Programming with equadratures: an open-source package for uncertainty quantification, dimension reduction, and much more

Pranay Seshadri*
The Alan Turing Institute, London, United Kingdom
Imperial College London, London, United Kingdom

Chun Yui Wong†
University of Cambridge, Cambridge, United Kingdom

Ashley Scillitoe‡ and Bryn Noel Ubald§
The Alan Turing Institute, London, United Kingdom

Barney Hill¶
University of Leeds, Leeds, United Kingdom

Irene Virdis‖ and Tiziano Ghisu**
University of Cagliari, Cagliari, Sardinia, Italy

Andrew B. Duncan††
The Alan Turing Institute, London, United Kingdom
Imperial College London, London, United Kingdom

This paper presents an overview of the open-source code equadratures. While originally developed to replicate polynomial chaos results seen in literature, it has since evolved to touch upon multiple aspects of computational engineering and machine learning. Today, the code uses orthogonal polynomial approximations to facilitate various parameter-based studies including uncertainty quantification, sensitivity analysis, dimension reduction, and classification. Additionally, it can address well-known limitations of polynomial approximations. These include the ability to fit to high-dimensional problems without requiring large input-output data pairs, and the ability to negotiate discontinues in any provided data. For the former, subspace-based polynomial approximations are employed, while for the latter, a tree-based piecewise polynomial hierarchy is adopted. Beyond this, ancillary topics such as coefficient computation, dealing with correlated inputs, moment computation, and gradient enhancement are also discussed. Following a deep-dive of the underpinning methods in the code, this paper details numerous case studies—with a slant towards computational aerodynamic problems.

I. Introduction

For nearly five years now, equadratures [1], a pure python code, has been made openly available to the computational methods community. Over the last two years alone, it has been downloaded well over 40,000 times, and is being used

*Research Fellow, Department of Mathematics, Imperial College London, and Group Leader in Aeronautics, Data-Centric Engineering, The Alan Turing Institute. Address all correspondence to p.seshadri@imperial.ac.uk

†PhD Research Student, Department of Engineering, University of Cambridge.

‡formerly Research Associate in Aeronautics, Data-Centric Engineering, The Alan Turing Institute.

§Research Associate in Aeronautics, Data-Centric Engineering, The Alan Turing Institute.

¶Undergraduate Student, Department of Mathematics, University of Leeds.

‖PhD Research Student, Department of Mechanical Engineering, University of Cagliari.

**Associate Professor, Department of Mechanical Engineering, University of Cagliari.

††Assistant Professor, Department of Mathematics, Imperial College London, and Group Leader in Data-Centric Engineering, The Alan Turing Institute.

across industry, government, and academia. Today, equadratures is a model development platform facilitating explainable and robust models that do not require large cloud infrastructure to train, unlike many deep learning frameworks. Models built in equadratures can be used for a wide range of tasks spanning uncertainty quantification, sensitivity analysis, numerical integration, optimisation, clustering, parameter-space exploration, dimension reduction, surrogate modelling, and even digital twinning.

When originally developed five years ago, equadratures (known then as Effective Quadratures) was purpose-built for facilitating non-intrusive uncertainty quantification through polynomial chaos expansions [2–4]. The unique selling point of the code was an adoption of least squares- and compressed sensing-based approaches for estimating polynomial coefficients, rather than opting for more conventional tensor and sparse grid strategies. This permitted greater anisotropy in the selection of basis functions used in the polynomial expansions, as well as a reduction in the number of samples required with rising problem dimensionality. The overall workflow can best be summarised in the steps below.

User prescribes input marginal distributions for each uncertainty.
$\Downarrow$
Least squares or compressed sensing framework is used to generate a design of experiment.
$\Downarrow$
Model is evaluated at the design of experiment.
$\Downarrow$
Polynomial coefficients are computed.
$\Downarrow$
Moments and sensitivities (Sobol' indices) are determined from the coefficients (see [5, 6]).

This shift towards least squares and compressed sensing is well captured in literature [7, 8] as are the different strategies for arriving at well conditioned matrices and sampling distributions [9–11]. There is a clear trend to opt for random sampling techniques paired with well-worn algorithms for identifying well-conditioned submatrices, e.g., QR with column pivoting and SVD-based subset selection [12], among other convex optimisation rooted techniques (see [13] for a comparison).

Over the past few years, equadratures has grown in capability, ingesting and synthesising key advances in literature to accelerate application impact. One of the most fruitful advances have been in parameter-space dimension reduction, where the central idea is to ascertain whether a function admits a dimension reducing subspace—i.e., a few linear combination of all the variables which may be used for function approximation. While the idea is itself not unprecedented [14], it has experienced a resurgence owing to a prodigious number of publications under the handles of active subspaces [15], sufficient dimension reduction [16] and ridge functions [17] (to name a few). These works have been championed by researchers spanning both academia and industry—with impactful use cases [18–21] that likely serve as a precursor to further advances within the areas of function approximation. A practical outlook on the success of data-driven dimension reduction in computational science may be enforced by the notion that we trust our models within a relatively small parameter space and conduct our parameter studies accordingly. Thus, function values around a notional centroid may be well-approximated via linear projections of neighboring parameter values.

Beyond dimension reduction, ancillary progress on robust methods for dealing with correlations in the inputs, i.e., identifying independent polynomial basis on correlated spaces [22] has been important for driving forth the uptake of polynomial-based methods and thus equadratures. This builds upon prior work with Nataf and Rosenblatt transformations [23]. These advances have permitted the construction of stable *global* polynomials across both high order and high dimensional problems. This naturally leads one to consider leveraging polynomials across a wider range of challenges including optimisation, multi-fidelity modelling, spatial-field modelling and dimension reduction.

It is important to remark that *global* smoothness and continuity are certainly not guaranteed for all problems. Thus, strategies to fit multiple polynomials over a domain are extremely useful, especially when working with problems that are characterised by a relatively large parameter space. This may take the form of trust region methods [24] or tree-based methods [25], where polynomials need to be defined over a subdomain in a recursive manner—based on certain approximation error criterion. Within equadratures these ideas have found utility in polynomial regression tree models and trust-region optimisation methods. In fact for the latter, if one further assumes that a subspace-based dimension reduction representation exists, then from the perspective of optimiser, as the trust region migrates through a larger parameter space, finding such projections iteratively may be incredibly valuable, both from the perspectives of convergence rate and optimality (see [26]).

The rather terse review above sets the stage for our paper, which has two main goals. First, we want to communicate how the building blocks in equadratures—i.e., classes and functions—are aligned with the underpinning mathematics

these utilities encapsulate, providing a rich prototyping framework for novice users and experienced practitioners. Second, we wish to demonstrate how problems well-beyond the standard uncertainty quantification mould can be adequately addressed by using equadratures. The remainder of this paper is structured as follows. The remainder of this paper is structured as follows. In II we introduce the core building blocks in `equadratures`. This is followed by III where we outline strategies for computing coefficients, with gradient enahncement, and under correlations. Section IV details how equadratures exploits parameter subspace-based dimension reduction approaches. This may be used to efficiently compute moments and sensitivities V particularly for high dimensional problems. Techniques for constructing piecewise polynomials using trees is presented in VII, followed by case studies in VIII.

II. Parameter, Basis, and Polynomials

The three key fundamental building blocks in equadratures are classes by the names of `Parameter`, `Basis` and `Polynomial`. In this section we detail their mathematical definitions and the role their namesake classes play in equadratures.

A. Parameter

Define a scalar-valued quantity of interest (qoi) as $f = f(\boldsymbol{x})$ where $\boldsymbol{x} = (x^{(1)}, \ldots, x^{(d)})$ is a d-dimensional vector of mutually independent variables. Thus the function f is a map from $\mathbb{R}^d \rightarrow \mathbb{R}$. Here, each parameter $x^{(i)}$ belongs to a domain $\mathcal{X}^{(i)} \in \mathbb{R}$, that can be an infinite interval $(-\infty, \infty)$, a semi-infinite interval $(-\infty, b]$ or $[a, \infty)$, or a closed (finite) interval $[a, b]$. This interval represents the support of each parameter. We generally consider the input domain to be a non-compact hypercube decomposed as a Cartesian product of the form $\mathcal{X} = \mathcal{X}^{(1)} \times \cdots \times \mathcal{X}^{(d)}$.

Along each interval, consider a positive weight function $\rho_i\left(x^{(i)}\right) > 0$ over the domain $\mathcal{X}^{(i)}$ such that

$$\int_{\mathcal{X}^i} \left(x^{(i)}\right)^k \rho_i\left(x^{(i)}\right) dx^{(i)} < \infty, \quad \text{where} \quad \int_{\mathcal{X}^i} \rho_i\left(x^{(i)}\right) dx^{(i)} = 1, \tag{1}$$

for $k = 1, 2, \ldots$, and where $i = 1, \ldots, d$. The second expression in (1) naturally leads to the interpretation that $\rho_i\left(x^{(i)}\right)$ is a probability density function over the domain $\mathcal{X}^{(i)}$. In making the assumption that $\boldsymbol{x}$ is a vector of independent variables, the joint density $\boldsymbol{\rho}$ of all the probability distributions associated with $\boldsymbol{x}$ is given by

$$\boldsymbol{\rho}(\boldsymbol{x}) = \prod_{i=1}^{d} \rho_i\left(x^{(i)}\right), \tag{2}$$

defined on $\mathbb{R}^d$; should the variables be correlated then (2) no longer holds.

B. Basis

A multi-index is a set of indices that enumerates composite univariate polynomial orders that are involved in forming the multivariate basis. As we intend to approximate f via a finite number of polynomials $\boldsymbol{\phi_i}$, we restrict indices $\boldsymbol{i}$ to lie in a finite multi-index set $\mathcal{I}$. Whilst considerable flexibility in specifying these multi-index sets exists, well-known $\mathcal{I}$ include tensor product index sets, total order index sets, Euclidean degree spaces [27], and hyperbolic spaces [28]. Each of these index sets in d dimensions is well-defined given a fixed $k \in \mathbb{N}_0$, which indicates the maximum polynomial degree associated to these sets. Tensor product index sets are characterised by

$$\max_k i_k \leq k, \tag{3}$$

and have a cardinality (number of elements) equal to $(k+1)^d$. Total order index sets contain multi-indices satisfying

$$\sum_{j=1}^{d} i_j \leq k \tag{4}$$

They effectively disregard some higher order interactions between dimensions present in tensorised spaces, and a total order index set $\mathcal{I}$ has cardinality

$$|\mathcal{I}| = \begin{pmatrix} k+d \\ k \end{pmatrix}. \tag{5}$$

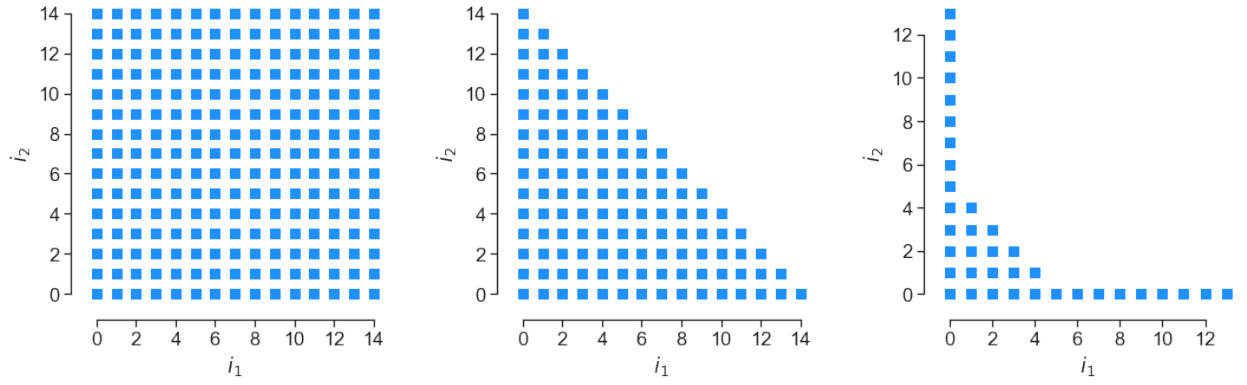


Fig. 1 Multi-indices for $d = 2$ with a maximum univariate degree of 14 for: (a) tensor order; (b) total order; (c) hyperbolic space.

For a hyperbolic space index set,

$$\left(\sum_{j=1}^{d} i_j^q\right)^{1/q} \leq k, \tag{6}$$

where q is a user-defined constant that can be varied from 0.2 to 1.0. When this parameter is set to unity $q = 1$, the hyperbolic index space is equivalent to a total order index space. For values less than unity higher-order interactions terms are eliminated as discussed before [28].

For illustrative purposes, plots of the main multi-index sets associated with $d = 2$ and with a maximum univariate degree of 14 are shown in Figure 1. Of particular importance is the growth and interaction of the higher order terms. In total order and hyperbolic spaces, higher order interaction terms are significantly reduced. For many physical systems, these type of *sparser* basis have found utility as lower order interactions are often far more dominant. Another important consideration when choosing a multi-index is the level of anisotropy along each dimension. For instance, one can have different maximum degrees in each direction.

C. Polynomial

Assume that f is sufficiently smooth and continuous such that it can be approximated by a global polynomial p

$$\begin{aligned} f(\boldsymbol{x}) &\approx p(\boldsymbol{x}) \\ &= \sum_{j=1}^{N} \alpha_j \boldsymbol{\phi}_j(\boldsymbol{x}), \\ &= \boldsymbol{P}\boldsymbol{\alpha} \end{aligned} \tag{7}$$

defined as a weighted sum of N known basis polynomials, where

$$\boldsymbol{\phi}_j(\boldsymbol{x}) = \prod_{k=1}^{d} \phi_{j_k}^{(k)}\left(x^{(k)}\right), \qquad (j_1, \ldots, j_d) \in \mathbb{N}_0^d, \tag{8}$$

and where $[\boldsymbol{P}]_{i,j} = \boldsymbol{\phi}_j(\boldsymbol{x}_i)$ for some discretised value $\boldsymbol{x}_i \in \mathcal{X}$. Note that the polynomials $\boldsymbol{\phi}_j$ defined in this way are mutually orthogonal in L^2 weighted by $\boldsymbol{\rho}$. More specifically, we can state that these composite univariate polynomials must satisfy

$$\int_{\mathcal{X}^k} \phi_g\left(x^{(k)}\right) \phi_h\left(x^{(i)}\right) \rho_k\left(x^{(k)}\right) dx^{(k)} = \delta_{gh}, \tag{9}$$

where δ_{gh} is the Kronecker delta; subscripts g and h denote polynomial orders. The above expression crystallizes the choice of the orthogonal polynomial family based on the choice of the weight function $\rho_k\left(x^{(k)}\right)$. For instance if $\rho_k\left(x^{(k)}\right)$

were the uniform distribution with $\mathcal{X}^{(k)} \in [a, b]$ then $\left\{\phi_j\left(x^{(k)}\right)\right\}_{j=0}^{\infty}$ would correspond to Legendre polynomials; for Gaussian weights one would use Hermite polynomials, and so on. Details about these weight-polynomial pairs can be found in [2]. Succinctly stated, choosing the correct weight-polynomial pairs will lead to an exponential convergence in (7) thereby reducing the number of terms N required. As a consequence, this will also reduce the number of model evaluations needed to estimate the unknown coefficients $(\alpha_1, \ldots, \alpha_N)$.

Before moving on to the computation of the coefficients, a few words on the computation of the individual univariate orthogonal polynomials $\phi_j\left(x^{(k)}\right)$ are in order. Orthogonal polynomials of any order may be given by a three-term recurrence relation

$$\beta_{j+1}\phi_{j+1}\left(x^{(k)}\right) = \left(x^{(k)} - \alpha_j\right)\phi_j\left(x^{(k)}\right) - \beta_j\phi_{j-1}\left(x^{(k)}\right), \quad \text{for} \quad j = 0, 1, 2, 3, \ldots \tag{10}$$

where (α_j, β_j) are a set of recurrence coefficients explicitly known, dependent upon $\rho_k\left(x^{(k)}\right)$ [29]. A four-term recurrence formula can also be derived for the derivatives of orthogonal polynomials.

III. Coefficient Computation

These coefficients are defined to be

$$\alpha_j = \int_{\mathcal{X}} f(\boldsymbol{x})\,\boldsymbol{\phi}_j(\boldsymbol{x})\,\boldsymbol{\rho}(\boldsymbol{x})\,d\boldsymbol{x} \tag{11}$$

which may be interpreted as the inner product of the function over the j-th polynomial term. The overarching goal in this section will be the utilisation of least squares and compressed sensing rooted methods for approximating (11) via

$$\alpha_j \approx \sum_{i=1}^{M} f(\boldsymbol{\chi}_i)\,\boldsymbol{\phi}_j(\boldsymbol{\chi}_i)\,\omega_i \tag{12}$$

using points $\{\boldsymbol{\chi}_i\}_{i=1}^{M} \in \mathcal{X}$, and a set of corresponding weights $\{\omega_i\}_{i=1}^{M}$ defined via

$$\omega_i = \frac{\tilde{\omega}_i}{\sum_{i=1}^{M}\tilde{\omega}_i}, \quad \text{where} \quad \tilde{\omega}_i = \frac{1}{\sum_{j=1}^{N}\boldsymbol{\phi}_j(\boldsymbol{\chi}_i)^2}. \tag{13}$$

Note that if the points $\{\boldsymbol{\chi}_i\}_{i=1}^{M} \in \mathcal{X}$ selected are quadrature points, then the weights $\{\omega_i\}_{i=1}^{M}$ will be equivalent to the corresponding quadrature weights. To solve (12) for all j's, we assume access to input-output model evaluations of the form $\{\boldsymbol{\chi}_i, f_i\}_{i=1}^{M}$.

In equadratures coefficient computation strategies are passed as string input methods to the `Polynomial` class and include `least-squares`, `compressed-sensing` and `relevance-vector-machine`.

A. Least squares

It will be useful to think of techniques for estimating the coefficients $\boldsymbol{\alpha} = (\alpha_1, \ldots, \alpha_N)^T$ in terms of matrix-vector operations. Define $\boldsymbol{A} \in \mathbb{R}^{M\times N}$, formed by evaluating multivariate orthogonal polynomials at a set of points $\boldsymbol{\chi_i}$ weighted by $\boldsymbol{\omega_i}$

$$[\boldsymbol{A}]_{i,j} = \boldsymbol{\phi}_j(\boldsymbol{\chi}_i)\sqrt{\omega_i}, \tag{14}$$

with M quadrature points and N polynomials. We also define a Gram matrix $\boldsymbol{G} = \boldsymbol{A}^T\boldsymbol{A}$, where each element is effectively given by

$$[\boldsymbol{G}]_{p,q} = \sum_{i=1}^{M}\boldsymbol{\phi_p}(\boldsymbol{\chi}_i)\,\boldsymbol{\phi_q}(\boldsymbol{\chi}_i)\,\omega_i \approx \int \boldsymbol{\phi_p}(\boldsymbol{x})\,\boldsymbol{\phi_q}(\boldsymbol{x})\,\boldsymbol{\rho}(\boldsymbol{x})\,d\boldsymbol{x} = \delta_{\boldsymbol{pq}}. \tag{15}$$

Should the points and weights be appropriately selected, then one would expect $\boldsymbol{G}$ to be very close to the $N \times N$ identity matrix. In fact if the correct numerical quadrature rule—correct in the degree of exactness sense—is chosen, then $\boldsymbol{G}$ will be equivalent to the identity matrix. There is a direct connection between this fact and the solution to the least squares problem

$$\underset{\boldsymbol{\alpha}\in\mathbb{R}^N}{\text{minimize}} \quad \|\boldsymbol{A}\boldsymbol{\alpha} - \mathbf{b}\|_2, \tag{16}$$

where entries in $\mathbf{b} \in \mathbb{R}^M$ are weighted evaluations of f at the chosen points, i.e., $\mathbf{b}(i) = \sqrt{\omega_i} f(\boldsymbol{\chi}_i)$. Assuming that $\boldsymbol{A}$ is not rank deficient, the solution to (16) is given by the *normal equations* $\boldsymbol{\alpha}^* = \left(\boldsymbol{A}^T\boldsymbol{A}\right)^{-1}\boldsymbol{A}^T\mathbf{b} = \boldsymbol{G}^{-1}\boldsymbol{A}^T\mathbf{b}$. A few remarks regarding this expression are in order. First, if the Gram matrix $\boldsymbol{G}$ is not the identity, then we will likely encounter aliasing errors when solving for the coefficients $\boldsymbol{\alpha}$. Second, to obtain solutions to this least squares problem we require $\boldsymbol{A}$ to be a tall matrix, or at the very least a square matrix. The key to identifying well-conditioned $\boldsymbol{A}$ when solving for the coefficients is a suitable selection of points and weights $\{\boldsymbol{\chi}_i, \omega_i\}_{i=1}^M$.

In [13] (and references therein) the authors breakdown the problem into two components. The first, is to generate a suitable sample of potential points based on the measure $\boldsymbol{\rho}$ and the choice of the basis $\mathcal{I}$. The standard approach is to sample from the measure $\boldsymbol{\rho}$ itself, however this can be suboptimal, as the convergence rate is rather slow in terms of the number of samples required. Another strategy, that both in theory and practice offers a better set of samples is based on the *induced distribution* $\boldsymbol{\rho}_V$. One generates independent identically distributed (iid) samples from this distribution, given by

$$\boldsymbol{\rho}_V(\boldsymbol{x}) = \sum_{i=1}^{n} \boldsymbol{\phi}_i^2(\boldsymbol{x})\,\boldsymbol{\rho}(\boldsymbol{x}), \tag{17}$$

and then uses the modified weights

$$\omega_{V,j} = \frac{\boldsymbol{\rho}}{\boldsymbol{\rho}_V}(\boldsymbol{\chi}_j) = \frac{1}{\sum_{i=1}^{n} \boldsymbol{\phi}_i^2(\boldsymbol{\chi}_j)}. \tag{18}$$

The challenge with this strategy is that the precise analytical forms for the induced distribution are not known for all $\boldsymbol{\rho}$ and multi-index sets $\mathcal{I}$ (see [30] and [31] for further details). We remark here that these sampling strategies are inherently non-deterministic. A deterministic sampling recipe can be found in the work of [4], where the authors subsample points from a tensor grid to approximate the coefficients associated with a total order basis. The idea is to construct $\boldsymbol{A}$ such that $M >> N$ and then prune down the number of samples using an optimization strategy. One well-known optimisation heuristic based on QR column pivoting seeks to subselect rows of $\boldsymbol{A}$ that best reduce the condition number. Other optimization strategies based on Newton's method and linear programming have also been discussed in literature, such as [32], [13] and [33].

To illustrate some of the benefits in moving away from standard quadrature rules, consider the point distributions $\boldsymbol{\xi}$ and coefficient values $\boldsymbol{\alpha}$ shown in Figure 2 for the function $f(\boldsymbol{x}) = \exp\left(x^{(1)} + 3x^{(2)}\right)$. These results are generated with a maximal order of 16 along both dimensions; Figure 2(a) is a tensor grid, (c) is a sparse grid with an exponential growth rule and a level of 4, (e) represents points that were subsampled from the tensor grid using QR column pivoting with a total order basis, while (g) is subsampled with a hyperbolic basis. The number of points in these figures are 289, 147, 153 and 84 respectively. Subplots (b), (d), (f) and (h) show the coefficient values in a base-10 logarithm scale. While the top row of this figure represents the results of numerical integration, the bottom row shows the results with least squares—offering far greater freedom in the basis terms selected, and in this case, with the same number of points as basis terms.

B. Compressed sensing strategies

Least squares solutions place a restriction on the polynomial evaluation matrix $\boldsymbol{A}$: there must be at least as many rows as columns. In this section, we focus on heuristics that permit bypassing this restriction. Under this condition, the possible solutions to $\boldsymbol{A}\boldsymbol{\alpha} = \mathbf{b}$ form an affine space, all of which are exactly accurate solutions when evaluated with the mean squared error. Thus, from this space, one must select the most plausible solution with some other criterion or constraint.

Assuming that the solution is *sparse*, i.e. with many zeros or near-zeros, it can be shown that heuristics based on ℓ_1-minimisation can be used to recover solutions given structural assumptions on the matrix $\boldsymbol{A}$, such as *incoherence*, which enforces conditions on the points $\boldsymbol{\chi}$. Using this approach, the sparsest compatible solution is found by solving the optimization problem known as the *basis pursuit denoising problem* ([34])

$$\begin{aligned} &\underset{\boldsymbol{\alpha}\in\mathbb{R}^N}{\text{minimise}} \ \|\boldsymbol{\alpha}\|_1 \\ &\text{subject to } \|\boldsymbol{A}\boldsymbol{\alpha} - \mathbf{b}\|_2 \le \epsilon, \end{aligned} \tag{19}$$

where ϵ is small positive value representing truncation error of the polynomial series and/or noise in observation data (see [35]). Second order conic program strategies for solving (19) are well-known; they lend themselves to solutions via interior point methods [36, 37].

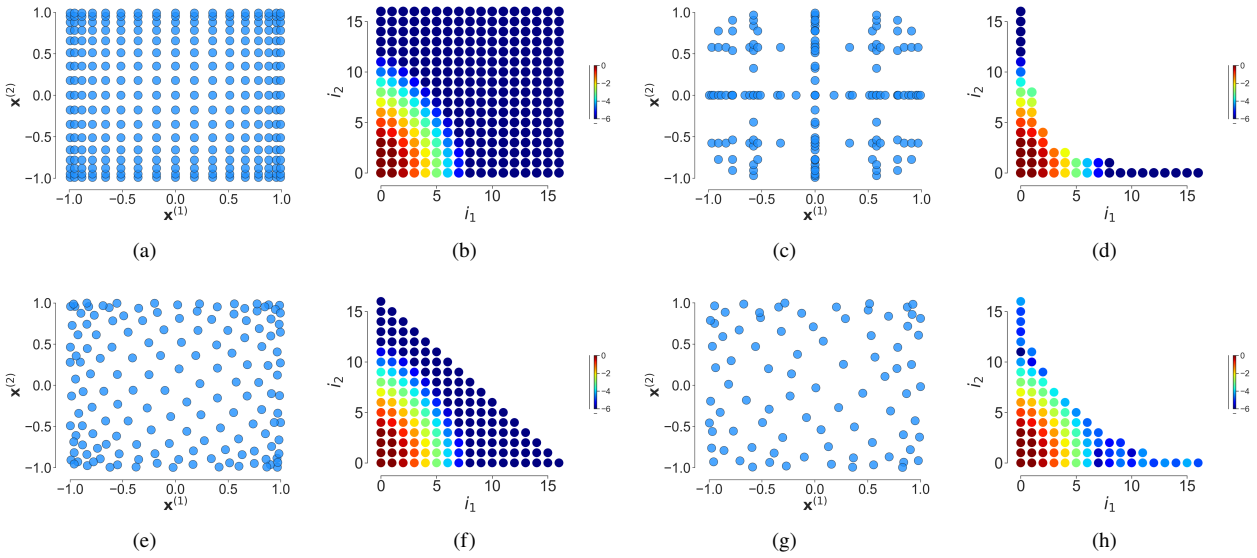


Fig. 2 Sample point distributions and coefficient estimates for: (a, b) tensor grid with maximal order of 16; (b, c) sparse grid with a level of 4 and an exponential growth rule; (e, f) subsampled points with a total order basis; (g, h) subsampled points with a hyperbolic basis.

As with least squares strategies, a key question that arises pertains to the sampling strategy for constructing $\boldsymbol{A}$. While compressed sensing strategies typically perform well with random sampling, there has been much work on devising sampling methods. However, unlike least squares techniques where a single measure that can quantify the overall *quality* of a matrix can be devised (e.g., the condition number), establishing such an easily computable metric remains a challenge [38]. We omit a theoretical exposition of the various sampling strategies for and refer the reader to Tang and Iaccarino [39] and Hampton and Doostan [35].

In practice, the use of ℓ_1-minimisation methods can give slow performance. The main bottleneck to the method is the determination of the unknown ϵ. Depending on the scale of the problem, it can take a wide range of values. Even when the output is normalized, several plausible values of ϵ still need to be tested with trial-and-error. This in turn implies that multiple optimization problems need to be solved for one regression task. Below we describe an alternative approach for underdetermined sparse regression that obviates the need for hyperparameter searching with cross validation, namely relevance vector machines (RVMs).

The RVM is first proposed by Tipping [40] and introduced as a Bayesian method for compressed sensing [41]. This method considers the task of coefficient computation in a probabilistic framework. The regression model is formulated using a generative process with Gaussian noise ε,

$$f(\boldsymbol{x}) \approx \boldsymbol{\alpha}^T \boldsymbol{\phi}(\mathbf{x}) + \varepsilon = y,$$

where $\varepsilon \sim \mathcal{N}(0, \sigma^2)$. The likelihood of the data is then a Gaussian,

$$\mathbb{P}(\mathbf{y} \mid \boldsymbol{\alpha}, \mathbf{X}, \sigma^2) = \mathcal{N}(\mathbf{y} \mid \boldsymbol{A}\boldsymbol{\alpha}, \sigma^2 \mathbf{I}).$$

The prior placed on the coefficients is a zero mean Gaussian, where the variances for each coefficient is specified with a hyperparameter κ_i.

$$\mathbb{P}(\boldsymbol{\alpha} \mid \boldsymbol{\kappa}) = \prod_{i=1}^{P} \mathcal{N}(\alpha_i \mid 0, \kappa^{-1})$$

The hyperparameters $\boldsymbol{\kappa}$ and $\beta := \sigma^{-2}$ are in turn specified by another prior distribution in the form of Gamma distributions

$$\mathbb{P}(\boldsymbol{\kappa}) = \prod_{i=1}^{P} \Gamma(\kappa \mid a, b)$$

$$\mathbb{P}(\beta) = \Gamma(\beta \mid d, e)$$

To simplify our expressions, we set $a = b = d = e = 0$ for an uninformative prior for the hyperparameters. Using Bayes' rule, The posterior distribution of all parameters is

$$\mathbb{P}(\boldsymbol{\alpha}, \boldsymbol{\kappa}, \beta \mid \mathbf{y}) = \frac{\mathbb{P}(\mathbf{y} \mid \boldsymbol{\alpha}, \boldsymbol{\kappa}, \beta)\mathbb{P}(\boldsymbol{\alpha}, \boldsymbol{\kappa}, \beta)}{\mathbb{P}(\mathbf{y})}.$$

For a full Bayesian treatment (e.g. getting the predictive mean and variance, not simply the maximum a posteriori estimator), the normalising term $\mathbb{P}(\mathbf{y}) = \int \mathbb{P}(\mathbf{y} \mid \boldsymbol{\alpha}, \boldsymbol{\kappa}, \beta)\ \mathbb{P}(\boldsymbol{\alpha}, \boldsymbol{\kappa}, \beta)\ d\boldsymbol{\alpha}\ d\boldsymbol{\kappa}\ d\beta$ is required. However, this high-dimensional integral cannot be computed—a characteristic of many Bayesian inference problems. In RVM, the posterior is decomposed into two terms

$$\mathbb{P}(\boldsymbol{\alpha}, \boldsymbol{\kappa}, \beta \mid \mathbf{y}) = \mathbb{P}(\boldsymbol{\alpha} \mid \boldsymbol{\kappa}, \beta, \mathbf{y})\ \mathbb{P}(\boldsymbol{\kappa}, \beta \mid \mathbf{y}).$$

The first term is a Gaussian,

$$\mathbb{P}(\boldsymbol{\alpha} \mid \boldsymbol{\kappa}, \beta, \mathbf{y}) = \mathcal{N}(\boldsymbol{\alpha} \mid \boldsymbol{\mu}, \boldsymbol{\Sigma}),$$

where

$$\boldsymbol{\Sigma} = (\beta \boldsymbol{A}^T \boldsymbol{A} + \mathbf{D})^{-1},$$
$$\boldsymbol{\mu} = \beta \boldsymbol{\Sigma} \boldsymbol{\Phi}^T \mathbf{y},$$

with $\mathbf{D} = \text{diag}(\kappa_1, ..., \kappa_P)$. The second term requires some further simplification. Since it is non-Gaussian, an approximation is made by approximating it as a point-mass centred at the maximum. To find the maximum, we observe that

$$\mathbb{P}(\boldsymbol{\kappa}, \beta \mid \mathbf{y}) \propto \mathbb{P}(\mathbf{y} \mid \boldsymbol{\kappa}, \beta)\ \mathbb{P}(\boldsymbol{\kappa})\ p(\beta),$$

and under the uniform priors for both $\boldsymbol{\kappa}$ and β only the first term needs to be considered. It is a Gaussian with respect to $\mathbf{y}$ but not to $\boldsymbol{\kappa}$ and β,

$$\mathbb{P}(\mathbf{y} \mid \boldsymbol{\kappa}, \beta) = \mathcal{N}(\mathbf{y} \mid 0, \beta^{-1}\mathbf{I} + \boldsymbol{\Phi}\mathbf{D}\boldsymbol{\Phi}^T)$$

The learning task for RVM is then to find the appropriate values of $\boldsymbol{\kappa}$ and β that maximizes this expression. Differentiating $\mathbb{P}(\mathbf{y} \mid \boldsymbol{\kappa}, \beta)$ with respect to the hyperparameters, their stationary points can be found in terms of $\boldsymbol{\mu}$ and $\boldsymbol{\Sigma}$. Thus, an iterative approach is used to find the maximising hyperparameters. Empirically, it is observed that many of the α_i's tend to infinity as the minimisation routine progresses.

C. Gradient enhancement

The ability to utilise gradients when constructing polynomials is beneficial, particularly when there can be a net reduction in the computational cost. From (7), we can estimate the gradients of f to be

$$\begin{aligned} \nabla_{\boldsymbol{x}} f\ (\boldsymbol{x}) &\approx \nabla_{\boldsymbol{x}} p\ (\boldsymbol{x}) \\ &= \sum_{j=1}^{N} \alpha_j \nabla_{\boldsymbol{x}} \boldsymbol{\phi}_j\ (\boldsymbol{x}) \\ &= \begin{bmatrix} \nabla_{x^{(1)}} \boldsymbol{P}\ (\boldsymbol{x}) \\ \vdots \\ \nabla_{x^{(d)}} \boldsymbol{P}\ (\boldsymbol{x}) \end{bmatrix} \boldsymbol{\alpha} \end{aligned} \tag{20}$$

where each block in the matrix above is given by

$$\nabla_{x^{(k)}} \boldsymbol{P}\ (\boldsymbol{x}) = \frac{\partial \phi_{j_k}^{(k)}\ \left(x^{(k)}\right)}{\partial x^{(k)}} \left(x^{(k)}\right) \prod_{q=1, q\neq k}^{d} \phi_{j_k}^{(k)} \left(x^{(k)}\right) \tag{21}$$

for $k = 1, \ldots, d$. In this subsection, we discuss an approach for leveraging gradient (adjoint) evaluations for estimating the unknown coefficients $\boldsymbol{\alpha}$, bearing in mind that the addition of gradients should abate the total number of model evaluations required. To clarify, consider fitting a polynomial in $d = 10$ with a total order basis with maximal degree 2,

which has 66 coefficients. With least squares, we would require atleast 66 model evaluations if not more. The intention behind using gradient evaluations would be to significantly reduce the number of model evaluations required.

While prior approaches in literature have adopted a *stacked* weighted least squares approach to this problem—where the gradient orthogonal polynomial matrices in (20) are stacked with the polynomial matrix (see (7))—the methodology used in equadratures is based on the work in [42] where a constrained least squares approach is used. To clarify this, let us assume access to input, output, and gradient evaluations $\{\boldsymbol{\chi}_i, f_i, \nabla_{\boldsymbol{x}} f_i\}_{i=1}^{R}$. As the gradient evaluations are likely to be more noisy, the following problem is solved

$$\underset{\boldsymbol{\alpha}}{\text{minimise}} \, \|\boldsymbol{C}\boldsymbol{\alpha} - \mathbf{d}\|_2^2, \quad \text{subject to} \quad \boldsymbol{A}\boldsymbol{\alpha} - \mathbf{b} \tag{22}$$

where $\boldsymbol{A} \in \mathbb{R}^{R\times N}$, $\mathbf{b} \in \mathbb{R}^{R}$, $\boldsymbol{C} \in \mathbb{R}^{Rd\times N}$, $\mathbf{d} \in \mathbb{R}^{Rd}$, with

$$\boldsymbol{C} = \begin{bmatrix} \boldsymbol{C}^{(1)} \\ \vdots \\ \boldsymbol{C}^{(d)} \end{bmatrix}, \quad \left[\boldsymbol{C}^{(k)}\right]_{ij} = \sqrt{\omega_i}\nabla_{x^{(k)}} \boldsymbol{P}_j\left(\boldsymbol{\xi}_i\right), \quad \text{and} \quad \mathbf{d} = \begin{bmatrix} \mathbf{d}^{(1)} \\ \vdots \\ \mathbf{d}^{(d)} \end{bmatrix}, \quad \left[\mathbf{d}^{(k)}\right]_{i} = \sqrt{\omega_i}\nabla_{x^{(k)}} f_i. \tag{23}$$

Solutions to (22) are obtained via the null-space method (see Chapter 2 in [12]).

IV. Subspaces

In this section, we assume that $f \approx p\left(\boldsymbol{U}^T\mathbf{x}\right)$, with $\boldsymbol{U} \in \mathbb{R}^{d\times n}$ where $n < d$ for some function m. This type of approximation is known as a polynomial ridge approximation [17, 43, 44] and it explicitly embeds $\boldsymbol{x}$ into a lower dimensional submanifold. We define the set of all n-dimensional subspaces in $\mathbb{R}^d$ to be the Grassmann manifold, denoted by $\mathbb{G}\left(n, d\right)$. If we add the additional constraint that we require the columns of $\boldsymbol{U}$ to be orthogonal, then we refer to the submanifold as the Stiefel manifold $\mathbb{S}t\left(n, d\right)$.

For low values of n, i.e., one or two, the columns of $\boldsymbol{U}$ permit visualisation of the discrete input-output data pairs $\{\mathbf{x}_i, f_i\}_{i=1}^{M}$. This sub-manifold can deliver tremendous insight into the scalar-valued quantity of interest, while offering tractable avenues for parameter studies. Discovering such manifold structure through affordable computational algorithms has been the goal of many papers and will be discussed below. In the code, all subspace-based functionality is encapsulated in the `Subspaces` class.

A. Active subspaces from polynomials

Constantine [15] draws our attention to a symmetric positive semidefinite $d \times d$ covariance matrix formed by the average outer product of the gradient with itself

$$\boldsymbol{C} = \int_{\mathcal{X}} \nabla_{\boldsymbol{x}} f\left(\boldsymbol{x}\right) \nabla_{\boldsymbol{x}} f\left(\boldsymbol{x}\right)^T \boldsymbol{\rho}\left(\boldsymbol{x}\right) d\boldsymbol{x}. \tag{24}$$

The eigenvectors $\boldsymbol{Q}$ of this matrix can be obtained via the eigendecomposition $\boldsymbol{C} = \boldsymbol{Q}\boldsymbol{\Lambda}\boldsymbol{Q}^T$, and can reveal important linear combinations along which the function's gradients have large values (on average). The eigenvalues of $\boldsymbol{C}$ are given as diagonal entries in $\boldsymbol{\Lambda}$ and can offer insight into the number of *true* orthogonal directions required. This is particularly useful for designing low-dimensional surrogate models that can in turn be used for uncertainty quantification and optimisation. One approach, although not necessarily optimal, is to set $\boldsymbol{U}$ to be the first $n < d$ columns of $\boldsymbol{Q}$, and use standard regression techniques to estimate the coefficients of $p\left(\boldsymbol{U}^T\boldsymbol{x}\right)$.

For numerous computational problems where adjoints or accurate gradient estimates are unavailable, one can easily evaluate f's polynomial approximation to obtain

$$\begin{aligned} \boldsymbol{C} &\approx \int_{\mathcal{X}} \nabla_{\boldsymbol{x}} p\left(\boldsymbol{x}\right) \nabla_{\boldsymbol{x}} p\left(\boldsymbol{x}\right)^T \boldsymbol{\rho}\left(\boldsymbol{x}\right) d\boldsymbol{x} \\ &= \int_{\mathcal{X}} \left(\sum_{j=1}^{N} \alpha_j \nabla_{\boldsymbol{x}} \boldsymbol{\phi}_j\left(\boldsymbol{x}\right)\right)\left(\sum_{j=1}^{N} \alpha_j \nabla_{\boldsymbol{x}} \boldsymbol{\phi}_j\left(\boldsymbol{x}\right)\right)^T \boldsymbol{\rho}\left(\boldsymbol{x}\right) d\boldsymbol{x}, \end{aligned} \tag{25}$$

which may be numerically evaluated through quadrature or Monte Carlo. It may be beneficial to limit the maximal degree of g to be 2 or 3 depending on the size of d. Spurious higher order induced oscillations may lead to incorrect gradient estimates at chosen values of $\boldsymbol{x}$. When d is very large, this strategy may be computationally prohibitive as it requires approximating the function in the full-space $\mathbb{R}^d$ with the polynomial p.

B. Variable projection

To mitigate this, we consider the algorithm of Hokanson and Constantine [45] that yields the optimal polynomial over a subspace, provided the subspace dimension n is known. The authors use variable projection [46] to express the problem of optimising jointly over both the subspace (Grassmann manifold) and the polynomial coefficients

$$r = \underset{\boldsymbol{U},\boldsymbol{\alpha}}{\text{minimise}} \ \|f(\boldsymbol{x}) - p_{\boldsymbol{\alpha}}\|_2^2, \tag{26}$$

with $p_{\boldsymbol{\alpha}} = p\left(\boldsymbol{U}^T\boldsymbol{x}\right)$, where the subscript $\boldsymbol{\alpha}$ denotes the unknown coefficients. Assuming one has access to input-output data pairs $\{\boldsymbol{\xi}_i, f_i\}_{i=1}^M$, and setting

$$\boldsymbol{A}\left(\left\{\boldsymbol{U}^T\boldsymbol{\xi}_i\right\}_{i=1}^M\right) = \begin{bmatrix} \boldsymbol{\phi}_1\left(\boldsymbol{U}^T\boldsymbol{\xi}_1\right)\sqrt{\omega_1} & \dots & \boldsymbol{\phi}_N\left(\boldsymbol{U}^T\boldsymbol{\xi}_1\right)\sqrt{\omega_1} \\ \vdots & \ddots & \vdots \\ \boldsymbol{\phi}_1\left(\boldsymbol{U}^T\boldsymbol{\xi}_M\right)\sqrt{\omega_M} & \dots & \boldsymbol{\phi}_N\left(\boldsymbol{U}^T\boldsymbol{\xi}_M\right)\sqrt{\omega_M} \end{bmatrix} \tag{27}$$

one can express (26) as

$$\begin{aligned} r &= \underset{\boldsymbol{U},\boldsymbol{\alpha}}{\text{minimise}} \ \left\|\mathbf{b} - \boldsymbol{A}\left(\left\{\boldsymbol{U}^T\boldsymbol{\xi}_i\right\}_{i=1}^M\right)\boldsymbol{\alpha}\right\|_2^2 \\ &= \underset{\boldsymbol{U}}{\text{minimise}} \ \left\|\mathbf{b} - \boldsymbol{A}\left(\left\{\boldsymbol{U}^T\boldsymbol{\xi}_i\right\}_{i=1}^M\right)\boldsymbol{A}^\dagger\left(\left\{\boldsymbol{U}^T\boldsymbol{\xi}_i\right\}_{i=1}^M\right)\mathbf{b}\right\|_2^2 \\ &= \underset{\boldsymbol{U}}{\text{minimise}} \ \left\|\left(\boldsymbol{I} - \boldsymbol{A}\left(\left\{\boldsymbol{U}^T\boldsymbol{\xi}_i\right\}_{i=1}^M\right)\boldsymbol{A}^\dagger\left(\left\{\boldsymbol{U}^T\boldsymbol{\xi}_i\right\}_{i=1}^M\right)\right)\mathbf{b}\right\|_2^2, \end{aligned} \tag{28}$$

where the superscript † denotes the pseudoinverse. The optimisation problem in (28) can be solved via a Gauss-Newton optimisation [45], following which the coefficients $\boldsymbol{\alpha}$ can be trivially computed. This effectively yields the subspace polynomial $p\left(\mathbf{U}^T x\right)$ which can be used as a surrogate.

C. Vector-valued dimension reduction

It is natural to extend the idea of a dimension reducing subspace to vector-valued quantities. Zahm et al. [47] formulate this as a vector-valued generalisation of active subspaces. Wong et al. [48] argue that as most scalar-valued qois are spatio-temporal integrals of vector-valued quantities (scalar fields), there is merit—both computational and inferential—in identifying dimension reducing subspaces for the composite vector-valued quantities. Their approach and some more recent work [44, 49] seeks to also construct a relationship between different elements of the vector-valued quantity. For instance, in [48], a recursive nearest neighbour algorithm is used to identify elements that have similar subspaces; in [49] the intersection of the subspaces of different quantities is used, while in [44] a separate covariance function is used to establish an empirical relationship between different polynomial subspace-based approximations.

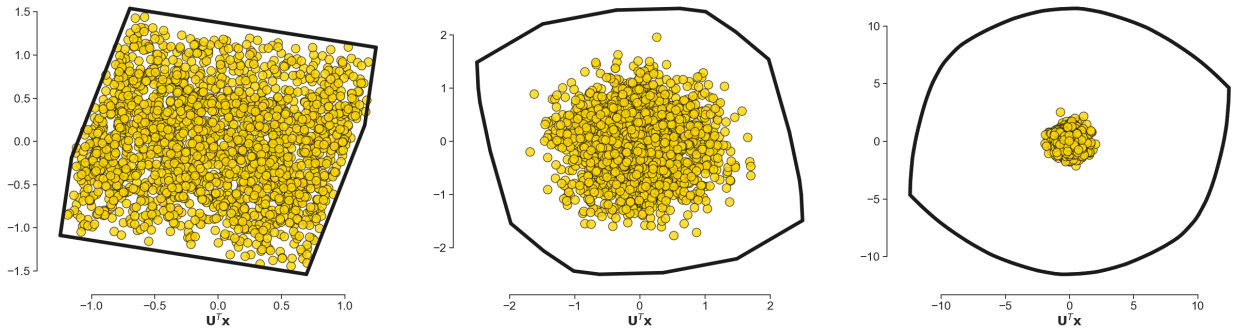
Generally, these works approximate a vector-valued function $\mathbf{h}(\boldsymbol{x}) \in \mathbb{R}^T$ as

$$\mathbf{h}(\boldsymbol{x}) = \begin{bmatrix} h_1(\boldsymbol{x}) \\ \vdots \\ h_T(\boldsymbol{x}) \end{bmatrix} \approx \begin{bmatrix} p_1\left(\boldsymbol{U}_1^T\boldsymbol{x}\right) \\ \vdots \\ p_T\left(\boldsymbol{U}_T^T\boldsymbol{x}\right) \end{bmatrix} \tag{29}$$

where $\{p_1, \dots, p_T\}$ represent the T different subspace-based polynomial approximations with subspaces $\{\boldsymbol{U}_1, \dots, \boldsymbol{U}_T\}$, each possibly of different dimension. Such approximations can be effectively computed in equadratures and used in a variety of applications.

In Scillitoe et al. [44], these ideas are used to develop a full flowfield estimation framework, which is found to demonstrate competitive accuracy to a state-of-the-art convolutional neural network. The dimension-reducing nature of the ridge approximations embedded within the flowfield can provide physical insight, and aids design space exploration.

The utility of multi-objective dimension reduction does not end with approximating flow fields. In [49, 50], multi-objective dimension reduction is used as the computational backbone for *blade envelopes*, which offer rigorous quantification of the effect of geometric deviations on manufactured and in-service turbomachinery blades. By using dimension reducing subspaces associated with multiple objectives, geometric variations that can cause performance deterioration can be differentiated from those that do not, taking into account multiple performance indices including surface flow features.



(a) (b) (c)

Fig. 3 Zonotopes in $\mathbb{R}^n$ with $n = 2$ associated uniform samples in (a) $d = 3$; (b) $d = 10$, and (c) $d = 200$.

D. Zonotopes and synthetic data generation

Consider the set of all projected samples given by $\mathcal{Z} = \left\{z = \mathbf{U}^T\boldsymbol{x}, \;\; \boldsymbol{x} \in \mathcal{X}\right\}$. In the case $\mathcal{X}$ is bounded with a closed interval along each direction, $\mathcal{Z}$ is a convex polytope residing in $\mathbb{R}^n$. One can think of $\mathcal{Z}$ as the silhouette of the d-dimensional hypercube, formally termed a zonotope. Figure 3 plots zonotopes for uniformly distributed random points in $\mathbb{R}^d$ with $d = 3, 10$ and 200. The apparent collapsing of these points is a consequence of the Johnson-Lindenstrauss lemma [51], which states that pairwise distances between points are roughly preserved without significant distortion.

Algorithms to compute the vertices of these zonotopes may be found in [52, 53]; in equadratures we utilise the former. This permits one to visualise the data projected over $\mathbb{R}^n$. Additionally, note that one can use convex optimisation strategies to generate synthetic data. We discuss this point in further detail in the examples section of this paper.

V. Statistics

The `Statistics` class in equadratures computes moments, Sobol' indices and higher-order statistical metrics from the calculated coefficients. This builds upon some core ideas arising from polynomial chaos in the context of uncertainty quantification.

A. Moments and sensitivities

Computation of the mean and variance are readily straightforward when using pseudospectral approximations [5]. We can write the mean as

$$\begin{aligned}\mathbb{E}\left(f\left(\boldsymbol{x}\right)\right) &\approx \mathbb{E}\left(p\left(\boldsymbol{x}\right)\right) \\ &= \mathbb{E}\left[\sum_{i=1}^{n} \alpha_i \boldsymbol{\phi}_i\left(\boldsymbol{x}\right)\right] \\ &= \mathbb{E}\left[\alpha_1 \boldsymbol{\phi}_1\left(\boldsymbol{x}\right)\right] + \underbrace{\mathbb{E}\left[\sum_{i=2}^{n} \alpha_i \boldsymbol{\phi}_i\left(\boldsymbol{x}\right)\right]}_{=0} \\ &= \alpha_1.\end{aligned} \tag{30}$$

In other words, the mean is simply given by the first coefficient of the expansion. Estimating the variance is also straightforward

$$\begin{aligned}\sigma^2\left(f\left(\boldsymbol{x}\right)\right) &\approx \sigma^2\left(p\left(\boldsymbol{x}\right)\right)\\ &= \mathbb{E}\left[\left(\sum_{i=1}^{n}\alpha_i\boldsymbol{\phi}_i\left(\boldsymbol{x}\right)-\alpha_1\right)^2\right]\\ &= \mathbb{E}\left[\left(\sum_{i=2}^{n}\alpha_i\boldsymbol{\phi}_i\left(\boldsymbol{x}\right)\right)^2\right]\\ &= \sum_{i=2}^{n}\alpha_i^2.\end{aligned} \tag{31}$$

It is this ability to rapidly estimate statistical moments in the absence of additional sampling that makes these polynomial approximations so useful.

B. Global sensitivity analysis

Engineers are often interested in the answer to the question, "which of my model parameters are the most important?" This is the one of the objectives of global sensitivity analysis. It seeks to apportion the uncertainty f according to the contribution of the inputs $\boldsymbol{x}$. Typically, in this context importance is characterized by the conditional variance. One relatively well-known strategy to quantify the importance of various inputs is through Sobol' indices (see page 323 in [5]), which can be readily approximated using orthogonal polynomial expansions [6]. As before let $\mathcal{I}$ be the multi-index set associated with a chosen polynomial basis used for approximating a function f. From (31), we can write the variance as

$$\sigma^2 = \sum_{\boldsymbol{i}\in\mathcal{I},\,\boldsymbol{i}\neq\mathbf{1}} x_{\boldsymbol{i}}^2. \tag{32}$$

Now Sobol' indices represent a fraction of the total variance that is attributed to each input variable (the first order Sobol' indices) or combinations thereof (higher order Sobol' indices). Let $\mathcal{I}_{\boldsymbol{s}}$ be the set of multi-indices that depend only on the subset of variables $\boldsymbol{s} = \{i_1,\ldots,i_s\}$, i.e.,

$$\mathcal{I}_{\boldsymbol{s}} = \left\{\boldsymbol{i}\in\mathbb{N}^d \;:\; l\in\boldsymbol{s}\Leftrightarrow i_l\neq 0\right\}. \tag{33}$$

The first order partial variances σ_i^2 are then obtained by summing up the square of the coefficients in $\mathcal{I}_{\boldsymbol{s}}$

$$\sigma_j^2 = \sum_{\boldsymbol{i}\in\mathcal{I}_j} x_{\boldsymbol{i}}^2, \quad \mathcal{I}_j = \left\{\boldsymbol{i}\in\mathbb{N}^d : i_j > 0\right\}, \tag{34}$$

and the higher order variances $\sigma^2_{\{i_1,\ldots,i_s\}}$ can be written as

$$\sigma_{\boldsymbol{s}}^2 = \sum_{\boldsymbol{i}\in\mathcal{I}_{\boldsymbol{s}}} x_{\boldsymbol{i}}^2, \quad \mathcal{I}_{\{i_1,\ldots,i_s\}} = \left\{\boldsymbol{i}\in\mathbb{N}^d : l\in\boldsymbol{s}\Leftrightarrow i_l > 0\right\}. \tag{35}$$

The first and higher order Sobol' indices are then given by

$$S_j = \frac{\sigma_j^2}{\sigma^2} \;\text{ and }\; S_{\boldsymbol{s}} = \frac{\sigma_{\boldsymbol{s}}^2}{\sigma^2}. \tag{36}$$

C. Higher order statistics

In some cases, statistics beyond the output mean and variance can provide useful information about the behavior of a model. Statistics such as the global skewness and kurtosis quantify the weight of the tail of a distribution, which can lead to better predictions about the occurrence of extreme events. For example, [54] finds that incorporating information about output skewness and kurtosis improves the accuracy of the output probability distribution function of an approximate model.

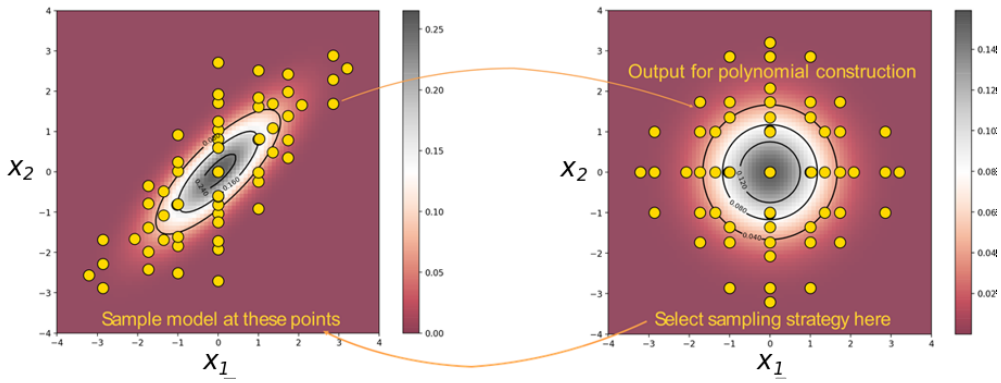


Fig. 4 Illustrating the use of the Nataf transform to sample input points for polynomial approximations.

In a similar vein to Sobol' indices, sensitivity indices can be constructed for skewness [55] and kurtosis [56]. In particular, the work of Owen et al. [57] hypothesizes that skewness-based indices are useful for indicating the sensitivities of different input parameters near regions where the output is driven to extrema. Given that engineering systems are often operated at points where output quantities are optimized, it is useful to know about the input factors that can affect the operation of these systems most strongly. In Wong et al. [58], the connections between skewness-based indices and *extremum Sobol' indices*—a set of variance-based indices adapted to output extrema via Monte Carlo Filtering and polynomial approximations—are examined in the context of calculating extremum sensitivities.

D. On correlations

Recall from Section II.A that the construction of multivariate orthogonal polynomials as products of univariate polynomials relies on the fact that the input distribution ρ can be decomposed into the product of its marginals. That is, it was assumed that input parameters are distributed with pairwise independence. Supposing that this assumption cannot be made, how can one construct orthogonal polynomial bases?

One approach is to construct a bijective map T from the space of correlated input variables $\mathbf{x}$ to another where the transformed inputs $\mathbf{z}$ are no longer correlated, and are distributed with normal distributions,

$$\mathbf{z} = T(\mathbf{x}). \tag{37}$$

With $\mathbf{z}$, one can then construct an orthogonal polynomial system with the methods described so far. The resultant polynomial can then be evaluated by the inverse transform T^{-1}

$$f(\mathbf{x}) = f(T^{-1}(\mathbf{z})) \approx \sum_{i=1}^{n} \alpha_i \boldsymbol{\phi}(\mathbf{z}) \tag{38}$$

This is known as the *Nataf transform* [59] (see Figure 4). The limitation of this method is that one now has to fit a polynomial to the function $f \circ T^{-1}$ instead of f, where the transform and its inverse can introduce strong non-linearity, increasing the order of polynomials required to adequately form an approximation.

Another approach [60] uses the *Gram-Schmidt process* to construct an orthogonal basis from an initially linear independent basis set. Ignoring the correlations between the inputs $\mathbf{x}$, one can construct a basis set $\{\boldsymbol{\psi}(\mathbf{x})\}$ that is orthogonal with respect to the product of the marginals of ρ, but not orthogonal with respect to $\boldsymbol{\rho}(\boldsymbol{x})$ (since correlations are ignored). Since they are still linearly independent, the Gram-Schmidt process can be applied on the basis functions. A discrete analogue of the process that can be implemented on a computer involves the QR decomposition

$$\frac{1}{\sqrt{M}} \boldsymbol{\Psi} = \boldsymbol{Q}\boldsymbol{R}, \tag{39}$$

where $\boldsymbol{\Psi}$ is a polynomial evaluation matrix for M points sampled randomly according to the *correlated* distribution ρ. A new set of polynomials $\{\boldsymbol{\phi}\}$ can be constructed by

$$\phi_j(\mathbf{x}) = \sum_{k=1}^{N} \psi_k(\mathbf{x})(\boldsymbol{R}^{-1})_{kj}. \tag{40}$$

The resultant set $\{\boldsymbol{\phi}\}$ is then orthogonal to $\boldsymbol{\rho}$.

With the added complexity of modelling correlations and constructing new polynomial bases, one may ask whether it is important to model correlations in the first place. In some simulation and experimental situations, we may have *a priori* reasons to believe that correlations are present in the inputs. In this case, resultant output statistics derived from the polynomial approximation can be significantly affected if such correlations are neglected [61].

VI. Classification

Equadratures is not solely limited to regression problems. Many classification paradigms can generate a binary response to whether a given set of observations fall in a class or not by wrapping an exponential function around a linear regression model. In what follows, we detail how this is done with equadratures—leveraging some of the subspace-based ideas presented above.

In classification tasks, we are given a set of training data, in the form of input/output data pairs

$$\mathcal{D} = \{(\boldsymbol{\xi}_1, c_1), (\boldsymbol{\xi}_2, c_2), \ldots, (\boldsymbol{\xi}_M, c_M)\}, \tag{41}$$

where the outputs c_i take discrete values from a finite set, corresponding to K different classes. The objective is to learn a function—the classifier—which allows the prediction of the class of an unseen data point. The simplest way to evaluate the performance of a classifier is to check its classification accuracy on a set of unseen test data: the fraction of correctly classified test data points. Classification, especially of various types of structured data, has been an active area of research, and many types of models such as support vector machines, convolutional neural networks and decision trees have been proposed and studied.

A. Standard logistic regression

We begin by considering the standard logistic classifier (also known as the perceptron), which takes the form

$$g(\boldsymbol{x}; \mathbf{w}) = \frac{1}{1 + \exp(-\mathbf{w}^T \mathbf{x})}. \tag{42}$$

The function takes in a linear transformation of the input $-\mathbf{w}^T\boldsymbol{x}$, and returns a value between 0 and 1. When applied to binary classification, where c_i can take two values—0 and 1—the output can be interpreted as a probability representing the confidence that the point belongs to class 1. To train the logistic classifier, one can optimize the weights $\mathbf{w}$ with respect to a loss function that characterize the misfit of the training data by the classifier. One popular loss function is the cross entropy, defined as

$$L(\mathbf{w}; \mathcal{D}) = -\frac{1}{M} \sum_{i=1}^{M} c_i \log g(\boldsymbol{\xi}_i; \mathbf{w}) + (1 - c_i) \log(1 - g(\boldsymbol{\xi}_i; \mathbf{w})). \tag{43}$$

The derivative of this function with respect to the parameters $\mathbf{w}$ is calculated as

$$\frac{\partial L}{\partial \mathbf{w}} = -\frac{1}{M} \sum_{i=1}^{M} \boldsymbol{\xi}_i (c_i - g(\boldsymbol{\xi}_i; \mathbf{w})) \tag{44}$$

Using this expression, the loss function can be minimized using a gradient-based method, such as conjugate gradients. The logistic classifier depends on the input in the form of $\mathbf{w}^T\boldsymbol{x}$, a linear function of $\boldsymbol{x}$. The set of points that yield a value of 0.5 when put through the logistic classifier forms the decision boundary, separating the input domain into two partitions. In one partition, we predict the point to be in class 0, and the other, class 1. For the logistic classifier, this boundary is a hyperplane. However, in most classification tasks, the data is not separable by a hyperplane.

B. Polynomial subspace-based logistic regression

Here, we describe an extension to this well-known formulation by allowing the decision surface to be defined by a polynomial ridge approximation. The classifier is now of the form

$$\begin{aligned} g(\boldsymbol{x}) &= \frac{1}{1 + \exp(-p(\boldsymbol{U}^T\boldsymbol{x}))} \\ &= \frac{1}{1 + \exp(-\sum_{j=1}^{N} \alpha_j \boldsymbol{\phi}_j (\boldsymbol{U}^T\boldsymbol{x}))}. \end{aligned} \tag{45}$$

where the polynomial expansion takes in a linear transformation of the input $\mathbf{U}^T\boldsymbol{x}$, where as before $\mathbf{U} \in \mathbb{R}^{d\times n}$ with $n \ll d$. This effectively constraints the decision function to the subspace spanned by the columns of $\mathbf{U}$. In equadratures, we formulate this as a manifold optimisation problem

$$\begin{aligned} \underset{\mathbf{U},\boldsymbol{\alpha}}{\text{minimize}} \quad & L(\boldsymbol{U},\boldsymbol{\alpha};\mathcal{D}) = -\frac{1}{M}\sum_{i=1}^{M} c_i \log f(\boldsymbol{\xi}_i;\boldsymbol{U},\boldsymbol{\alpha}) + (1-c_i)\log(1-f(\boldsymbol{\xi}_i;\boldsymbol{U},\boldsymbol{\alpha})). \\ \text{subject to} \quad & \boldsymbol{U} \in \text{St}(d,n). \end{aligned} \tag{46}$$

A few remarks on the above are in order, particularly in light of our exposition of the variable projection approach for solving the regression problem (see (26)). First, our choice in selecting the Stiefel manifold is based on the desire to have $\boldsymbol{U}$ be orthonormal, i.e., $\boldsymbol{U}^T\boldsymbol{U} = \boldsymbol{I}$. Second, unlike the regression formulation, the above classification one is not readily separable and therefore an alternating approach is used. As the optimisation over $\boldsymbol{U}$ is constrained to the Stiefel manifold, it needs to be performed separately from that of $\boldsymbol{\alpha}$. This necessitates the approach described below.

- Fix $\boldsymbol{U}$, optimise over c using a standard gradient-based method.
- Fix $\boldsymbol{\alpha}$, and optimise over $\boldsymbol{U}$ over the Stiefel manifold using a gradient-based method.

The two steps are repeated until convergence in the objective. To perform the latter step, tools from differential geometry can be used to formulate the gradient on the Stiefel manifold (see [62]).

VII. Polynomial Trees

There are numerous instances when fitting a global polynomial to data is not appropriate. Figure 5(a,b) highlights some of the issues, including the gross over prediction at the boundaries and the smoothing out of discontinuities in the data. Both of these can be misleading, particularly in high-dimensional problems where the polynomial approximation may not be easily visualised. This motivates the utilisation of tree-based hierarchies to delineate where to best split a polynomial.

A. Gradient-based splitting

The majority of tree-based learning models are constructed using recursive partitioning, yielding an efficient yet essentially locally optimal approach. Notable exceptions include tree induction methods using genetic algorithms [63], and those utilising mathematical programming [64]. Well-known CART decision trees [65] recursively partition $\mathcal{X}$, by greedily performing binary left/right splits splits at each node. The *split-dimension* and *split-threshold* are chosen to maximise the *gain* in predictive accuracy. In an attempt to achieve predictive accuracy, whilst keeping training costs low, [66] propose a new splitting criterion based upon gradients of the weak models' loss. We adopt their criterion in equadratures to build regression trees.

Our objective is to identify a split along a given dimension $x^{(k)} \in \mathcal{X}^{(k)}$ for some $k \leq d$. We assume the existence of a dataset of input-output pairs $\{\boldsymbol{\xi}_i, f_i\}_{i=1}^{M}$ where we denote the complete set of input samples as $D = \{\boldsymbol{\xi}_1, \ldots, \boldsymbol{\xi}_M\}$. As we seek to partition this data into two child nodes, we define $\mathcal{L} \subseteq \mathcal{D}$ as the set of data falling in the left child node, and $\mathcal{R} = \mathcal{D} \setminus \mathcal{L}$ as being its complement and therefore the set of samples falling in the right child node. Within the two child nodes, the standard L_2 norm loss function l is used to determine the quality of the approximation. At the parent node, a *gain* function $g(\mathcal{L},\mathcal{R})$ that aggregates these two loss functions needs to be computed. To minimise computational costs, [66] propose a Taylor series based approximation to this gain function yielding

$$g(\mathcal{L},\mathcal{R}) = \|\nabla_{\boldsymbol{\alpha}} l(\mathcal{L})\|_2^2 + \|\nabla_{\boldsymbol{\alpha}} l(\mathcal{R})\|_2^2 \tag{47}$$

Setting the loss function to be the standard weighted L_2 norm error

$$l_{\boldsymbol{\alpha}}(\mathcal{D}) = \|\boldsymbol{A}\boldsymbol{\alpha} - \mathbf{b}\|_2^2, \tag{48}$$

the gradients in (47) can be trivially computed. In equadratures this functionality is embedded within the `Polytree` class, which provides the user with flexibility on the (i) maximum order of the polynomial to be utilised; (ii) the regression method used for computing $\boldsymbol{\alpha}$, and (iii) the ability to prune down the number of child nodes based on the loss function value. A simple demonstration of this is shown in Figure 5(c, d).

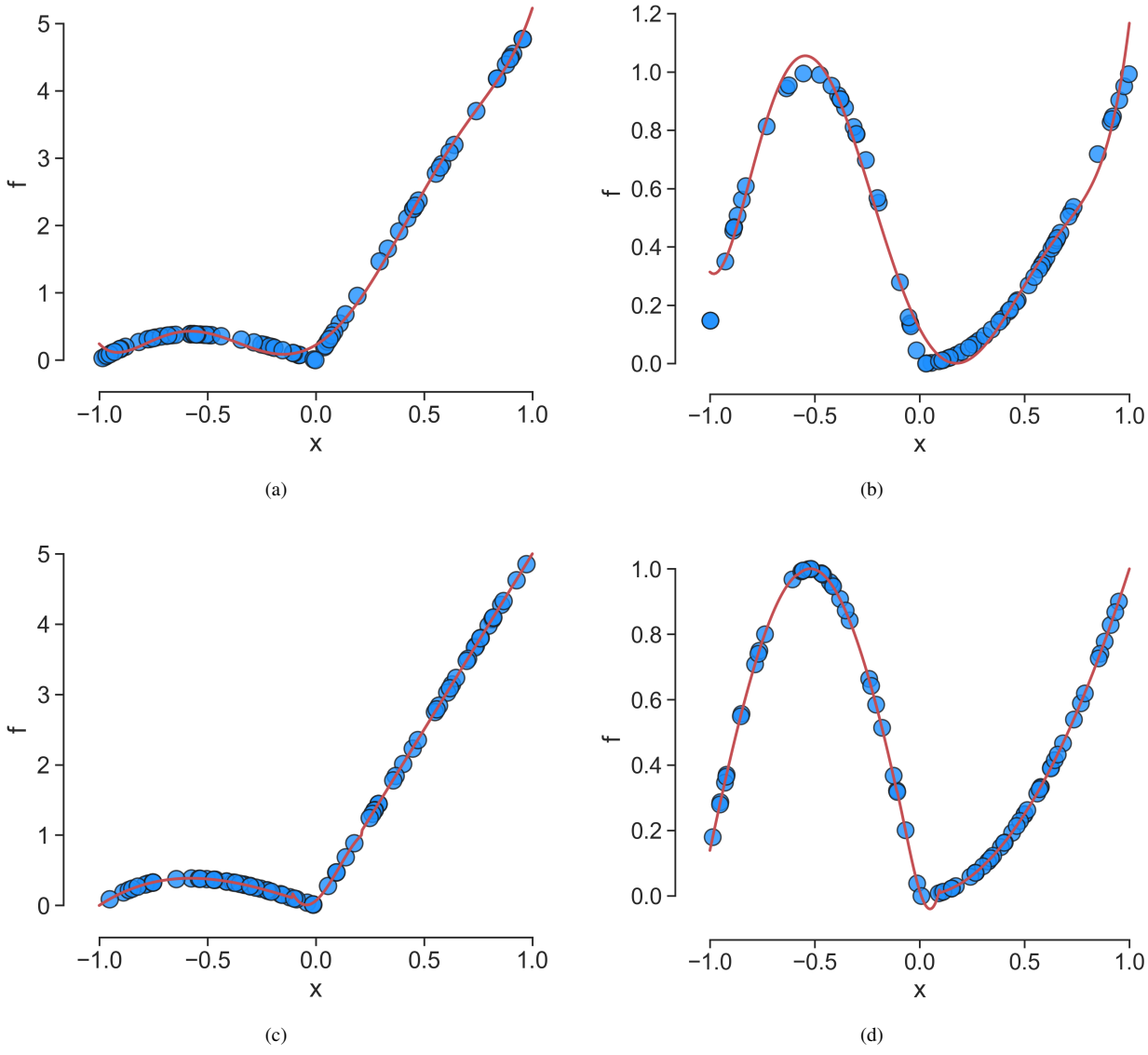


Fig. 5 Fitting a global polynomial to two different datasets in (a, b); fitting a piecewise polynomial tree-based model to the same datasets in (c, d).

B. Computing moments on trees

As established in Section V, a salient advantage to using orthogonal polynomials is the ease of computations for output moments. With polynomial trees, moments can also be readily furnished through minor modifications that appropriately sum up contributions from the different node polynomials.

With a polynomial tree, the input domain is partitioned into regions, and within each an orthogonal polynomial is fit. The task is to quantify global moments given the coefficients of the polynomial in each sub-domain. To ease exposition, we will make three simplifying assumptions:

1) the input distribution is uniform over a bounded hypercube;
2) in each subdomain, we also have uniform distributions, and
3) legendre orthogonal polynomials are used in each subdomain.

This implies that in subdomain j, the distribution is

$$\rho^{(j)}(\mathbf{x}) = \frac{1}{Vol^{(j)}} \tag{49}$$

where $Vol^{(j)}$ is the volume of the hypercube defining the subdomain. Also, note that

$$\frac{\rho(\mathbf{x})}{\rho^{(j)}(\mathbf{x})} = \frac{Vol^{(j)}}{Vol} \tag{50}$$

where Vol is the overall input domain's volume. The mean can thus be computed as:

$$\begin{aligned}
\mathbb{E}(f(\mathbf{x})) &= \int p(\mathbf{x})\,\rho(\mathbf{x})\,d\mathbf{x} \\
&= \sum_j \int_j p^{(j)}(\mathbf{x})\,\rho(\mathbf{x})\,d\mathbf{x} \\
&= \sum_j \frac{Vol^{(j)}}{Vol} \int_j p^{(j)}(\mathbf{x})\,\rho^{(j)}(\mathbf{x})\,d\mathbf{x} \\
&= \sum_j \frac{Vol^{(j)}}{Vol} \alpha_1^{(j)}.
\end{aligned}$$

where $\int_j$ is an integral over subdomain j, while $p^{(j)}$ and $\alpha^{(j)}$ are the polynomial fitted in this subdomain and its coefficients respectively. The variance can also be computed:

$$\begin{aligned}
\sigma^2(f(\mathbf{x})) &= \int p(\mathbf{x})^2\,\rho(\mathbf{x})\,d\mathbf{x} - \left(\int p(\mathbf{x})\,\rho(\mathbf{x})\,d\mathbf{x}\right)^2 \\
&= \sum_j \frac{Vol^{(j)}}{Vol} \int_j p^{(j)}(\mathbf{x})^2\,\rho^{(j)}(\mathbf{x})\,d\mathbf{x} - \left(\sum_j \frac{Vol^{(j)}}{Vol} \int_j p^{(j)}(\mathbf{x})\,\rho^{(j)}(\mathbf{x})\,d\mathbf{x}\right)^2 \\
&= \sum_j \frac{Vol^{(j)}}{Vol} \sum_{i=1}^{P} \alpha_i^{(j)2} - \left(\sum_j \frac{Vol^{(j)}}{Vol} \alpha_1^{(j)}\right)^2.
\end{aligned}$$

For distributions other than the uniform distribution, the ratio of densities $\rho(\mathbf{x})/\rho^{(j)}(\mathbf{x})$ takes a different form, but the moments can be evaluated in a similar manner.

VIII. Case studies

In this final section, we share four illustrations of a subset of the methodologies presented earlier, in the form of case studies. All are focused on building surrogate models to Computational Fluid Dynmaics (CFD) based datasets.

A. Uncertainty quantification with missing data

Consider a simulation of the von Karman Institute LS89 turbine cascade [67], shown in Figure 6, simulated with the SU2 CFD code [68]. This is a popular test case for exploring the performance of different CFD approaches for

turbomachinery applications [69]. As is common in industry, we choose to use a RANS model, the Shear Stress Transport (SST) model [70], to account for turbulence. This requires two additional transport equations to be solved, one for the turbulent kinetic energy k, and one for the specific dissipation rate ω. The inflow and outflow boundary conditions are well specified in the experiment. However, the introduction of the SST model necessitates the specification of two additional inflow boundary conditions, the turbulence intensity Ti, and the turbulent viscosity ratio ν_t/ν*. These quantities are not measured in the experiment, and so they introduce some additional uncertainty. Our aim here is to quantify the uncertainty in our output qoi, due the uncertainty in these turbulent boundary conditions. We take the stagnation pressure loss coefficient as our qoi

$$Y_p = \frac{p_{0_\infty} - p_0}{p_{0_\infty} - p_\infty}. \tag{51}$$

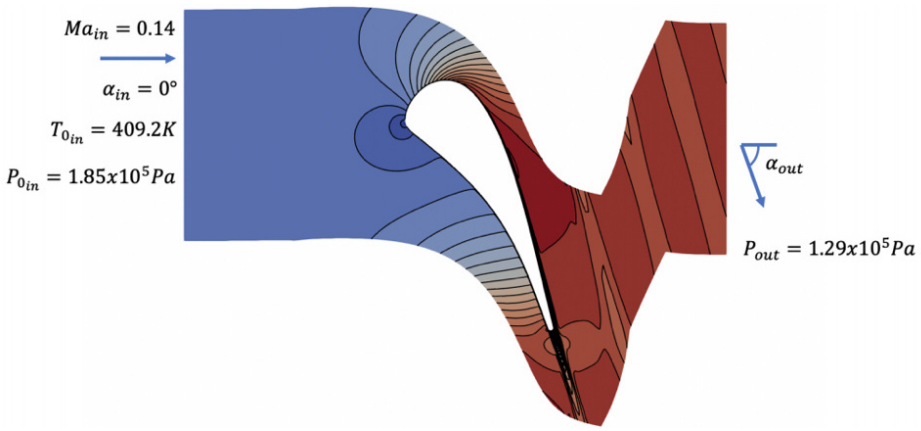


Fig. 6 A RANS simulation of the LS89 turbine cascade, with contours coloured by Mach number.

We assume that we have a rough estimate for the value of Ti, since this quantity is commonly measured using a hot-wire probe(s). For this reason, we define it as a normal distribution $\mathcal{N}(10, 5)$, indicating an estimate of $Ti \approx 10\%$, albeit with a considerable uncertainty of $\sigma^2 = 5\%$. On the other hand, ν_t/ν is a physically vague quantity, and values ranging from 1 to 100 are often recommended. Therefore, we define it as a uniform distribution $\mathcal{U}(1, 100)$. The next task is to define a polynomial basis, for which we select a standard tensorial basis.

The resulting design of experiments (doe) consists of an $N = 16$ set of $\mathbf{x}$ vectors containing the Ti and ν_t/ν values at which the CFD model must be evaluated, as shown in Figure 7(a). The CFD model is evaluated 16 times to obtain the loss coefficient values at each doe point, and the Y_p values are stored. It is then trivial to compute the mean and variance; and in this case we find that the 95% confidence interval in Y_p due to the uncertainty in inflow turbulence specification is $\pm 4.7 \times 10^{-4} p.p.$. This may appear small, but it equates to 0.88% of the mean value, so may be significant for some applications.

Now, in practice, CFD simulations may fail or have poor convergence, and not all 16 Y_p values may be available, as shown in Figure 7(b), where four simulations have been randomly assigned `NaN` values, and fed into the code. The code recognises this, and alters the polynomial coefficient computing strategy, opting to use least squares on the data that is available. The results are summarised in Table 1, where it is clear that despite the missing data, the code is able to offer reasonable predictions.

Table 1 Estimated variance of turbine stage efficiency with different methods.

Case	Mean	Confidence interval %
full tensor grid	0.0532	0.881
missing entries	0.0532	0.873

For completeness, the code used to generate these results is given below.

*The SU2 solver uses turbulence intensity Ti, and turbulent viscosity ratio ν_t/ν as its turbulent inflow boundary conditions. The original turbulent field variables can be recovered with $k = \frac{3}{2}(U\,Ti)$ and $\omega = \frac{k}{\nu}\left(\frac{\nu_t}{\nu}\right)^{-1}$, where ν is the laminar kinematic viscosity.

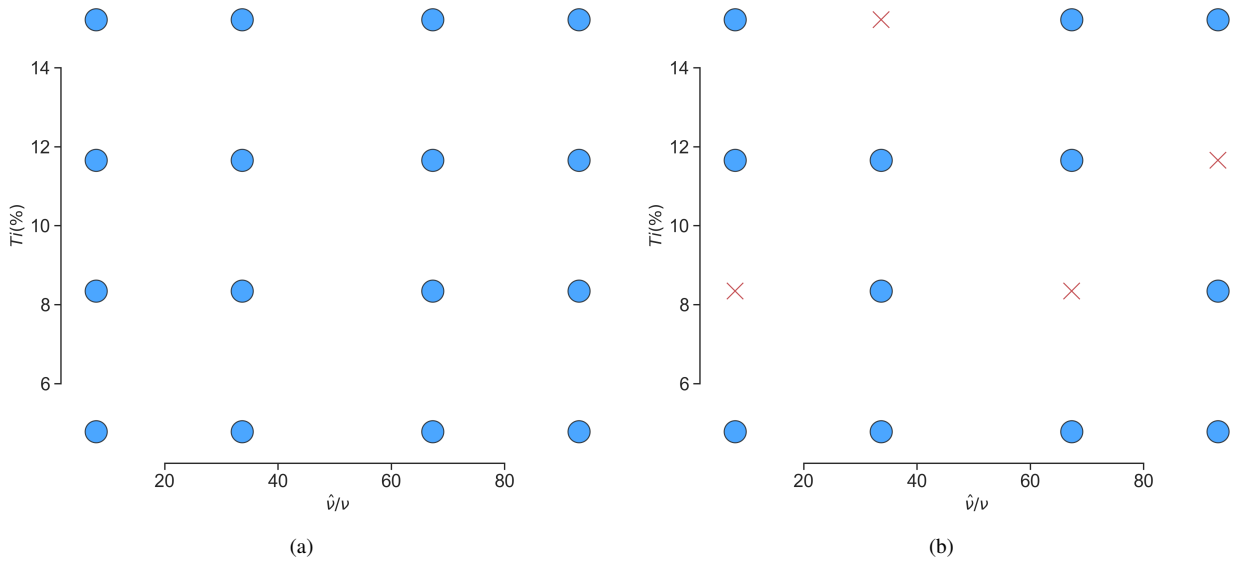


Fig. 7 A tensor grid set of points for the VKI turbine blade design of experiment: (a) full grid; (b) grid with missing evaluations.

```python
from equadratures import *

# define parameters and basis
turb_visc = Parameter(distribution='Uniform', lower=1.0, upper=100, order=3)
Ti = Parameter(distribution='Gaussian', shape_parameter_A=10,
            shape_parameter_B=5, order=3)
mybasis = Basis('tensor-grid')

# Define polynomial and get quadrature points
mypoly = Poly(parameters=[turb_visc,Ti],
            basis=mybasis,
            method='numerical-integration')
pts = mypoly.get_points() # evaluate CFD at these points

Yp = np.load('CFD_results.npy') # load CFD based Yp
mypoly.set_model(Yp) # compute polynomial coefficients
mean, var = mypoly.get_mean_and_variance() # get mean and variance
CI = 1.96*math.sqrt(var)/mean
```

Listing 1 Code for VKI LS89 turbine blade uncertainty quantification study.

B. ONERA M6 with polynomial trees

We consider transonic flow over the 3D ONERA M6 wing. Using the inviscid compressible solver in the SU2 CFD suite [68], we simulate this for a range of freestream conditions, as shown in Figure 8. Due to its transonic nature, the flowfield undergoes a pronounced change as the freestream Mach number is increased. This results in a near step-change in the quantity of interest, the drag coefficient C_D, making it difficult to approximate with a global polynomial.

To mitigate this issue, we invoke equadrature's `PolyTree` class to partition the input parameter space with a decision tree, and then fit polynomials to the leaf nodes' data. Originally intended as a highly interpretable model for supervised machine learning tasks, they can also be used for uncertainty quantification applications where a continuous polynomial approximation is inaccurate. Syntax for the `PolyTree` class closely resembles that of other popular machine learning python libraries like *scikit-learn* [71]. The resulting tree, and its response surface, are visualised in Figure 9. Splitting the input domain into two sub-domains has allowed for more appropriate handling of the discontinuity around $Ma_\infty = 0.92$.

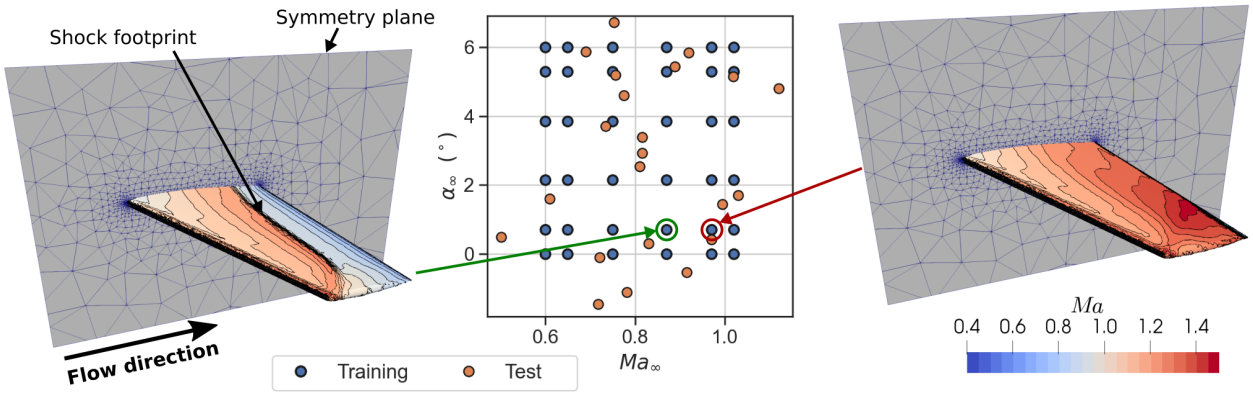


Fig. 8 The DOE for the ONERA M6 transonic wing test case, with freestream angle of incidence α_∞ and freestream Mach number Ma_∞ varied. Contours of surface Mach number are shown for two of the DOE points.

The drag coefficient is seen to be a function of α_∞ only for $Ma_\infty > 0.92$, whilst elsewhere, the drag coefficient is influenced by Ma_∞, and the interaction between Ma_∞ and α_∞.

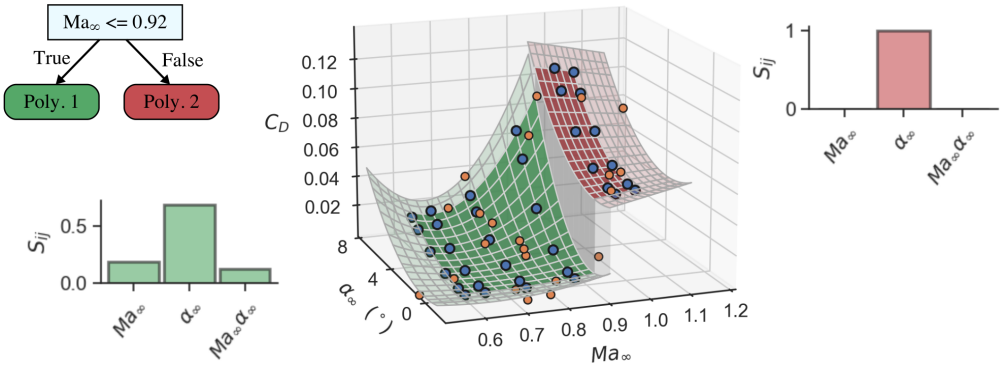


Fig. 9 A one split deep polynomial regression tree, fitted to the training data from the ONERA M6 transonic wing DOE. First and second order Sobol' indices are shown for the left and right polynomials.

To demonstrate the ease with which the results above can be generated, we include the code below. Note that for computing the moments and sensitivities both local (specific to the node, i.e., one polynomial) and global (across the entire tree) metrics can be readily obtained following VII.B. Additionally, note that a graphical representation of the tree can be obtained as shown in Figure 9.

```
from equadratures import *

mytree = PolyTree(splitting_criterion='loss-gradient', max_depth=1, order=2) # define
mytree.fit(X_train, y_train) # fit to training data
test_prediction = mytree.predict(X_test) # make predictions on test data
left_poly = mytree.get_node(1)
left_poly.plot_sobol_indices()
mytree.get_mean_and_variance() # returns mean and variance for the entire tree
mytree.get_graphviz() # plot the tree with graphviz
```

Listing 2 Code for constructing a piecewise polynomial regression tree.

C. Subspace-based polynomial approximation

In this case study we consider the CFD-generated datasets found in [72]. The data set comprises the efficiency and pressure ratio for a fan blade parameterised by 25 design variables. The objective of this study is to understand

how subspace-based projections may be used to analyse the geometrical parameters that drive efficiency and pressure ratio values, in the same manner as [18] but in a few lines of equadratures code using the variable projection approach detailed in IV.B.

```
from equadratures import *

# load the data
data = datasets.load_eq_dataset('3Dfan_blades',verbose=False)
X  = data['X_a']
y_efficiency = data['y1_a']
y_pr = data['y1_b']

# efficiency subspace
subspace_efficiency = Subspaces(method='variable-projection',
                        sample_points=X,
                        sample_outputs=y_efficiency)
plot_2D_contour_zonotope(subspace_efficiency)

# pressure ratio subspace
subspace_pr = Subspaces(method='variable-projection',
                        sample_points=X,
                        sample_outputs=y_pr)
plot_2D_contour_zonotope(subspace_pr)
```

Listing 3 Code for generating polynomial subspace-based approximations.

By default equadratures assumes the subspace dimension $n = 2$. The results of the code above are captured in Figure 10. Note that the subspaces used for the projections in (a) and (b) are different as they correspond to the two different objectives. However, utilities based on a hit-and-run method [52] are used in equadratures to generate samples in one subspace and project them to the other. The results of generating samples in the pressure ratio subspace and projecting them to the efficiency subspace are shown in Figure 11. Note that we include both a mean and a standard deviation. This is because infinitely many samples in $\mathbb{R}^d$ that map to $\mathbb{R}^n$ can be generated. Thus, when quantifying the pressure ratio variation across the efficiency subspace, in equadratures, 500 random samples in $\mathbb{R}^d$ are generated for each $\mathbb{R}^n$ coordinate on a grid of approximately 10,000 points on the efficiency subspace in Figure 10(a). Each of these 5,000,000 $\mathbb{R}^d$ samples is then pushed through the polynomial subspace-based approximation for the pressure ratio. The mean and standard deviation of the 500 samples are reported for each coordinate in Figure 11(a) and (b) respectively. These results illustrate that in regions where the efficiency is high, there is on average a tendency for the pressure ratio values to be low. The code for the last two plots are given below for completeness.

```
plot_samples_from_second_subspace_over_first(mysubspace_1=subspace_efficiency, \
                                             mysubspace_2=subspace_pr)
```

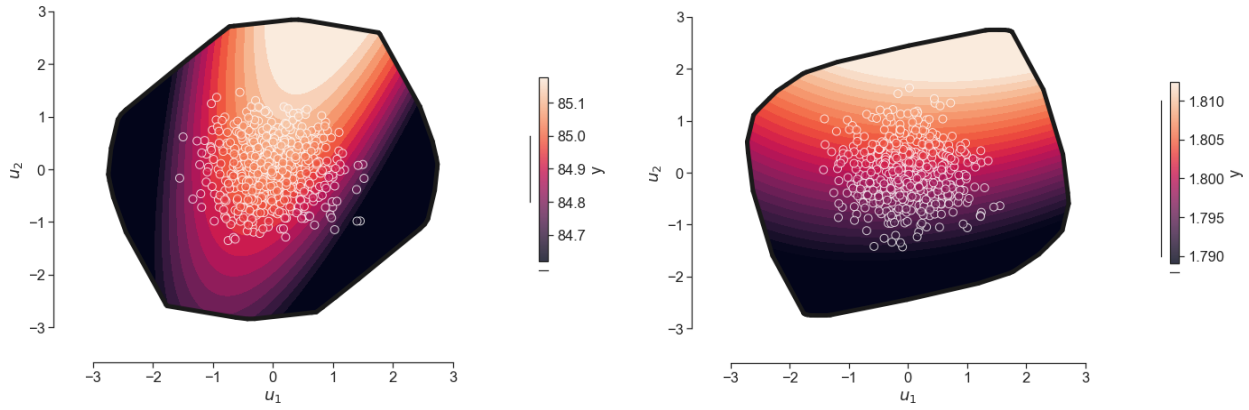
Listing 4 Code for generating polynomial subspace-based approximations.

IX. Conclusions

This paper presented an overview of equadratures along with a deep-dive into certain aspects of the code. What started off as a project for facilitating reproducible research in polynomial chaos has now transformed into a much larger project touching multiple aspects of computational engineering, data science, and machine learning. Through the numerical examples in this paper, we demonstrated the easy approach to build polynomials—be it global, piecewise, gradient-enhanced, sparse, or subspace-based—to compute moments and sensitivities for both regression and classification tasks.

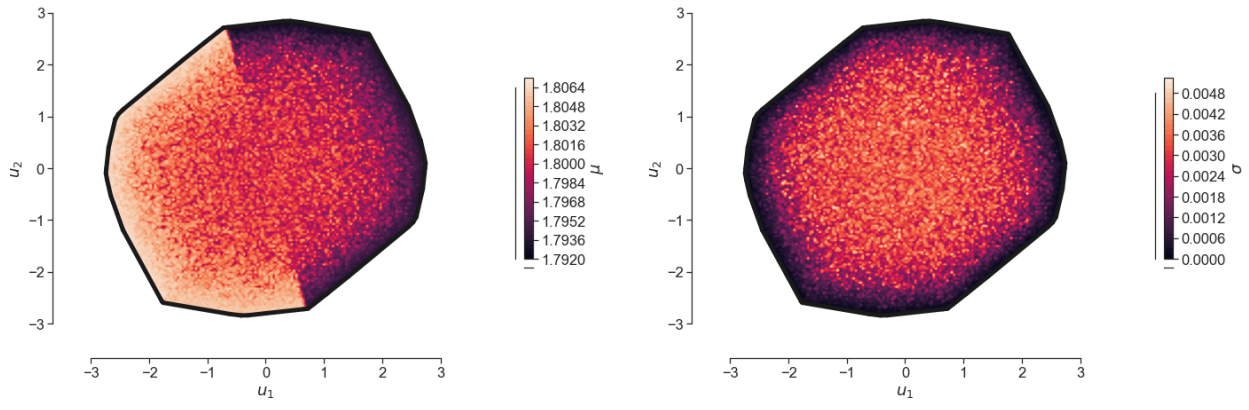
Some of the novel methodology contributions of our work include:

- The utilisation of relevance vector machine for estimating coefficients for polynomial approximations, building upon ideas from the Bayesian machine learning community.
- The development of a polynomial tree framework with the ability to estimate global statistics, using the coefficients and basis polynomials defined at each node. The tree also has flexibility in the way the coefficients are computed.
- The development of a polynomial subspace-based classification paradigm that pairs manifold optimisation techniques with well-worn cross entropy objective for logistic regression.



(a) (b)

Fig. 10 Plots of the (a) efficiency, and (b) pressure ratio subspace, obtained via polynomial variable projection. The polynomial contours are shown over the zonotopes; the circular markers (and colors therein) denote the data.



(a) (b)

Fig. 11 Plots of the pressure ratio (a) mean, and (b) standard deviation on the efficiency subspace.

Additionally, the code has been purpose-built to be easily accessible, ensuring engineers and data scientists can prototype code that can be quickly deployed on relevant industrial problems.

Acknowledgments

Our code equadratures has been supported by numerous grants. It was founded through an Engineering and Physical Sciences Research Council (EPSRC) Knowledge Transfer Fellowship (KTF) from the Department of Engineering at the University of Cambridge in 2016. Since 2018, the project has been supported by The UKRI Strategic Priorities Fund under the EPSRC Grant EP/T001569/1, particularly the "Digital twins for complex systems engineering" theme within that grant, and by the Lloyd's Register Foundation-Alan Turing Institute programme on Data-Centric Engineering under the LRF grant G0095. The project is also grateful to the financial support of Jesus College Cambridge and the Cambridge Trusts.

References

[1] Seshadri, P., and Parks, G., "Effective-quadratures (EQ): Polynomials for computational engineering studies," *The Journal of Open Source Software*, Vol. 2, 2017, pp. 166–166.

[2] Xiu, D., and Karniadakis, G. E., "The Wiener–Askey polynomial chaos for stochastic differential equations," *SIAM journal on scientific computing*, Vol. 24, No. 2, 2002, pp. 619–644.

[3] Constantine, P. G., Eldred, M. S., and Phipps, E. T., "Sparse pseudospectral approximation method," *Computer Methods in Applied Mechanics and Engineering*, Vol. 229, 2012, pp. 1–12.

[4] Seshadri, P., Narayan, A., and Mahadevan, S., "Effectively subsampled quadratures for least squares polynomial approximations," *SIAM/ASA Journal on Uncertainty Quantification*, Vol. 5, No. 1, 2017, pp. 1003–1023.

[5] Smith, R. C., *Uncertainty quantification: theory, implementation, and applications*, Vol. 12, Siam, 2013.

[6] Sudret, B., "Global sensitivity analysis using polynomial chaos expansions," *Reliability engineering & system safety*, Vol. 93, No. 7, 2008, pp. 964–979.

[7] Cohen, A., Davenport, M. A., and Leviatan, D., "On the stability and accuracy of least squares approximations," *Foundations of computational mathematics*, Vol. 13, No. 5, 2013, pp. 819–834.

[8] Lüthen, N., Marelli, S., and Sudret, B., "Sparse polynomial chaos expansions: Literature survey and benchmark," *SIAM/ASA Journal on Uncertainty Quantification*, Vol. 9, No. 2, 2021, pp. 593–649.

[9] Guo, L., Narayan, A., and Zhou, T., "Constructing least-squares polynomial approximations," *SIAM Review*, Vol. 62, No. 2, 2020, pp. 483–508.

[10] Hadigol, M., and Doostan, A., "Least squares polynomial chaos expansion: A review of sampling strategies," *Computer Methods in Applied Mechanics and Engineering*, Vol. 332, 2018, pp. 382–407.

[11] Peng, J., Hampton, J., and Doostan, A., "A weighted 1-minimization approach for sparse polynomial chaos expansions," *Journal of Computational Physics*, Vol. 267, 2014, pp. 92–111.

[12] Björck, Å., *Numerical methods in matrix computations*, Vol. 59, Springer, 2015.

[13] Seshadri, P., Iaccarino, G., and Ghisu, T., "Quadrature Strategies for Constructing Polynomial Approximations," *Uncertainty Modeling for Engineering Applications*, Springer, 2019, pp. 1–25.

[14] Samarov, A. M., "Exploring regression structure using nonparametric functional estimation," *Journal of the American Statistical Association*, Vol. 88, No. 423, 1993, pp. 836–847.

[15] Constantine, P. G., *Active subspaces: Emerging ideas for dimension reduction in parameter studies*, SIAM, 2015.

[16] Cook, R. D., and Ni, L., "Sufficient dimension reduction via inverse regression: A minimum discrepancy approach," *Journal of the American Statistical Association*, Vol. 100, No. 470, 2005, pp. 410–428.

[17] Pinkus, A., *Ridge functions*, Vol. 205, Cambridge University Press, 2015.

[18] Seshadri, P., Shahpar, S., Constantine, P., Parks, G., and Adams, M., "Turbomachinery active subspace performance maps," *Journal of Turbomachinery*, Vol. 140, No. 4, 2018, p. 041003.

[19] Scillitoe, A. D., Ubald, B., Seshadri, P., and Shahpar, S., "Design Space Exploration of Stagnation Temperature Probes via Dimension Reduction," *Turbo Expo: Power for Land, Sea, and Air*, Vol. 84089, American Society of Mechanical Engineers, 2020, p. V02CT35A057.

[20] Constantine, P. G., and Diaz, P., "Global sensitivity metrics from active subspaces," *Reliability Engineering & System Safety*, Vol. 162, 2017, pp. 1–13.

[21] Constantine, P. G., Emory, M., Larsson, J., and Iaccarino, G., "Exploiting active subspaces to quantify uncertainty in the numerical simulation of the HyShot II scramjet," *Journal of Computational Physics*, Vol. 302, 2015, pp. 1–20.

[22] Jakeman, J. D., Franzelin, F., Narayan, A., Eldred, M., and Plfüger, D., "Polynomial chaos expansions for dependent random variables," *Computer Methods in Applied Mechanics and Engineering*, Vol. 351, 2019, pp. 643–666.

[23] Novak, L., and Novak, D., "Polynomial chaos expansion for surrogate modelling: Theory and software," *Beton-und Stahlbetonbau*, Vol. 113, 2018, pp. 27–32.

[24] Nocedal, J., and Wright, S. J., "Trust-region methods," *Numerical Optimization*, 2006, pp. 66–100.

[25] Breiman, L., Friedman, J. H., Olshen, R. A., and Stone, C. J., *Classification and regression trees*, Routledge, 2017.

[26] Gross, J. C., and Parks, G. T., "Optimization by moving ridge functions: derivative-free optimization for computationally intensive functions," *Engineering Optimization*, 2021, pp. 1–23.

[27] Trefethen, L. N., "Cubature, approximation, and isotropy in the hypercube," *SIAM Review*, Vol. 59, No. 3, 2017, pp. 469–491.

[28] Blatman, G., and Sudret, B., "Adaptive sparse polynomial chaos expansion based on least angle regression," *Journal of computational Physics*, Vol. 230, No. 6, 2011, pp. 2345–2367.

[29] Gautschi, W., "On generating orthogonal polynomials," *SIAM Journal on Scientific and Statistical Computing*, Vol. 3, No. 3, 1982, pp. 289–317.

[30] Narayan, A., Jakeman, J., and Zhou, T., "A Christoffel function weighted least squares algorithm for collocation approximations," *Mathematics of Computation*, Vol. 86, No. 306, 2017, pp. 1913–1947.

[31] Narayan, A., "Computation of induced orthogonal polynomial distributions," *arXiv preprint arXiv:1704.08465*, 2017.

[32] Ryu, E. K., and Boyd, S. P., "Extensions of Gauss quadrature via linear programming," *Foundations of Computational Mathematics*, Vol. 15, No. 4, 2015, pp. 953–971.

[33] Jakeman, J. D., and Narayan, A., "Generation and application of multivariate polynomial quadrature rules," *Computer Methods in Applied Mechanics and Engineering*, Vol. 338, 2018, pp. 134–161.

[34] Chen, S. S., Donoho, D. L., and Saunders, M. A., "Atomic decomposition by basis pursuit," *SIAM review*, Vol. 43, No. 1, 2001, pp. 129–159.

[35] Hampton, J., and Doostan, A., "Compressive sampling methods for sparse polynomial chaos expansions," *Handbook of uncertainty quantification*, 2016, pp. 1–29.

[36] Nocedal, J., and Wright, S., *Numerical optimization*, Springer Science & Business Media, 2006.

[37] Candès, E. J., Romberg, J., and Tao, T., "Robust uncertainty principles: Exact signal reconstruction from highly incomplete frequency information," *IEEE Transactions on information theory*, Vol. 52, No. 2, 2006, pp. 489–509.

[38] Alemazkoor, N., and Meidani, H., "A near-optimal sampling strategy for sparse recovery of polynomial chaos expansions," *Journal of Computational Physics*, Vol. 371, 2018, pp. 137–151.

[39] Tang, G., and Iaccarino, G., "Subsampled Gauss quadrature nodes for estimating polynomial chaos expansions," *SIAM/ASA Journal on Uncertainty Quantification*, Vol. 2, No. 1, 2014, pp. 423–443.

[40] Tipping, M. E., "The relevance vector machine," *Advances in neural information processing systems*, 2000, pp. 652–658.

[41] Ji, S., Xue, Y., and Carin, L., "Bayesian compressive sensing," *IEEE Transactions on signal processing*, Vol. 56, No. 6, 2008, pp. 2346–2356.

[42] Ghisu, T., Lopez, D. I., Seshadri, P., and Shahpar, S., "Gradient-enhanced Least-square Polynomial Chaos Expansions for Uncertainty Quantification and Robust Optimization," *AIAA AVIATION 2021 FORUM*, 2021, p. 3073.

[43] Constantine, P. G., Eftekhari, A., Hokanson, J., and Ward, R. A., "A near-stationary subspace for ridge approximation," *Computer Methods in Applied Mechanics and Engineering*, Vol. 326, 2017, pp. 402–421.

[44] Scillitoe, A., Seshadri, P., Wong, C. Y., and Duncan, A., "Polynomial ridge flowfield estimation," *Physics of Fluids*, Vol. 33, 2021.

[45] Hokanson, J. M., and Constantine, P. G., "Data-driven polynomial ridge approximation using variable projection," *SIAM Journal on Scientific Computing*, Vol. 40, No. 3, 2018, pp. A1566–A1589.

[46] Golub, G., and Pereyra, V., "Separable nonlinear least squares: the variable projection method and its applications," *Inverse problems*, Vol. 19, No. 2, 2003, p. R1.

[47] Zahm, O., Constantine, P. G., Prieur, C., and Marzouk, Y. M., "Gradient-based dimension reduction of multivariate vector-valued functions," *SIAM Journal on Scientific Computing*, Vol. 42, No. 1, 2020, pp. A534–A558.

[48] Wong, C. Y., Seshadri, P., Parks, G. T., and Girolami, M., "Embedded ridge approximations," *Computer Methods in Applied Mechanics and Engineering*, Vol. 372, 2020, p. 113383.

[49] Wong, C. Y., Seshadri, P., Scillitoe, A., Ubald, B. N., Duncan, A. B., and Parks, G., "Blade Envelopes Part II: Multiple Objectives and Inverse Design," *ASME Journal of Turbomachinery (in press)*, 2021.

[50] Wong, C. Y., Seshadri, P., Scillitoe, A., Duncan, A. B., and Parks, G., "Blade Envelopes Part I: Concept and Methodology," *ASME Journal of Turbomachinery (in press)*, 2021.

[51] Dasgupta, S., and Gupta, A., "An elementary proof of the Johnson-Lindenstrauss lemma," *International Computer Science Institute, Technical Report*, Vol. 22, No. 1, 1999, pp. 1–5.

[52] Constantine, P., Howard, R., Glaws, A., Grey, Z., Diaz, P., and Fletcher, L., "Python active-subspaces utility library," *Journal of Open Source Software*, Vol. 1, No. 5, 2016, p. 79.

[53] Fukuda, K., "From the zonotope construction to the Minkowski addition of convex polytopes," *Journal of Symbolic Computation*, Vol. 38, No. 4, 2004, pp. 1261–1272.

[54] Geraci, G., Congedo, P. M., Abgrall, R., and Iaccarino, G., "High-order statistics in global sensitivity analysis: Decomposition and model reduction," *Computer Methods in Applied Mechanics and Engineering*, Vol. 301, 2016, pp. 80–115.

[55] Wang, S. C., "Quantifying passive and driven large-scale evolutionary trends," *Evolution*, Vol. 55, No. 5, 2001, pp. 849–858.

[56] Dell'Oca, A., Riva, M., and Guadagnini, A., "Moment-based metrics for global sensitivity analysis of hydrological systems," *Hydrology and Earth System Sciences*, Vol. 21, 2017, pp. 6219–6234.

[57] Owen, A. B., Dick, J., and Chen, S., "Higher order Sobol' indices," *Information and Inference: A Journal of the IMA*, Vol. 3, No. 1, 2014, pp. 59–81.

[58] Wong, C. Y., Seshadri, P., and Parks, G., "Extremum sensitivity analysis with polynomial Monte Carlo filtering," *Reliability Engineering System Safety*, Vol. 212, 2021, p. 107609.

[59] Liu, P.-L., and Der Kiureghian, A., "Multivariate distribution models with prescribed marginals and covariances," *Probabilistic Engineering Mechanics*, Vol. 1, No. 2, 1986, pp. 105–112.

[60] Jakeman, J. D., Eldred, M. S., Geraci, G., and Gorodetsky, A., "Adaptive multi-index collocation for uncertainty quantification and sensitivity analysis," *International Journal for Numerical Methods in Engineering*, Vol. 121, No. 6, 2020, pp. 1314–1343. https://doi.org/https://doi.org/10.1002/nme.6268.

[61] Seshadri, P., Duncan, A., Simpson, D., Thorne, G., and Parks, G., "Spatial Flow-Field Approximation Using Few Thermodynamic Measurements—Part II: Uncertainty Assessments," *Journal of Turbomachinery*, Vol. 142, No. 2, 2020, p. 021007.

[62] Edelman, A., Arias, T. A., and Smith, S. T., "The geometry of algorithms with orthogonality constraints," *SIAM journal on Matrix Analysis and Applications*, Vol. 20, No. 2, 1998, pp. 303–353.

[63] Barros, R. C., Basgalupp, M. P., de Carvalho, A. C. P. L. F., and Freitas, A. A., "A Survey of Evolutionary Algorithms for Decision-Tree Induction," *IEEE Transactions on Systems, Man, and Cybernetics, Part C (Applications and Reviews)*, Vol. 42, No. 3, 2012, pp. 291–312. https://doi.org/10.1109/tsmcc.2011.2157494, URL https://doi.org/10.1109/tsmcc.2011.2157494.

[64] Yang, L., Liu, S., Tsoka, S., and Papageorgiou, L. G., "A regression tree approach using mathematical programming," *Expert Systems with Applications*, Vol. 78, 2017, pp. 347–357. https://doi.org/10.1016/j.eswa.2017.02.013, URL https://doi.org/10.1016/j.eswa.2017.02.013.

[65] Breiman, L., Friedman, J., Stone, C. J., and Olshen, R. A., *Classification and Regression Trees*, 1st ed., Chapman & Hall, 1984.

[66] Broelemann, K., and Kasneci, G., "A gradient-based split criterion for highly accurate and transparent model trees," *Int. Jt. Conf. Artif. Intell.*, 2019, pp. 2030–2037. https://doi.org/10.24963/ijcai.2019/281.

[67] Arts, T., and de Rouvroit, M. L., "Aero-Thermal Performance of a Two-Dimensional Highly Loaded Transonic Turbine Nozzle Guide Vane: A Test Case for Inviscid and Viscous Flow Computations," *Journal of Turbomachinery*, Vol. 114, No. 1, 1992, pp. 147–154. https://doi.org/10.1115/1.2927978, URL https://doi.org/10.1115/1.2927978.

[68] Economon, T. D., Palacios, F., Copeland, S. R., Lukaczyk, T. W., and Alonso, J. J., "SU2: An Open-Source Suite for Multiphysics Simulation and Design," *AIAA Journal*, Vol. 54, No. 3, 2016, pp. 828–846. https://doi.org/10.2514/1.j053813, URL https://doi.org/10.2514/1.j053813.

[69] Segui, L. M., Gicquel, L., Duchaine, F., and de Laborderie, J., "LES of the LS89 cascade: influence of inflow turbulence on the flow predictions," *12th European Conference on Turbomachinery Fluid Dynamics and hermodynamics*, European Turbomachinery Society, 2017. https://doi.org/10.29008/etc2017-159, URL https://doi.org/10.29008/etc2017-159.

[70] Menter, F. R., "Two-equation eddy-viscosity turbulence models for engineering applications," *AIAA Journal*, Vol. 32, No. 8, 1994, pp. 1598–1605. https://doi.org/10.2514/3.12149, URL https://doi.org/10.2514/3.12149.

[71] Pedregosa, F., Varoquaux, G., Gramfort, A., Michel, V., Thirion, B., Grisel, O., Blondel, M., Prettenhofer, P., Weiss, R., Dubourg, V., Vanderplas, J., Passos, A., Cournapeau, D., Brucher, M., Perrot, M., and Duchesnay, E., "Scikit-learn: Machine Learning in Python," *Journal of Machine Learning Research*, Vol. 12, 2011, pp. 2825–2830.

[72] Seshadri, P., Yuchi, S., Parks, G., and Shahpar, S., "Supporting multi-point fan design with dimension reduction," *The Aeronautical Journal*, Vol. 124, No. 1279, 2020, pp. 1371–1398.