrature nodes obtained from the Jacobi matrix and using these evaluations to construct a pseudospectral approximation. This allows for accurate estimation of g while placing a majority of the computational cost on evaluating f instead of both f and g in (45). Such an approach is valuable when g is difficult to compute relative to f. In the next section, we explain how this methodology can be used to construct approximations to $\boldsymbol{C}_{\text{IR}}$ and $\boldsymbol{C}_{\text{AVE}}$ from (20).

5 Lanczos–Stieltjes methods for inverse regression

In this section, we use the tools reviewed in Sect. 4 to develop a new Lanczos–Stieltjes approach to inverse regression methods—specifically to approximate the matrices $\boldsymbol{C}_{\text{IR}}$ and $\boldsymbol{C}_{\text{AVE}}$. This approach avoids approximating $\boldsymbol{C}_{\text{IR}}$ and $\boldsymbol{C}_{\text{AVE}}$ by Riemann sums (or slicing) as discussed in Sect. 3. Instead, we use orthonormal polynomials and quadrature approximations to build more accurate estimates of these matrices. For notation reference, we restate the problem setup from (15),

$$y = f(\mathbf{x}), \qquad \mathbf{x} \in \mathcal{X} \subseteq \mathbb{R}^m, \quad y \in \mathcal{F} \subseteq \mathbb{R}, \tag{54}$$

with $\mathcal{X}$ weighted by the known probability measure $\pi_{\mathbf{x}}$ and $\mathcal{F}$ weighted by the unknown probability measure π_y.

5.1 Lanczos–Stieltjes inverse regression (LSIR)

In Sect. 3, we showed that the SIR algorithm approximates the matrix

$$\boldsymbol{C}_{\text{IR}} = \int \boldsymbol{\mu}(y)\,\boldsymbol{\mu}(y)^\top \,\mathrm{d}\pi_y(y) \tag{55}$$

using a sliced mapping of the output (i.e., a Riemann sum approximation). We wish to approximate $\boldsymbol{C}_{\text{IR}}$ without such slicing; however, the structure of $\boldsymbol{\mu}(y)$ makes this difficult. Recall that the conditional expectation is the average of the inverse image of $f(\mathbf{x})$ for a fixed value of y,

$$\boldsymbol{\mu}(y) = \int \mathbf{x}\,\mathrm{d}\pi_{\mathbf{x}|y}(\mathbf{x}). \tag{56}$$

Approximating $\boldsymbol{\mu}(y)$ requires knowledge of the conditional measure $\pi_{\mathbf{x}|y}$, which may not be available if f is complex. However, using the tools from Sect. 4, we can exploit composite structure in $\boldsymbol{\mu}(y)$.

The conditional expectation (56) is a function that maps values of y to values in $\mathbb{R}^m$. Furthermore, y is itself a function of $\mathbf{x}$; see (54). Thus, the conditional expectation has composite structure, $\boldsymbol{\mu}(f(\mathbf{x}))$, where (using the notation from (46))

$$\begin{aligned} f &: \left(\mathcal{X} \subseteq \mathbb{R}^m\right) \to \left(\mathcal{F} \subseteq \mathbb{R}\right), \\ \boldsymbol{\mu} &: \left(\mathcal{F} \subseteq \mathbb{R}\right) \to \left(\mathcal{G} \subseteq \mathbb{R}^m\right). \end{aligned} \tag{57}$$

Using the techniques from Sect. 4.4, we seek to construct a pseudospectral expansion of $\boldsymbol{\mu}$ using the orthogonal polynomials and Gauss–Christoffel quadrature with respect to π_y.

Recall Assumption 1 ensures the inputs have finite fourth moments and are standardized according to (17). By Jensen's inequality,

$$\boldsymbol{\mu}(y)^\top \boldsymbol{\mu}(y) \leq \int \mathbf{x}^\top \mathbf{x}\, d\pi_{\mathbf{x}|y}(\mathbf{x}). \tag{58}$$

Integrating both sides of (58) with respect to π_y,

$$\begin{aligned} \int \boldsymbol{\mu}(y)^\top \boldsymbol{\mu}(y)\,\mathrm{d}\pi_y(y) &\leq \iint \mathbf{x}^\top \mathbf{x}\,\mathrm{d}\pi_{\mathbf{x}|y}(\mathbf{x})\,\mathrm{d}\pi_y(y) \\ &= \int \mathbf{x}^\top \mathbf{x}\,\mathrm{d}\pi_{\mathbf{x}}(\mathbf{x}). \end{aligned} \tag{59}$$

Under Assumption 1, the right-hand side of (59) is finite. This guarantees that each component of $\boldsymbol{\mu}(y)$ is square-integrable

with respect to π_y, ensuring that $\boldsymbol{\mu}(y)$ has a Fourier expansion in orthonormal polynomials with respect to π_y,

$$\boldsymbol{\mu}(y) \;=\; \sum_{i=0}^{\infty} \boldsymbol{\mu}_i \, \phi_i(y) \tag{60}$$

with equality in the $L_2(\pi_y)$ sense. The Fourier coefficients in (60) are

$$\boldsymbol{\mu}_i \;=\; \int \boldsymbol{\mu}(y) \, \phi_i(y) \, \mathrm{d}\pi_y(y). \tag{61}$$

Plugging (60) into (55),

$$\begin{aligned} \boldsymbol{C}_{\mathrm{IR}} \;&=\; \int \boldsymbol{\mu}(y) \, \boldsymbol{\mu}(y)^{\top} \, \mathrm{d}\pi_y(y) \\ &=\; \int \left[\sum_{i=0}^{\infty} \boldsymbol{\mu}_i \, \phi_i(y) \right] \left[\sum_{j=0}^{\infty} \boldsymbol{\mu}_j \, \phi_j(y) \right]^{\top} \mathrm{d}\pi_y(y) \\ &=\; \sum_{i=0}^{\infty} \sum_{j=0}^{\infty} \boldsymbol{\mu}_i \, \boldsymbol{\mu}_j^{\top} \left[\int \phi_i(y) \, \phi_j(y) \, \mathrm{d}\pi(y) \right] \\ &=\; \sum_{i=0}^{\infty} \boldsymbol{\mu}_i \, \boldsymbol{\mu}_i^{\top}. \end{aligned} \tag{62}$$

Thus, the $\boldsymbol{C}_{\mathrm{IR}}$ matrix can be computed as the sum of the outer products of Fourier coefficients from (61).

We cannot compute the Fourier coefficients directly since they require knowledge of $\boldsymbol{\mu}(y)$. However, we can rewrite (61) as

$$\begin{aligned} \boldsymbol{\mu}_i \;&=\; \int \boldsymbol{\mu}(y) \, \phi_i(y) \, \mathrm{d}\pi_y(y) \\ &=\; \int \left[\int \mathbf{x} \, \mathrm{d}\pi_{\mathbf{x}|y}(\mathbf{x}) \right] \phi_i(y) \, \mathrm{d}\pi_y(y) \\ &=\; \iint \mathbf{x} \, \phi_i(y) \, d\pi_{\mathbf{x}|y}(\mathbf{x}) \, \mathrm{d}\pi_y(y) \\ &=\; \int \mathbf{x} \, \phi_i(f(\mathbf{x})) \, \mathrm{d}\pi_{\mathbf{x}}(\mathbf{x}). \end{aligned} \tag{63}$$

This form of the Fourier coefficients is more amenable to numerical approximation. Since $\pi_{\mathbf{x}}$ is known, we can obtain an N-point integration rule with nodes $\mathbf{x}_j$ and weights ν_j for $j = 0, \ldots, N-1$. Recall from Sect. 4.2 that we require all of the quadrature weights ν_j to be positive so that they define a valid inner product and norm. We then define the pseudospectral coefficients with respect to the N-point multivariate integration rule,

$$\hat{\boldsymbol{\mu}}_i \;=\; \sum_{j=0}^{N-1} \nu_j \, \mathbf{x}_j \, \phi_i(f(\mathbf{x}_j)). \tag{64}$$

To evaluate ϕ_i at $f(\mathbf{x}_j)$, we use the Lanczos vectors in $\boldsymbol{V}$. Recall from (53) that these vectors approximate the orthonormal polynomials from Stieltjes at f evaluated at the nodes scaled by the square root of the associated weights. Algorithm 5 formalizes this process as the Lanczos–Stieltjes inverse regression (LSIR) algorithm.

Algorithm 5 Lanczos–Stieltjes inverse regression (LSIR)

Given: The function $f : \mathbb{R}^m \to \mathbb{R}$ and input probability measure $\pi_{\mathbf{x}}$.
Assumptions: Assumption 1 holds.

1. Obtain the nodes and weights for a valid N-point integration rule with respect to $\pi_{\mathbf{x}}$,

$$\mathbf{x}_i, \;\; \nu_i, \quad \text{for} \quad i = 0, \ldots, N-1. \tag{65}$$

2. Evaluate $f_i = f(\mathbf{x}_i)$ for $i = 0, \ldots, N-1$.
3. Perform k iterations of Algorithm 4 on

$$\boldsymbol{A} \;=\; \begin{bmatrix} f_0 & & \\ & \ddots & \\ & & f_{N-1} \end{bmatrix}, \qquad \tilde{\mathbf{v}}_0 \;=\; \begin{bmatrix} \sqrt{\nu_0} \\ \sqrt{\nu_1} \\ \vdots \\ \sqrt{\nu_{N-1}} \end{bmatrix} \tag{66}$$

to obtain $\boldsymbol{A}\boldsymbol{V} = \boldsymbol{V}\boldsymbol{T} + \eta_k \mathbf{v}_k \mathbf{e}_k^{\top}$.
4. For $i = 0, \ldots, m-1$, $\ell = 0, \ldots, k-1$,
Compute the ith component of the ℓth pseudospectral coefficient

$$\left(\hat{\boldsymbol{\mu}}_{\ell}\right)_i \;=\; \sum_{p=0}^{N-1} \sqrt{\nu_p} \left(\mathbf{x}_p\right)_i \left(\boldsymbol{V}\right)_{p,\ell}. \tag{67}$$

5. For $i, j = 0, \ldots, m-1$,
Compute the i, jth component of $\hat{\boldsymbol{C}}_{\mathrm{IR}}$

$$\left(\hat{\boldsymbol{C}}_{\mathrm{IR}}\right)_{i,j} \;=\; \sum_{\ell=0}^{k-1} \left(\hat{\boldsymbol{\mu}}_{\ell}\right)_i \left(\hat{\boldsymbol{\mu}}_{\ell}\right)_j. \tag{68}$$

Output: The matrix $\hat{\boldsymbol{C}}_{\mathrm{IR}}$.

Algorithm 5 depends on two levels of approximation: (i) approximation due to the numerical integration rule on $\mathcal{X}$ and (ii) approximation due to truncating the polynomial expansion of $\boldsymbol{\mu}(y)$. The former depends on the number N of points in the numerical integration rule over $\mathcal{X}$, while the latter depends on the number k of Lanczos iterations. Performing more Lanczos iterations includes more terms in the approximation of $\boldsymbol{C}_{\mathrm{IR}}$; however, the additional terms correspond to integrals against higher-degree polynomials in (61). For fixed N, the quality of the pseudospectral approximation deteriorates as the degree of polynomial increases. Therefore, sufficiently many points are needed to ensure quality estimates of the orthonormal polynomials of high degree. Quantifying this statement precisely is outside the scope of the current paper; the numerical experiments in Sect. 6 show these phenomena.

To compare the computational cost of the LSIR algorithm to its slice-based counterpart, we consider the approximate

costs of the Lanczos method and the slicing procedure. Due to its origins as an iterative procedure for approximating eigenvalues, the computational costs of the Lanczos algorithm have been well studied. For the diagonal matrix $\boldsymbol{A} \in \mathbb{R}^{N\times N}$, performing k iterations the Lanczos algorithm requires $\mathcal{O}(kN)$ operations (Golub and Van Loan 1996, Ch. 9). The costs associated with the SIR algorithm arise from sorting the outputs in order to define the slices. Sorting algorithms are known to take an average of $\mathcal{O}(N\,\log(N))$ operations (Ajtai et al. 1983). However, in practice the most significant cost is typically the cost of the evaluations $f_i = f(\mathbf{x}_i)$—a necessary step for both the Lanczos–Stieltjes and the slice-based approaches.

5.2 Lanczos–Stieltjes average variance estimation (LSAVE)

In this section, we apply the Lanczos–Stieltjes approach from Sect. 5.1 to the $\boldsymbol{C}_{\text{AVE}}$ matrix to construct an alternative algorithm to the slice-based SAVE. Recall from (20) that

$$\boldsymbol{C}_{\text{AVE}} \;=\; \int (\boldsymbol{I} - \boldsymbol{\Sigma}(y))^2 \,\mathrm{d}\pi_y(y), \tag{69}$$

where the conditional covariance of the inverse image of $f(\mathbf{x})$ for a fixed value of y is

$$\boldsymbol{\Sigma}(y) \;=\; \int (\mathbf{x} - \boldsymbol{\mu}(y))\,(\mathbf{x} - \boldsymbol{\mu}(y))^\top \,\mathrm{d}\pi_{\mathbf{x}|y}(\mathbf{x}). \tag{70}$$

Similar to the conditional expectation, $\boldsymbol{\Sigma}(y)$ has composite structure due to the relationship $y = f(\mathbf{x})$ such that we can write $\boldsymbol{\Sigma}(f(\mathbf{x}))$, where

$$\begin{aligned} f &: \left(\mathcal{X} \subseteq \mathbb{R}^m\right) \to (\mathcal{F} \subseteq \mathbb{R}), \\ \boldsymbol{\Sigma} &: (\mathcal{F} \subseteq \mathbb{R}) \to \left(\mathcal{G} \subseteq \mathbb{R}^{m\times m}\right). \end{aligned} \tag{71}$$

We want to build a pseudospectral expansion of $\boldsymbol{\Sigma}$ with respect to π_y similar to $\boldsymbol{\mu}$ in Sect. 5.1.

Recall Assumption 1 as we consider the Frobenius norm of the conditional covariance. By Jensen's inequality,

$$\|\boldsymbol{\Sigma}(y)\|_F^2 \;\le\; \int \left\|(\mathbf{x} - \boldsymbol{\mu}(y))\,(\mathbf{x} - \boldsymbol{\mu}(y))^\top\right\|_F^2 \mathrm{d}\pi_{\mathbf{x}|y}(\mathbf{x}). \tag{72}$$

Integrating each side with respect to π_y,

$$\begin{aligned} &\int \|\boldsymbol{\Sigma}(y)\|_F^2 \,\mathrm{d}\pi_y(y) \\ &\quad\le \iint \left\|(\mathbf{x} - \boldsymbol{\mu}(y))\,(\mathbf{x} - \boldsymbol{\mu}(y))^\top\right\|_F^2 \mathrm{d}\pi_{\mathbf{x}|y}(\mathbf{x})\,\mathrm{d}\pi_y(y) \\ &\quad\le \int \left\|(\mathbf{x} - \boldsymbol{\mu}(f(\mathbf{x})))\,(\mathbf{x} - \boldsymbol{\mu}(f(\mathbf{x})))^\top\right\|_F^2 \mathrm{d}\pi_{\mathbf{x}}(\mathbf{x}). \end{aligned} \tag{73}$$

Expanding the integrand on the right-hand side produces sums and products of fourth and lower conditional moments of the inputs. Assumption 1 guarantees that all of these conditional moments are finite. Thus,

$$\int \|\boldsymbol{\Sigma}(y)\|_F^2 \,\mathrm{d}\pi_y(y) \;<\; \infty, \tag{74}$$

which implies that each component of $\boldsymbol{\Sigma}(y)$ is square-integrable with respect to π_y; therefore, it has a convergent Fourier expansion in terms of orthonormal polynomials with respect to π_y,

$$\boldsymbol{\Sigma}(y) \;=\; \sum_{i=0}^{\infty} \boldsymbol{\Sigma}_i\,\phi_i(y), \tag{75}$$

where equality is in the $L_2(\pi_y)$ sense. The coefficients in (75) are

$$\boldsymbol{\Sigma}_i \;=\; \int \boldsymbol{\Sigma}(y)\,\phi_i(y)\,\mathrm{d}\pi_y(y). \tag{76}$$

Plugging (75) into (69)

$$\begin{aligned} \boldsymbol{C}_{\text{AVE}} \;&=\; \int (\boldsymbol{I} - \boldsymbol{\Sigma}(y))^2 \,\mathrm{d}\pi_y(y) \\ &=\; \int \left(\boldsymbol{I} - \sum_{i=0}^{\infty} \boldsymbol{\Sigma}_i\,\phi_i(y)\right)^2 \mathrm{d}\pi_y(y) \\ &=\; \int \boldsymbol{I}\,d\pi_y(y) - 2\sum_{i=0}^{\infty} \boldsymbol{\Sigma}_i \int \phi_i(y)\,\mathrm{d}\pi_y(y) \\ &\qquad + \sum_{i=0}^{\infty}\sum_{j=0}^{\infty} \boldsymbol{\Sigma}_i\,\boldsymbol{\Sigma}_j \int \phi_i(y)\,\phi_j(y)\,\mathrm{d}\pi_y(y) \\ &=\; \boldsymbol{I} - 2\,\boldsymbol{\Sigma}_0 + \sum_{i=0}^{\infty} \boldsymbol{\Sigma}_i^2. \end{aligned} \tag{77}$$

Therefore, we can compute $\boldsymbol{C}_{\text{AVE}}$ using the Fourier coefficients of $\boldsymbol{\Sigma}(y)$. To simplify the computation of $\boldsymbol{\Sigma}_i$, we rewrite 76 as

$$\begin{aligned} \boldsymbol{\Sigma}_i \;&=\; \int \boldsymbol{\Sigma}(y)\,\phi_i(y)\,\mathrm{d}\pi_y(y) \\ &=\; \int\int (\mathbf{x} - \boldsymbol{\mu}(y))\,(\mathbf{x} - \boldsymbol{\mu}(y))^\top \,\mathrm{d}\pi_{\mathbf{x}|y}(\mathbf{x})\,\phi_i(y)\,\mathrm{d}\pi_y(y) \\ &=\; \int\int (\mathbf{x} - \boldsymbol{\mu}(y))\,(\mathbf{x} - \boldsymbol{\mu}(y))^\top\,\phi_i(y)\,\mathrm{d}\pi_{\mathbf{x}|y}(\mathbf{x})\,\mathrm{d}\pi_y(y) \\ &=\; \int (\mathbf{x} - \boldsymbol{\mu}(f(\mathbf{x})))\,(\mathbf{x} - \boldsymbol{\mu}(f(\mathbf{x})))^\top\,\phi_i(f(\mathbf{x}))\,\mathrm{d}\pi_{\mathbf{x}}(\mathbf{x}). \end{aligned} \tag{78}$$

We approximate this integral using the N-point numerical integration rule with respect to $\pi_{\mathbf{x}}$ to obtain

$$\hat{\boldsymbol{\Sigma}}_i \;=\; \sum_{j=0}^{N-1} \nu_j \left(\mathbf{x}_j - \boldsymbol{\mu}(f(\mathbf{x}_j))\right) \times \left(\mathbf{x}_j - \boldsymbol{\mu}(f(\mathbf{x}_j))\right)^\top \phi_i(f(\mathbf{x}_j)). \tag{79}$$

We again approximate $\phi_i(f(\mathbf{x}_j))$ using the Lanczos vectors similar to LSIR; see (53). Notice that (79) also depends on $\boldsymbol{\mu}(f(\mathbf{x}_j))$ for $j = 0, \ldots, N-1$. To obtain these values, we compute the pseudospectral coefficients of $\boldsymbol{\mu}(f(\mathbf{x}))$ from (64) and construct its pseudospectral expansion at each $\mathbf{x}_j$. Algorithm 6 provides an outline for Lanczos–Stieltjes average variance estimation (LSAVE).

Algorithm 6 contains the same two-level approximation as the LSIR algorithm. As such, it also requires a sufficiently high-order integration rule to accurately approximate the high-degree polynomials resulting from k Lanczos iterations. In the next section, we provide numerical studies of the LSIR and LSAVE algorithms on several test problems as well as comparisons to the traditional SIR and SAVE algorithms.

6 Numerical results

In this section, we numerically study the LSIR and LSAVE algorithms. In Sect. 6.1, we study the approximation of the Gauss–Christoffel quadrature and orthonormal polynomials on the output space by the Lanczos algorithm as described in Sect. 4.4. This study is performed on a multivariate quadratic function of dimension $m = 3$. In Sect. 6.2, we examine the approximation of $\boldsymbol{C}_{\text{IR}}$ and $\boldsymbol{C}_{\text{AVE}}$ by the Lanczos–Stieltjes estimates $\hat{\boldsymbol{C}}_{\text{IR}}$ and $\hat{\boldsymbol{C}}_{\text{AVE}}$, and we compare these results to the slice-based estimates $\hat{\boldsymbol{C}}_{\text{SIR}}$ and $\hat{\boldsymbol{C}}_{\text{SAVE}}$. This study is performed using a standard test problem in dimension reduction and uncertainty quantification: the OTL circuit function (Surjanovic and Bingham 2015). This is a model of the output from a transformerless push–pull circuit with $m = 6$ inputs. MATLAB code for performing these studies is available at [https://bitbucket.org/aglaws/gauss-christoffel-quadrature-for-inverse-regression].

Recall that the goal of the proposed algorithms is to approximate the $\boldsymbol{C}_{\text{IR}}$ and $\boldsymbol{C}_{\text{AVE}}$ matrices more efficiently than the slice-based (Riemann sum) approach in SIR and SAVE. Therefore, most of the presented results are in terms of the Frobenius norm of the relative matrix errors. Recall that the Frobenius norm is

$$\|\boldsymbol{E}\|_F \;=\; \left(\sum_{i=0}^{m-1}\sum_{j=0}^{m-1} (\boldsymbol{E})_{i,j}^2\right)^{1/2}, \tag{86}$$

which we use to measure error in the matrix approximations.

Algorithm 6 Lanczos–Stieltjes average variance estimation (LSAVE)

Given: The function $f : \mathbb{R}^m \to \mathbb{R}$ and input probability measure $\pi_{\mathbf{x}}$.
Assumptions: Assumption 1 holds.

1. Obtain the nodes and weights for a valid N-point integration rule with respect to $\pi_{\mathbf{x}}$,
$$\mathbf{x}_i, \;\; \nu_i, \quad \text{for} \quad i = 0, \ldots, N-1. \tag{80}$$
2. Evaluate $f_i = f(\mathbf{x}_i)$ for $i = 0, \ldots, N-1$.
3. Perform k iterations of Algorithm 4 on
$$\boldsymbol{A} \;=\; \begin{bmatrix} f_0 & & \\ & \ddots & \\ & & f_{N-1} \end{bmatrix}, \qquad \tilde{\mathbf{v}}_0 \;=\; \begin{bmatrix} \sqrt{\nu_0} \\ \sqrt{\nu_1} \\ \vdots \\ \sqrt{\nu_{N-1}} \end{bmatrix} \tag{81}$$
to obtain $\boldsymbol{A}\boldsymbol{V} = \boldsymbol{V}\boldsymbol{T} + \eta_k \mathbf{v}_k \mathbf{e}_k^\top$.
4. For $i = 0, \ldots, m-1$, $\ell = 0, \ldots, k-1$,
Compute the ith component of the ℓth pseudospectral coefficient of $\boldsymbol{\mu}(y)$
$$\left(\hat{\boldsymbol{\mu}}_\ell\right)_i \;=\; \sum_{p=0}^{N-1} \sqrt{\nu_p} \left(\mathbf{x}_p\right)_i (\boldsymbol{V})_{p,\ell}\,. \tag{82}$$
5. For $i = 0, \ldots, m-1$,
Compute the ith component of the pseudospectral expansion of $\boldsymbol{\mu}(f(\mathbf{x}_p))$
$$\left(\hat{\boldsymbol{\mu}}(f(\mathbf{x}_p))\right)_i \;=\; \sum_{\ell=0}^{k-1} \frac{1}{\sqrt{\nu_p}} \left(\hat{\boldsymbol{\mu}}_\ell\right)_i (\boldsymbol{V})_{p,\ell}\,. \tag{83}$$
6. For $i, j = 0, \ldots, m-1$, $\ell = 0, \ldots, k-1$,
Compute the i, jth component of the ℓth pseudospectral coefficient of $\boldsymbol{\Sigma}(y)$
$$\left(\hat{\boldsymbol{\Sigma}}_\ell\right)_{i,j} \;=\; \sum_{p=0}^{N-1} \sqrt{\nu_p} \left((\mathbf{x}_p)_i - \left(\hat{\boldsymbol{\mu}}(f(\mathbf{x}_p))\right)_i\right) \times \left((\mathbf{x}_p)_j - \left(\hat{\boldsymbol{\mu}}(f(\mathbf{x}_p))\right)_j\right) (\boldsymbol{V})_{p,\ell}\,. \tag{84}$$
7. For $i, j = 0, \ldots, m-1$,
Compute the i, jth component of $\hat{\boldsymbol{C}}_{\text{AVE}}$
$$\left(\hat{\boldsymbol{C}}_{\text{AVE}}\right)_{i,j} \;=\; \delta_{i,j} - 2\left(\hat{\boldsymbol{\Sigma}}_0\right)_{i,j} + \sum_{\ell=0}^{k-1}\sum_{p=0}^{m-1} \left(\hat{\boldsymbol{\Sigma}}_\ell\right)_{i,p} \left(\hat{\boldsymbol{\Sigma}}_\ell\right)_{p,j}. \tag{85}$$

Output: The matrix $\hat{\boldsymbol{C}}_{\text{AVE}}$.

6.1 Example 1: Lanczos–Stieltjes convergence

To study numerically the convergence of the Lanczos–Stieltjes approach, we consider the function

$$y \;=\; f(\mathbf{x}) \;=\; \mathbf{g}^\top \mathbf{x} + \mathbf{x}^\top \boldsymbol{H} \mathbf{x}, \qquad \mathbf{x} \in [-1, 1]^3 \subset \mathbb{R}^3, \tag{87}$$

where $\mathbf{g} \in \mathbb{R}^3$ is a constant vector and $\boldsymbol{H} \in \mathbb{R}^{3\times 3}$ is a constant matrix. We assume the inputs are weighted by the uniform

density over the input space $\mathcal{X} = [-1, 1]^3$,

$$\mathrm{d}\pi_{\mathbf{x}}(\mathbf{x}) = \begin{cases} \frac{1}{2^3}\, d\mathbf{x} & \text{if } ||\mathbf{x}||_\infty \leq 1, \\ 0\, d\mathbf{x} & \text{otherwise.} \end{cases} \tag{88}$$

Recall from Sect. 4.4 that we can use the Lanczos algorithm to obtain a k-point Gauss–Christoffel quadrature rule and the first k orthonormal polynomials relative to the discrete measure $\pi_y^{(N)}$. We treat these as approximations to the quadrature rule and orthonormal polynomials relative to the continuous measure π_y [see (51) and (53)]. In this section, we study the behavior of these approximations for (87).

We use two different integration rules for this study: a tensor product Clenshaw–Curtis quadrature rule (Clenshaw and Curtis 1960) and simple Monte Carlo (Owen 2013). We do this to emphasize that approximation accuracy of the Gauss–Christoffel quadrature rule and the orthonormal polynomials with respect to π_y depends on the quality of the integration rule chosen with respect to $\pi_{\mathbf{x}}$.

Recall from (53) that the Lanczos vectors contain evaluations of the first k (corresponding to the number of Lanczos iterations performed) orthonormal polynomials with respect to $\pi_y^{(N)}$ at the points $f_i = f(\mathbf{x}_i)$ weighted by the square root of the associated weights, $\sqrt{v_i}$. To compare the approximations for increasing numbers of samples, we must ensure that our integration rules are nested. That is, the $\mathbf{x}_i$'s of the N_j-point integration rule must be included in the N_{j+1}-point integration rule, where N_j and N_{j+1} denote subsequently increasing numbers of points. We use the Clenshaw–Curtis quadrature rule here because it is a nested quadrature rule. For Monte Carlo, we append new independent random samples to our current set to ensure the nested structure.

First, we study convergence of the Gauss–Christoffel quadrature rule produced on the output space $\mathcal{F}$ by the Lanczos algorithm. Gautschi (2004) shows the convergence of the Jacobi matrix (42) to (30) as the discrete approximation $\pi_y^{(N)}$ approaches π_y; however, we are specifically interested in the quadrature rule resulting from the eigendecomposition of the Jacobi matrix. Recall from (51) that $\lambda_i^{(N)}$ and $\omega_i^{(N)}$ denote the ith Gauss–Christoffel quadrature node and weight with respect to $\pi_y^{(N)}$. Figure 2 shows the differences in the 5-point quadrature rules with increasing samples over $\mathcal{X}$,

$$\left|\hat{\lambda}_i^{(N_{j+1})} - \hat{\lambda}_i^{(N_j)}\right| \quad \text{and} \quad \left|\hat{\omega}_i^{(N_{j+1})} - \hat{\omega}_i^{(N_j)}\right|, \tag{89}$$

for $i = 0, \ldots, 4$. Using Clenshaw–Curtis quadrature rules over the input space (Fig. 2a, b), the Gauss–Christoffel quadrature rule with respect to π_y converges exponentially. In Fig. 2c, d, we see the expected $N^{-1/2}$ convergence in the Gauss–Christoffel quadrature rule when using Monte Carlo to approximate $\pi_{\mathbf{x}}$. Quality estimates of the quadrature rule over the output space $\mathcal{F}$ are required to produce good approximations of the $\boldsymbol{C}_{\mathrm{IR}}$ and $\boldsymbol{C}_{\mathrm{AVE}}$ using the Lanczos–Stieltjes approach.

Next, we examine the convergence of the Lanczos vectors to the orthonormal polynomials with respect to π_y. Let $\boldsymbol{V}^{(N_j)}$ be the matrix of k Lanczos vectors resulting from performing Lanczos with an N_j-point integration rule on the input space $\mathcal{X}$ with respect to $\pi_{\mathbf{x}}$. Define the matrix $\boldsymbol{W}_v = \mathrm{diag}\left([\sqrt{v_0} \ \ldots \ \sqrt{v_{N_j-1}}]\right)$, and let $\tilde{\boldsymbol{V}}^{(N_j)} = \boldsymbol{W}_v^{-1}\boldsymbol{V}^{(N_j)}$ be the matrix of orthonormal polynomials evaluated at the f_i's (no longer scaled by the $\sqrt{v_i}$'s). Due to the nestedness of the integration rules, $\tilde{\boldsymbol{V}}^{(N_j)}$ contains evaluations of the same k orthonormal polynomials at nested sets of f_i's as we increase the number of samples. Let $\boldsymbol{P} \in \mathbb{R}^{N_j \times N_{j+1}}$ be the matrix that removes the rows of $\tilde{\boldsymbol{V}}^{(N_{j+1})}$ that do not correspond to the rows in $\tilde{\boldsymbol{V}}^{(N_j)}$. We write the maximum difference in the ith polynomial for subsequent orders of the quadrature rule as

$$\left\| \boldsymbol{P}\left(\tilde{\boldsymbol{V}}^{(N_{j+1})}\right)_i - \left(\tilde{\boldsymbol{V}}^{(N_j)}\right)_i \right\|_\infty, \tag{90}$$

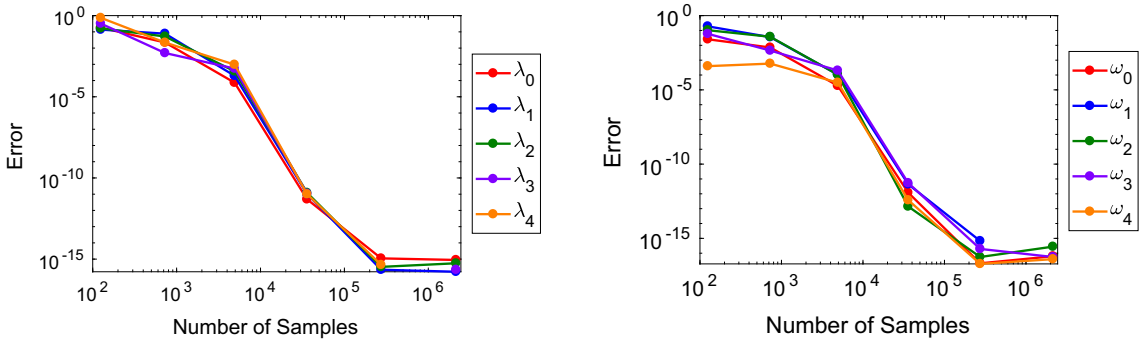
where $(\cdot)_i$ denotes the ith column of the given matrix for $i = 0, \ldots, k-1$.

Figure 3a, b contains plots of the maximum differences in (90) for increasing numbers of Clenshaw–Curtis points and Monte Carlo points, respectively. In both plots, higher-degree polynomials require more samples to produce accurate approximations. This is not surprising, but it does highlight an important relationship between the numerical integration rule used with respect to $\pi_{\mathbf{x}}$ and the number of Lanczos iterations performed. Namely, as the number of Lanczos iterations increases, more samples are required to ensure accurate approximation of the orthonormal polynomials (and, in turn, $\boldsymbol{C}_{\mathrm{IR}}$ and $\boldsymbol{C}_{\mathrm{AVE}}$). Additionally, we see that the convergence rate of the orthonormal polynomials depends on the integration rule chosen over $\mathcal{X}$.

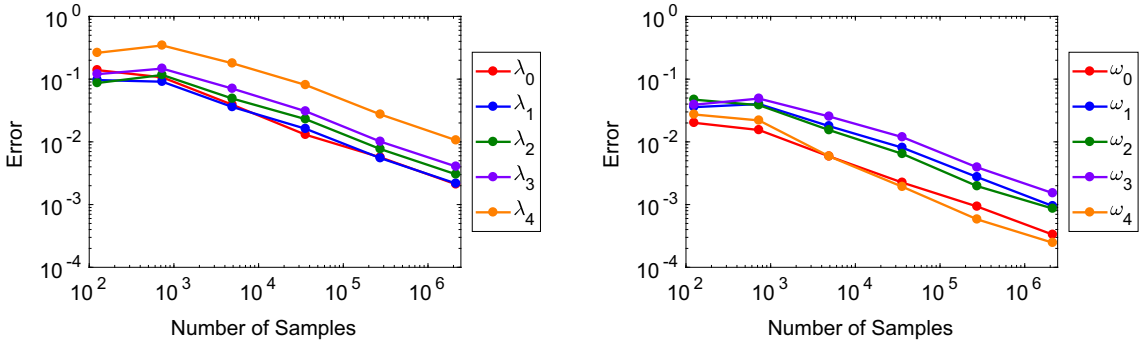
6.2 Example 2: OTL circuit function

We consider the physically motivated OTL circuit function. This function models the midpoint voltage (V_m) from a transformerless push–pull circuit [see Surjanovic and Bingham (2015) and Ben-Ari and Steinberg (2007)]. It has the form

$$\begin{aligned} V_m = & \frac{(V_{b1} + 0.74)\,\beta\,(R_{c2} + 9)}{\beta\,(R_{c2} + 9) + R_f} \\ & + \frac{11.35\,R_f}{\beta\,(R_{c2} + 9) + R_f} \\ & + \frac{0.74\,R_f\,\beta\,(R_{c2} + 9)}{(\beta\,(R_{c2} + 9) + R_f)\,R_{c1}}, \end{aligned} \tag{91}$$

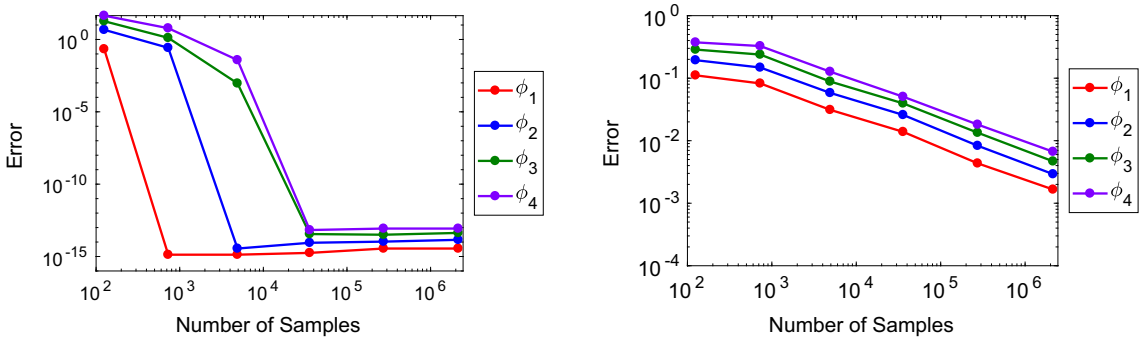


(a) Node convergence with CC **(b)** Weight convergence with CC



(c) Node convergence with MC **(d)** Weight convergence with MC

Fig. 2 Differences in the 5-point Gauss–Christoffel quadrature nodes λ_i and weights ω_i resulting from the Lanczos algorithm applied to (87). **a**, **b** contain the differences using a tensor product Clenshaw–Curtis quadrature rule used over $\mathcal{X}$, and **c**, **d** use Monte Carlo samples



(a) CC integration **(b)** MC integration

Fig. 3 Maximum differences in the approximated orthonormal polynomials as the number of quadrature points on $\mathcal{X}$ increases. **a** uses a tensor product Clenshaw–Curtis quadrature rule over the input space, while **b** uses Monte Carlo integration

where

$$V_{b1} = \frac{12\, R_{b2}}{R_{b1} + R_{b2}}. \tag{92}$$

The six physical inputs to the OTL circuit model are described in Table 1. The table also contains the ranges of the inputs. We assume the density $\pi_{\mathbf{x}}$ on the input space, defined by the product of the ranges in Table 1, is a uniform density.

For our first study, we use tensor product Gauss–Christoffel quadrature rules on the input space. The number of points in a tensor product rule grows exponentially with dimension, so they are not appropriate for more than a handful of inputs. The point of this study is to emphasize and demonstrate how the LSIR and LSAVE algorithms remove the approximation bottleneck caused by the Riemann sums in SIR and SAVE and place the burden of accuracy on the numerical integration rule over the input space. The results for this study are shown in Fig. 4.

Figure 4a demonstrates the convergence of the LSIR algorithm in terms of the number N of Gauss–Christoffel

Table 1 Six physical inputs for the OTL circuit function with their ranges. We assume a uniform density over these ranges

Input	Description (units)	Range
R_{b1}	Resistance b1 (K-ohms)	[50, 150]
R_{b2}	Resistance b2 (K-ohms)	[25, 70]
R_f	Resistance f (K-ohms)	[0.5, 3]
R_{c1}	Resistance c1 (K-ohms)	[1.2, 2.5]
R_{c2}	Resistance c2 (K-ohms)	[0.25, 1.2]
β	Current gain (amperes)	[50, 300]

quadrature nodes on the input space, and Fig. 4b shows convergence in terms of the number k of Lanczos iterations performed. These plots show the Frobenius norm of the matrix differences between subsequent Lanczos–Stieltjes approximations of $\boldsymbol{C}_{\text{IR}}$ computed using increasing numbers of quadrature nodes (with $k = 35$ fixed) and Lanczos iterations (with $N = 17^6 = 24{,}137{,}569$ fixed), respectively. We see a decay in these differences as we increase N and k, suggesting that the LSIR algorithm is converging.

For Fig. 4c, we perform Algorithm 5 for $N = 17^6 = 24{,}137{,}569$ and $k = 35$ and treat the resulting matrix as the "true" value of $\boldsymbol{C}_{\text{IR}}$. We then compute errors relative to this matrix for various values of N and k. Figure 4c shows the relative error decaying as we increase both the number of quadrature nodes and Lanczos iterations (i.e., as we move down and to the right). Consider this plot for increasing N with k fixed (i.e., moving downward at a fixed point along the horizontal axis). The error decays up to a point at which it remains constant. This decay corresponds to more accurate computation of the pseudospectral coefficients $\hat{\boldsymbol{\mu}}_i$ from (64) by taking more quadrature nodes. The leveling off corresponds to the point at which errors in the coefficients are smaller than errors due to truncating the pseudospectral expansion at k (the number of Lanczos iterations). Conversely, if we fix N and study the error as we increase k (i.e., fix a point along the vertical axis and move right), we see the error decay until a point at which it begins to grow again. This behavior agrees with the results from Sect. 6.1 that suggest that sufficiently many quadrature nodes are needed to accurately estimate the high-degree polynomials associated with large values of k. As we move right in Fig. 4c, we are approximating higher-degree polynomials using the Lanczos algorithm. Poor approximation of these polynomials results in an inaccurate estimate of $\boldsymbol{C}_{\text{IR}}$.

Figure 4d–f shows the results of the same studies as above, but performed on the LSAVE algorithm (Algorithm 6). The results and interpretations are similar to those for the LSIR algorithm.

Next we examine how the approximated SIR and SAVE matrices from Algorithms 1 and 2, respectively, compare to the LSIR and LSAVE approximations. Recall $\hat{\boldsymbol{C}}_{\text{SIR}}$ and $\hat{\boldsymbol{C}}_{\text{SAVE}}$ contain two levels of approximation—one due to the number of samples N and one due to the number of terms in the Riemann sum (i.e., the number of slices) R over the output space. We first focus on convergence in terms of Riemann sums. Figure 5 shows the comparison of the approx-

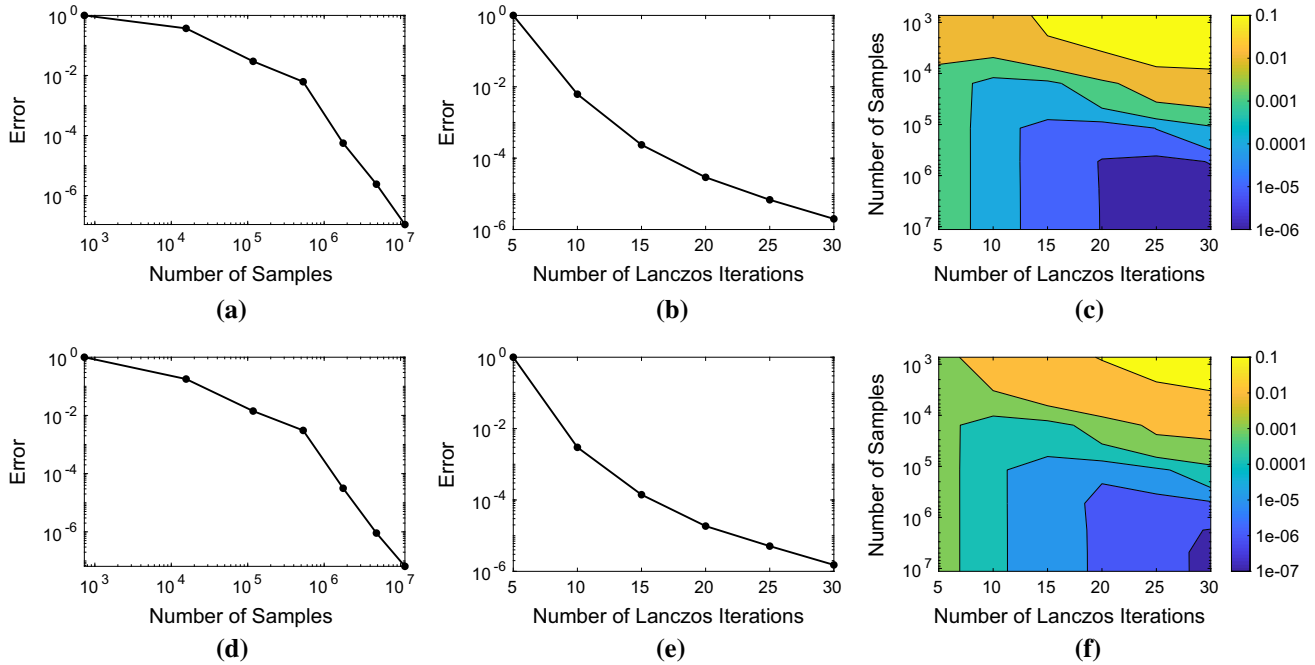


Fig. 4 Convergence studies for the LSIR algorithm (**a**, **b**, **c**) and the LSAVE algorithm (**d**, **e**, **f**) on the OLT circuit function from (91)

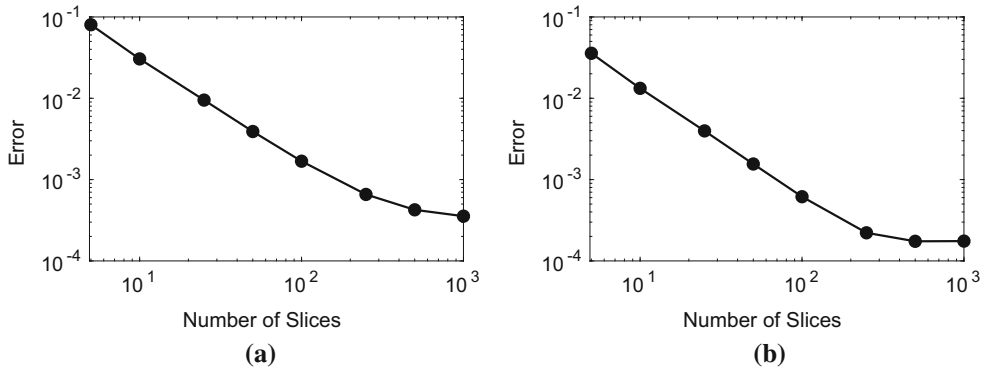


Fig. 5 A comparison of the Riemann sum approximation and the Lanczos–Stieltjes approximation of $\boldsymbol{C}_{\mathrm{IR}}$ (**a**) and $\boldsymbol{C}_{\mathrm{AVE}}$ (**b**) for increasing number of Riemann sums (or slices) for (91). For the Lanczos–Stieltjes approximations, we used $N = 17^6 = 24{,}137{,}569$ Gauss–Christoffel quadrature nodes on $\mathcal{X}$ and $k = 35$ Lanczos iterations. For the slice-based approximations, we used $N = 10^8$ Monte Carlo samples

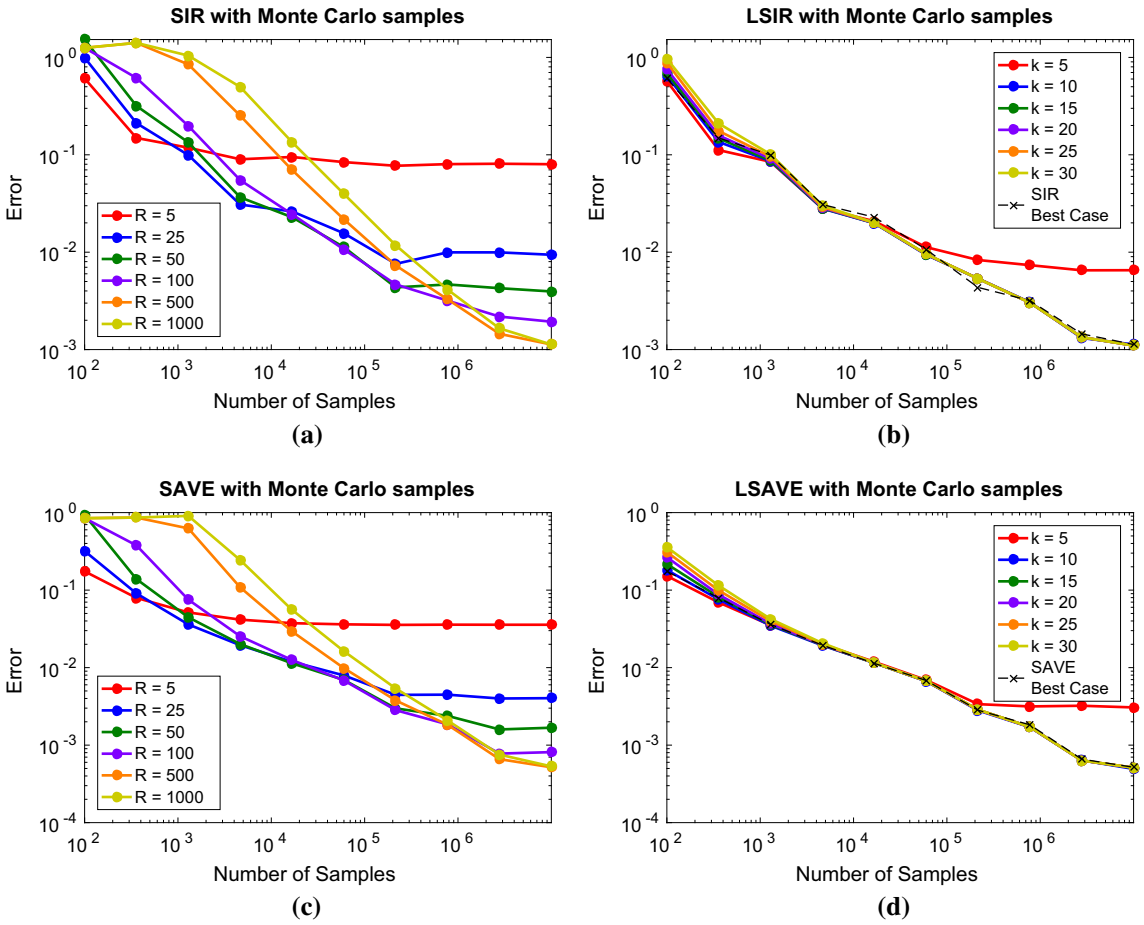


Fig. 6 A comparison of the SIR/SAVE and LSIR/LSAVE algorithms for (91) using Monte Carlo integration on the input space. **a** and **c** show the relative matrix errors of the SIR and SAVE algorithms, respectively, as a function of the number of samples for various values of R (the number of slices). **b** and **d** show the relative matrix errors of the LSIR and LSAVE algorithms, respectively, as a function of the number of samples for various values of k (the number of Lanczos iterations). These plots also show the best-case results from their slice-based counterparts for reference

imated SIR and SAVE matrices for increasing R to their Lanczos–Stieltjes counterparts. For the Lanczos–Stieltjes approximations—$\hat{\boldsymbol{C}}_{\mathrm{IR}}$ and $\hat{\boldsymbol{C}}_{\mathrm{AVE}}$—we use $N = 17^6 =$ 24,137,569 Gauss–Christoffel quadrature nodes and $k = 35$ Lanczos iterations as this produces sufficiently converged matrices; see the previous numerical study. For the SIR and

SAVE algorithms, we use $N = 10^8$ samples randomly drawn according to $\pi_{\mathbf{x}}$. Additionally, we define the slices adaptively based on the sampling to balance the number of samples in each slice as discussed in Sect. 2. The sliced approximations converge to their Lanczos–Stieltjes counterparts at a rate R^{-1} as expected for Riemann sums (Davis and Rabinowitz 1984, Ch. 2).

Lastly, we compare the slice-based algorithms SIR and SAVE to their Lanczos–Stieltjes counterparts LSIR and LSAVE with Monte Carlo integration on the input space. This comparison is the most appropriate for practical models with several input parameters, where tensor product quadrature on the input space is infeasible.

We again use the Lanczos–Stieltjes algorithms with $N = 17^6 = 24{,}137{,}569$ quadrature nodes and $k = 35$ Lanczos iterations as the "true" values of $\boldsymbol{C}_{\mathrm{IR}}$ and $\boldsymbol{C}_{\mathrm{AVE}}$ for computation of the relative matrix errors. Figure 6a, b shows the comparison of the SIR and LSIR algorithms (Algorithms 1 and 5, respectively) for increasing numbers of Monte Carlo samples. Additionally, we perform this comparison for various values of R (the number of terms in the Riemann sum or slices) and k (the number of Lanczos iterations) for each of the methods. We notice less variance in the LSIR plot as a function of k than in the SIR plot as a function of R. The Lanczos–Stieltjes approach refines the approximation over the output space such that the final approximation depends most strongly on the chosen integration rule over the input space. That is, the issue of choosing how to slice up the output space does not exist in the Lanczos–Stieltjes approach as the method automatically chooses the best integration rule over $\mathcal{F}$. Figure 6b also shows the best-case error from the SIR plot. This is the minimum error among all of the tested values of R for each value of N. The LSIR algorithm performs approximately as well as the best case in SIR, regardless of the value of k chosen. Figure 6c, d performs the same study as above comparing the SAVE and LSAVE algorithms (Algorithms 2 and 6, respectively). The results and interpretations are similar to those for the SIR/LSIR study.

7 Conclusion

We propose alternative approaches to the sliced inverse regression (SIR) and sliced average variance estimation (SAVE) algorithms for approximating $\boldsymbol{C}_{\mathrm{IR}}$ and $\boldsymbol{C}_{\mathrm{AVE}}$. The traditional methods approximate these matrices by applying a partitioning (i.e., slicing) to the range of output values. In the context of deterministic functions, this slice-based approach can be interpreted as a Riemann sum approximation of the integrals in (20). The proposed algorithms use orthonormal polynomials and Gauss–Christoffel quadrature to produce high-order approximations of $\boldsymbol{C}_{\mathrm{IR}}$ and $\boldsymbol{C}_{\mathrm{AVE}}$. We call the new algorithms Lanczos–Stieltjes inverse regression (LSIR) and Lanczos–Stieltjes average variance estimation (LSAVE).

We use two numerical test problems to study convergence of the Lanczos–Stieltjes algorithms with respect to the algorithm parameters. We first examine the convergence of the approximate quadrature and orthonormal polynomial components resulting from the Lanczos method's discrete approximation of the Stieltjes procedure. This study highlights the interplay between the number of quadrature nodes on the input space and the number of Lanczos iterations. More Lanczos iterations correspond to higher-degree polynomials, which require more samples for the same accuracy. Poor approximations of these polynomials lead to poor approximations of $\boldsymbol{C}_{\mathrm{IR}}$ and $\boldsymbol{C}_{\mathrm{AVE}}$. We then compare the Lanczos–Stieltjes approximations of $\boldsymbol{C}_{\mathrm{IR}}$ and $\boldsymbol{C}_{\mathrm{AVE}}$ to their slice-based counterparts. These numerical studies emphasize a key characteristic of the Lanczos–Stieltjes approaches. Due to the composite structure of $\boldsymbol{C}_{\mathrm{IR}}$ and $\boldsymbol{C}_{\mathrm{AVE}}$, both the slicing approach and Lanczos–Stieltjes contain two levels of approximation: (i) numerical integration on the input space $\mathcal{X}$ and (ii) approximation on the output space $\mathcal{F}$. There is a trade-off between (i) and (ii) in terms of which approximation is the dominant source of numerical error for various choices of N and R. The Lanczos–Stieltjes approach significantly reduces the errors due to approximation over $\mathcal{F}$, placing the burden of accuracy on the approximation over $\mathcal{X}$. This enables Gauss–Christoffel quadrature on $\mathcal{X}$ to produce high-order accuracy when such integration rules are appropriate—e.g., when the number of inputs is sufficiently small. When tensor product quadrature rules are infeasible, the Lanczos–Stieltjes approach allows Monte Carlo integration to perform as expected without significant dependence on the approximation on the output space $\mathcal{F}$.

References

Adragni, K.P., Cook, R.D.: Sufficient dimension reduction and prediction in regression. Philos. Trans. R. Soc. London A Math. Phys. Eng. Sci. **367**(1906), 4385–4405 (2009)

Ajtai, M., Janos, K., and Szemeredi, E.: An $\mathcal{O}(n \log n)$ sorting network. In: Fifteenth Annual ACM Symposium on Theory of Computing, pp. 1–9 (1983)

Ben-Ari, E.N., Steinberg, D.M.: Modeling data from computer experiments: an empirical comparison of kriging with mars and projection pursuit regression. Qual. Eng. **19**(4), 327–338 (2007)

Chang, J.T., Pollard, D.: Conditioning as disintegration. Stat. Neerlandica **51**(3), 287–317 (1997)

Clenshaw, C.W., Curtis, A.R.: A method for numerical integration on an automatic computer. Numer. Math. **2**(1), 197–205 (1960)

Constantine, P.G.: Active Subspaces: Emerging Ideas for Dimension Reduction in Parameter Studies. SIAM, Philadelphia (2015)

Constantine, P.G., Phipps, E.T.: A Lanczos method for approximating composite functions. Appl. Math. Comput. **218**(24), 11751–11762 (2012)

Constantine, P.G., Eldred, M.S., Phipps, E.T.: Sparse pseudospectral approximation method. Comput. Methods Appl. Mech. Eng. **229**, 1–12 (2012)
Cook, R.D.: Using dimension-reduction subspaces to identify important inputs in models of physical systems. In: Proceedings of the Section on Physical and Engineering Sciences, pp. 18–25. American Statistical Association, Alexandria, VA (1994)
Cook, R.D.: Regression Graphics: Ideas for Studying Regression through Graphics. Wiley, New York (1998)
Cook, R.D.: SAVE: a method for dimension reduction and graphics in regression. Commun. Stat. Theory Methods **29**(9–10), 2109–2121 (2000)
Cook, R.D., Forzani, L.: Likelihood-based sufficient dimension reduction. J. Am. Stat. Assoc. **104**(485), 197–208 (2009)
Cook, R.D., Ni, L.: Sufficient dimension reduction via inverse regression: a minimum discrepancy approach. J. Am. Stat. Assoc. **100**(470), 410–428 (2005)
Cook, R.D., Weisberg, S.: Sliced inverse regression for dimension reduction: comment. J. Am. Stat. Assoc. **86**(414), 328–332 (1991)
Davis, P.J., Rabinowitz, P.: Methods Numer. Integr. Academic Press, San Diego (1984)
de Boor, C., Golub, G.H.: The numerically stable reconstruction of a Jacobi matrix from spectral data. Linear Algebra Appl. **21**(3), 245–260 (1978)
Donoho, D.L.: High-dimensional data analysis: the curses and blessings of dimensionality. In: AMS Conference on Math Challenges of the 21st Century (2000)
Forsythe, G.E.: Generation and use of orthogonal polynomials for data-fitting with a digital computer. J. Soc. Ind. Appl. Math. **5**(2), 74–88 (1957)
Gautschi, W.: The interplay between classical analysis and (numerical) linear algebra–a tribute to Gene H. Golub. Electron. Trans. Numer. Anal. **13**, 119–147 (2002)
Gautschi, W.: Orthogonal Polynomials. Oxford Press, Oxford (2004)
Glaws, A., Constantine, P.G., and Cook, R.D.: Inverse regression for ridge recovery I: Theory. (2018) arXiv:1702.02227v2
Golub, G.H., Meurant, G.: Matrices, Moments, and Quadrature with Applications. Princeton University, Princeton (2010)
Golub, G.H., Van Loan, C.F.: Matrix Computations. JHU Press, Baltimore (1996)
Golub, G.H., Welsch, J.H.: Calculation of Gauss quadrature rules. Math. Comput. **23**(106), 221–230 (1969)
Jolliffe, I.T.: Principal Component Analysis. Springer, New York (2002)
Jones, D.R.: A taxonomy of global optimization methods based on response surfaces. J. Glob. Optim. **21**(4), 345–383 (2001)
Koehler, J.R., Owen, A.B.: Computer experiments. Handb. Stat. **13**(9), 261–308 (1996)
Lanczos, C.: An iteration method for the solution of the eigenvalue problem of linear differential and integral operators. J. Res. Natl. Bureau Stand. **45**(4), 255–282 (1950)
Li, B., Wang, S.: On directional regression for dimension reduction. J. Am. Stat. Assoc. **102**(479), 997–1008 (2007)
Li, K.C.: Sliced inverse regression for dimension reduction. J. Am. Stat. Assoc. **86**(414), 316–327 (1991)
Li, K.C.: On principal Hessian directions for data visualization and dimension reduction: another application of Stein's lemma. J. Am. Stat. Assoc. **87**(420), 1025–1039 (1992)
Li, K.C., Duan, N.: Regression analysis under link violation. Ann. Stat. **17**(3), 1009–1052 (1989)
Li, W., Lin, G., Li, B.: Inverse regression-based uncertainty quantification algorithms for high-dimensional models: theory and practice. J. Comput. Phys. **321**, 259–278 (2016)
Liesen, J., Strakoš, Z.: Krylov Subspace Methods: Principles and Analysis. Oxford Press, Oxford (2013)
Ma, Y., Zhu, L.: A review on dimension reduction. Int. Stat. Rev. **81**(1), 134–150 (2013)
Meurant, G.: The Lanczos and Conjugate Gradient Algorithms: From Theory to Finite Precision Computations. SIAM, Philadelphia (2006)
Myers, R.H., Montgomery, D.C.: Response Surface Methodology: Process and Product Optimization Using Designed Experiments. Wiley, New York (1995)
Owen, A.B.: Monte Carlo Theory, Methods and Examples (2013). http://statweb.stanford.edu/~owen/mc/
Pinkus, A.: Ridge Functions. Cambridge University Press, Cambridge (2015)
Sacks, J., Welch, W.J., Mitchell, T.J., Wynn, H.P.: Design and analysis of computer experiments. Stat. Sci. **4**(4), 409–423 (1989)
Saltelli, A., Ratto, M., Andres, T., Francesca Campolongo, J.C., Galtelli, D., Saisana, M., Tarantola, S.: Global Sensitivity Analysis. The Primer. Wiley, England (2008)
Santner, T.J., Williams, B.J., Notz, W.I.: The Design and Analysis of Computer Experiments. Springer, New York (2003)
Stieltjes, T.J.: Quelques recherches sur la théorie des quadratures dites mécaniques. Annales scientifiques de l' École Normale Supérieure **1**, 409–426 (1884)
Surjanovic, S., Bingham, D.: Virtual library of simulation experiments: test functions and datasets (2015). http://www.sfu.ca/~ssurjano
Traub, J.F., Werschulz, A.G.: Complexity and Information. Cambridge University Press, Cambridge (1998)
Trefethen, L.N.: Approximation Theory and Approximation Practice. SIAM, Philadelphia (2013)