ORNL/TM-2015/596

User Manual: Toolkit for Adaptive Stochastic Modeling and Non-Intrusive Approximation (TASMANIAN)

Miroslav Stoyanov

September 2019, version 7.0

OAK RIDGE NATIONAL LABORATORY
MANAGED BY UT-BATTELLE FOR THE US DEPARTMENT OF ENERGY

ORNL/TM-2015/596

Computer Science and Mathematics Division

USER MANUAL: TOOLKIT FOR ADAPTIVE STOCHASTIC MODELING AND NON-INTRUSIVE APPROXIMATION (TASMANIAN)

Miroslav Stoyanov

Date Published: September 2017

Prepared by
OAK RIDGE NATIONAL LABORATORY
Oak Ridge, TN 37831-6283
managed by
UT-Battelle, LLC
for the
US DEPARTMENT OF ENERGY
under contract DE-AC05-00OR22725

CONTENTS

LIST OF FIGURES

LIST OF TABLES

ABSTRACT

This documents serves to explain the general mathematics of the Toolkit for Adaptive Stochastic Modeling And Non-Intrusive Approximation (TASMANIAN) and to provide further references to the methods and approaches. Detailed documentation of the C++ code and other interfaces is included in the on-line Doxygen documentation.

Doxygen documentation for the latest stable release can be found at:
https://tasmanian.ornl.gov/documentation/

The documentation of the rolling release can be found at:
https://ornl.github.io/TASMANIAN/

1 Quick Overview

1.1 Sparse grids

Sparse Grids refers to a family of algorithms for approximation of multidimensional functions and integrals, where the approximation operator is constructed as a linear combination of tensors of multiple one dimensional operators[41, 19, 35, 36, 31, 22, 30, 1, 3, 21, 13, 4, 18, 44, 46, 47, 43, 51, 52, 17, 7, 23, 25, 26, 27, 29, 33, 34, 38, 39, 45, 40, 12, 15].

The Tasmanian sparse grids library implements a wide variety of sparse grids methods with different one dimensional operators and different ways of constructing the linear combination of tensors.

Let $\Gamma^{a_k,b_k} = [a_k, b_k] \subset \mathbb{R}$, for $k = 1, 2, \ldots, d$, indicate a set of one dimensional intervals and let $\Gamma^{\boldsymbol{a},\boldsymbol{b}} = \bigotimes_{k=1}^{d} \Gamma^{a_k,b_k} \subset \mathbb{R}^d$ be a d-dimensional sparse grids domain. A sparse grid consists of a set of points $\{\boldsymbol{x}_i\}_{i=1}^{N} \in \Gamma^{\boldsymbol{a},\boldsymbol{b}}$ and associated numerical quadrature weights $\{\boldsymbol{x}_i\}_{i=1}^{N} \in \mathbb{R}$ or interpolation basis functions $\{\phi_i(\boldsymbol{x})\}_{i=1}^{N} \in \mathbb{C}^0(\Gamma^{\boldsymbol{a},\boldsymbol{b}})$. Usually, a_k and b_k are finite, however, Gauss-Hermite and Gauss-Laguerre rules allow for the use of unbounded domain. Note that Tasmanian constructs grids using the canonical interval $[-1, 1]$ and the result is then translated (via a linear transformation) to the specific $[a_k, b_k]$; also Gauss-Hermite and Gauss-Laguerre rules use canonical intervals $(-\infty, +\infty)$ and $[0, \infty)$ respectively.

Let $f(\boldsymbol{x}) : \Gamma \to \mathbb{R}$ indicate a d-dimensional function, where w.l.o.g. we assume Γ is the canonical domain. We consider two types of approximations, point-wise approximations $\tilde{f}(\boldsymbol{x})$ where $\tilde{f}(\boldsymbol{x}) \approx f(\boldsymbol{x})$ for all $\boldsymbol{x} \in \Gamma$ and numerical integration $Q(f)$ where $Q(f) \approx \int_\Gamma f(\boldsymbol{x})\rho(\boldsymbol{x})d\boldsymbol{x}$. The weight $\rho(\boldsymbol{x})$ is specific to the one dimensional rule that induces the grid; most rules assume uniform weight $\rho(\boldsymbol{x}) = 1$, however, Gauss-Chebyshev, Gegenbauer, Jacobi, Hermite, and Laguerre, rules use different weights (see Table 2). Note: Tasmanian can handle functions with multiple outputs (e.g., vector valued functions), then $\tilde{f}(\boldsymbol{x})$ and $Q(f)$ have a corresponding number of outputs.

Point-wise approximations can be implemented in two different ways, since both ways result in identical $\tilde{f}(\boldsymbol{x})$ there is no official language to distinguish between the two method, hence we'll use the terms *internal* and *adjoint*. The internal form is

$$\tilde{f}(\boldsymbol{x}) = \sum_{i=1}^{N} c_i \phi_i(\boldsymbol{x}), \tag{1}$$

where $\phi_i(\boldsymbol{x})$ are basis functions determined by the one dimensional rule and the chosen set of tensors, and the weights c_i are computed from the values of $f(\boldsymbol{x}_i)$. The term *internal* refers to the fact that the software library needs direct access to the values $f(\boldsymbol{x}_i)$ in order to compute the coefficients c_i. In contrast, the *adjoint* form is given by

$$\tilde{f}(\boldsymbol{x}) = \sum_{i=1}^{N} \psi_i(\boldsymbol{x}) f(\boldsymbol{x}_i), \tag{2}$$

where $\psi_i(\boldsymbol{x})$ depend on the 1-D rule and tensors and can be computed independent from $f(\boldsymbol{x}_i)$. Using the *adjoint* approach, Tasmanian can approximate functions with arbitrary output and arbitrary data-structures, i.e., the library can generate the $\psi_i(\boldsymbol{x})$ weights and the sum can be computed by user written or third party code. Note that (1) and (2) result in point-wise identical approximation, however, in general, the adjoint approach is usually significantly more expensive (computationally). When $\phi_i(\boldsymbol{x})$ are Lagrange polynomials, then $c_i = f(\boldsymbol{x}_i)$ and $\psi_i(\boldsymbol{x}) = \phi_i(\boldsymbol{x})$ and both approximation methods are computationally equivalent.

In general, sparse grids approximations are not interpolatory, however, when the underlying one dimensional rule is *nested* (i.e., the nodes at level l are a subset of the nodes at level $l+1$), then $\tilde{f}(\boldsymbol{x}_i) = f(\boldsymbol{x}_i)$ at all grid points $\{\boldsymbol{x}_i\}_{i=1}^N$. The Gauss rules implemented in Tasmanian (except Gauss-Patterson) and the Chebyshev rule are non-nested, all other rules are nested. In general, nested grids have fewer points which leads to fewer evaluations of $f(\boldsymbol{x}_i)$ and nesting allows the employment of various refinement strategies. Tasmanian implements two types of refinement based on hierarchical surpluses[43] and anisotropic quasi-optimal polynomial spaces[46].

Employing numerical quadrature, the integral of $f(\boldsymbol{x})$ is approximated as

$$\int_\Gamma f(\boldsymbol{x})\rho(\boldsymbol{x})d\boldsymbol{x} \approx Q[f] = \sum_{i=1}^N w_i f(\boldsymbol{x}_i), \tag{3}$$

where the points $\{\boldsymbol{x}_i\}_{i=1}^N$ and the weights $\{w_i\}_{i=1}^N$ depend on the one dimensional rule and the selection of tensors. In general, $Q(f)$ can be constructed from $\tilde{f}(\boldsymbol{x})$ by integrating the approximation (i.e., $w_i = \int_\Gamma \psi_i(\boldsymbol{x})d\boldsymbol{x}$), however, Gauss rules allow for better accuracy by selecting the points $\boldsymbol{x}_i$ at the roots of polynomials that are orthogonal with respect to $\rho(\boldsymbol{x})$ (see table 2). Gauss-Patterson and Gauss-Legendre rules use the same uniform $\rho(\boldsymbol{x})$, however, Gauss-Patterson points have the additional constraint of being nested. In one dimension, Gauss-Legendre rule is more accurate than Gauss-Patterson, however, in a multidimensional setting the nested property of Gauss-Patterson leads to better accuracy per number of points. Unlike Gauss-Legendre, the Gauss-Patterson points and weights are very difficult to compute and this library provides only the first 9 levels as hard-coded constants.

Tasmanian implements a variety of different grids and those are grouped into 4 categories:

- **Global Grids**: $\tilde{f}(\boldsymbol{x})$ is constructed using Lagrange polynomials and the grids are suitable for approximating smooth and analytic functions. All Gauss integration rules fall in this category. See §2.1.
- **Sequence Grids**: for a class of rules (namely Leja, $\mathcal{R}$-Leja, $\mathcal{R}$-Leja-Shifted, min/max-Lebesgue and min-Delta, see Table 3) the sequence grids offer an alternative implementation based on Newton polynomials. Sequence grids can evaluate $\tilde{f}(\boldsymbol{x})$ (for a given $\boldsymbol{x}$) much faster, however, speed comes with higher storage overhead as well as higher computational cost for most other operations, especially loading the values and using ajoint interpolation. Note that the difference between global and sequence grids is only in implementation, otherwise a sequence and a global grid with the same rule and points would result in identical $\tilde{f}(\boldsymbol{x})$. See §2.1.
- **Local Polynomial Grids**: suitable for non-smooth functions with locally sharp behavior. Interpolation is based on hierarchical piece-wise polynomials with local support and varying order. See §2.7.
- **Wavelet Grids**: are similar to the local polynomials, however, using wavelet basis. Coupled with local refinement, often times wavelet grids provide the same accuracy with fewer abscissas. See §2.7.

1.2 DREAM

Starting with version 5.0, Tasmanian includes a module for random sampling based on the DiffeRential Evolution Adaptive Metropolis (DREAM) algorithm. Suppose $\rho(\boldsymbol{x})$ be a non-negative function defined over a domain $\Gamma \subset \mathbb{R}^d$ where we make no assumptions regarding compactness or structure of Γ. We assume

that

$$\int_{\Gamma} \rho(\boldsymbol{x}) d\boldsymbol{x} < \infty,$$

in which case normalizing $\rho(\boldsymbol{x})$ will give us a probability density function (PDF). It is not necessary to explicitly compute the normalizing constant and the random sampling algorithm works with unscaled $\rho(\boldsymbol{x})$. The goal of the random sampling procedure is to generate a number of samples $\{\boldsymbol{x}_i\}_{i=1}^{N}$ that are distributed according to the $\rho(\boldsymbol{x})$ PDF. This is done by iteratively evolving a series of chains and the update algorithm depends on the current distribution of the chains and a random correction factor[6, 16, 49, 50, 53].

Bayesian inference is a common application area for this type of sampling algorithms [6]. In the inference paradigm, we have a model $f : \Gamma \to \mathbb{R}^{\mu}$ and (potentially noisy) observation data $d \in \mathbb{R}^{\mu}$, the objective is to assign "belief" to the values of $\boldsymbol{x}$ that correspond to the data. The "belief" is defined by a posterior probability distribution $\rho(\boldsymbol{x})$ defined by Bayes' rule

$$\rho(\boldsymbol{x}) = L(d, f(\boldsymbol{x}))\rho_p(\boldsymbol{x}), \tag{4}$$

where $L(d, f(\boldsymbol{x}))$ is the likelihood function indicating the probability that the discrepancy between d and $f(\boldsymbol{x})$ is due entirely to noise, and $\rho_p(\boldsymbol{x})$ is a probability distribution indicating our prior "belief" regarding the values of $\boldsymbol{x}$. The scaling factor in (4) is omitted. The statistics of $\rho(\boldsymbol{x})$ can be computed from a sufficiently large number of random samples collected by the DREAM algorithm

Accurate statistical analysis requires a huge number of random samples, which is prohibitive when the $f(\boldsymbol{x})$ is expensive to compute. A common practice is to replace the expensive $f(\boldsymbol{x})$ by a cheap to evaluate sparse grid surrogate. The Tasmanian DREAM module can collect samples from an arbitrary user defined $\rho(\boldsymbol{x})$ (not necessarily associated with an inference problem), a sparse grid approximation of the likelihood function or model. The user can provide a custom defined model as well. Currently, Tasmanian includes implementation of Gaussian likelihood, i.e.,

$$L(d, f(\boldsymbol{x})) = \exp\left(-(d - f(\boldsymbol{x}))^T \sigma^{-1} (d - f(\boldsymbol{x}))\right), \tag{5}$$

where $\sigma \in \mathbb{R}^{\mu \times \mu}$ is user defined covariance. Tasmanian also includes priors based on Uniform, Gaussian, truncated Gaussian, Beta, Gamma, and Exponential pdfs. The C++ library makes extensive use of polymorphism and can be easily extended with additional custom prior distributions, likelihood functions, and custom models.

2 Sparse Grids

2.1 Global Grids: General construction

Let $\{x_j\}_{j=1}^{\infty} \in \mathbb{R}$ denote a sequence of distinct points (in either a canonical or transformed interval $\Gamma^{a,b}$), and let $m : \mathbb{N} \to \mathbb{N}$ be a strictly increasing *growth function*. We define a one dimensional nested family of interpolants $\{\mathcal{U}^{m(l)}\}_{l=0}^{\infty}$, where $\mathcal{U}^{m(l)}$ is associated with the first $m(l)$ points $\{x_j\}_{j=1}^{m(l)}$ and Lagrange basis functions $\{\phi_j^l(x)\}_{j=1}^{m(l)}$ defined by $\phi_j^l(x) = \prod_{i=1, i\neq j}^{m(l)} \frac{x-x_i}{x_j-x_i}$, i.e.,

$$\tilde{f}^{(l)}(x) = \mathcal{U}^{m(l)}[f](x) = \sum_{j=1}^{m(l)} f(x_j)\phi_j^l(x). \tag{6}$$

The corresponding numerical quadrature is given by

$$\int f(x)\rho(x)dx \approx \mathcal{Q}[f] = \sum_{j=1}^{m(l)} w_j^l f(x_j), \tag{7}$$

where $w_j^l = \int \phi_j^l(x)\rho(x)dx$. In a non-nested case, different nodes are associated with each level, i.e., $\{\{x_j^l\}_{j=1}^{m(l)}\}_{l=0}^{\infty}$ and the basis function and operators are defined accordingly. Examples of nested and non-nested one dimensional rules are listed in Tables 1, 2, and 3.

The point-wise approximation and quadrature construction can be expressed in the same operator notation, hence, we define the surplus operators as

$$\Delta^{m(l)} = \mathcal{U}^{m(l)} - \mathcal{U}^{m(l-1)}, \qquad \text{or} \qquad \Delta^{m(l)} = \mathcal{Q}^{m(l)} - \mathcal{Q}^{m(l-1)} \tag{8}$$

depending on whether we are interested in constructing $\tilde{f}(\boldsymbol{x})$ or $\mathcal{Q}[f]$. We also use the convention that $\Delta^{m(0)} = \mathcal{U}^{m(0)}$ or $\Delta^{m(0)} = \mathcal{Q}^{m(0)}$.

The d-dimensional tenor operators are given by

$$\Delta^{\boldsymbol{m}(\boldsymbol{i})} = \bigotimes_{k=1}^{d} \Delta^{m(i_k)}, \qquad \mathcal{U}^{\boldsymbol{m}(\boldsymbol{i})} = \bigotimes_{k=1}^{d} \mathcal{U}^{m(i_k)}, \qquad \mathcal{Q}^{\boldsymbol{m}(\boldsymbol{i})} = \bigotimes_{k=1}^{d} \mathcal{Q}^{m(i_k)}$$

where we assume standard multi-index notation*. A sparse grid operator is defined as

$$G_{\Theta}[f] = \sum_{\boldsymbol{i}\in\Theta} \Delta^{\boldsymbol{m}(\boldsymbol{i})}, \tag{9}$$

where Θ is a lower set†. An explicit form of the points associated with the sparse grid can be obtained by first defining the tensors

$$\boldsymbol{m}(\boldsymbol{i}) = \bigotimes_{k=1}^{d} m(i_k), \qquad \boldsymbol{x_j} = \bigotimes_{k=1}^{d} x_{j_k},$$

*For the remainder of this document we let $\mathbb{N}$ be the set of natural numbers including zero, and $\Lambda, \Theta \subset \mathbb{N}^d$ will denote set of multi-indexes. For any two vectors, we define $\boldsymbol{x}^{\boldsymbol{\nu}} = \prod_{k=1}^{d} x_k^{\nu_k}$ with the usual convention $0^0 = 1$.

†A set Λ is caller lower or admissible if $\boldsymbol{\nu} \in \Lambda$ implies $\{\boldsymbol{i} \in \mathbb{N}^d : \boldsymbol{i} \leq \boldsymbol{\nu}\} \subset \Lambda$, where $\boldsymbol{i} \leq \boldsymbol{\nu}$ if and only if $i_k \leq \nu_k$ for all $1 \leq k \leq d$.

then the points associated with (9) are given by

$$\{x_{\boldsymbol{j}}\}_{\boldsymbol{j}\in X(\Theta)}, \qquad \text{where} \quad X(\Theta) = \bigcup_{\boldsymbol{i}\in\Theta} \{\mathbf{1} \le \boldsymbol{j} \le \boldsymbol{m}(\boldsymbol{i})\}. \tag{10}$$

In the non-nested case, $X(\Theta)$ consists of pairs of multi-indexes $X(\Theta) = \bigcup_{\boldsymbol{i}\in\Theta} \bigcup_{\mathbf{1}\le\boldsymbol{j}\le\boldsymbol{m}(\boldsymbol{i})} \{(\boldsymbol{i},\boldsymbol{j})\}$, and the points are $\{\boldsymbol{x}_{\boldsymbol{j}}^{\boldsymbol{i}}\}_{(\boldsymbol{i},\boldsymbol{j})\in X(\Theta)}$ where $\boldsymbol{x}_{\boldsymbol{j}}^{\boldsymbol{i}} = \bigotimes_{k=1}^{d} x_{j_k}^{i_k}$.

For every lower set Θ, there is a set of (integer) weights $\{t_{\boldsymbol{j}}\}_{\boldsymbol{j}\in\Theta(L)}$ that satisfy $\sum_{\boldsymbol{i}\le\boldsymbol{j},\boldsymbol{j}\in\Theta(L)} t_{\boldsymbol{j}} = 1$ for every $\boldsymbol{i} \in \Theta(L)$, i.e., $t_{\boldsymbol{i}}$ solve a linear system of equations. Then,

$$G_\Theta[f] = \sum_{\boldsymbol{i}\in\Theta} \Delta^{\boldsymbol{m}(\boldsymbol{i})} = \sum_{\boldsymbol{i}\in\Theta} t_{\boldsymbol{i}} \mathcal{U}^{\boldsymbol{m}(\boldsymbol{i})}, \tag{11}$$

or in the context of integration $G_\Theta[f] = \sum_{\boldsymbol{i}\in\Theta} t_{\boldsymbol{i}} \mathcal{Q}^{\boldsymbol{m}(\boldsymbol{i})}$. Thus, we explicitly write the Lagrange basis functions and quadrature weights as

$$\phi_{\boldsymbol{j}}(\boldsymbol{x}) = \sum_{\boldsymbol{i}\in\Theta,\boldsymbol{m}(\boldsymbol{i})\ge\boldsymbol{j}} t_{\boldsymbol{i}} \prod_{k=1}^{d} \phi_{j_k}^{i_k}, \tag{12}$$

where each $\phi_{j_k}^{i_k}$ is evaluated at the corresponding k-th component of $\boldsymbol{x}$ and we note that in the nested case $\phi_{\boldsymbol{j}}(\boldsymbol{x}) = \psi_{\boldsymbol{j}}(\boldsymbol{x})$ where $\psi_{\boldsymbol{j}}(\boldsymbol{x})$ are defined in (2). Similarly, the quadrature weights are given by

$$w_{\boldsymbol{j}} = \sum_{\boldsymbol{i}\in\Theta,\boldsymbol{m}(\boldsymbol{i})\ge\boldsymbol{j}} t_{\boldsymbol{i}} \prod_{k=1}^{d} w_{j_k}^{i_k}. \tag{13}$$

Therefore, the explicit form of the sparse grids approximation is given by

$$\tilde{f}_\Theta(\boldsymbol{x}) = \sum_{\boldsymbol{j}\in X(\Theta)} f(\boldsymbol{x}_{\boldsymbol{j}})\phi_{\boldsymbol{j}}(\boldsymbol{x}), \qquad Q_\Theta[f] = \sum_{\boldsymbol{j}\in X(\Theta)} f(\boldsymbol{x}_{\boldsymbol{j}}) w_{\boldsymbol{j}}. \tag{14}$$

For the non-nested case, we have

$$\tilde{f}_\Theta(\boldsymbol{x}) = \sum_{(\boldsymbol{i},\boldsymbol{j})\in X(\Theta)} f(\boldsymbol{x}_{\boldsymbol{j}}^{\boldsymbol{i}}) t_{\boldsymbol{i}} \prod_{k=1}^{d} \phi_{j_k}^{i_k}, \qquad Q_\Theta[f] = \sum_{(\boldsymbol{i},\boldsymbol{j})\in X(\Theta)} f(\boldsymbol{x}_{\boldsymbol{j}}^{\boldsymbol{i}}) t_{\boldsymbol{i}} \prod_{k=1}^{d} w_{j_k}^{i_k}. \tag{15}$$

Note, that some non-nested rules may share points, e.g., all one dimensional Gauss-Legendre rules with odd number of points include 0, thus, it is possible to have the same point for different index pairs $(\boldsymbol{i},\boldsymbol{j})$. Tasmanian automatically groups the functions and weights associated with those points and the library uses only unique points.

2.2 Global Grids: Approximation error

First we consider the polynomial space* for which the approximation is exact (i.e., no error). For interpolation $\mathcal{U}^{m(l)}[p] = p$ for all $p \in \mathcal{P}^{m(l)-1}$ and for quadrature rules there is a non-decreasing function $q : \mathbb{N} \to \mathbb{N}$

*$\mathcal{P}^l = span\{x^\nu : \nu \le l\}$ and for a lower multi-index set define $\mathcal{P}_\Lambda = span\{\boldsymbol{x}^{\boldsymbol{\nu}} : \boldsymbol{\nu} \le \boldsymbol{i}\}_{\boldsymbol{i}\in\Lambda}$.

so that $\mathcal{Q}^{m(l)}[p] = p$ for all $p \in \mathcal{P}^{q(l)}$. For Gauss rules $q(l) = 2m(l) - 1$, except Gauss-Patterson where $q(l) = \frac{3}{2}m(l) - \frac{1}{2}$. For other rules generally $q(l) = m(l) - 1$ except for rules with symmetric and odd number of points (e.g., Clenshaw-Curtis), where $q(l) = m(l)$ since any symmetric rule integrates exactly all odd power monomials.

For a general sparse grid point-wise approximation

$$G_\Theta[p] = p, \qquad \text{for all} \quad p \in \mathcal{P}_{\Lambda^m(\Theta)}, \qquad \text{where} \quad \Lambda^m(\Theta) = \bigcup_{\boldsymbol{i} \in \Theta} \{\boldsymbol{j} : \boldsymbol{j} \le \boldsymbol{m}(\boldsymbol{i}) - \mathbf{1}\}. \tag{16}$$

And for numerical quadrature

$$G_\Theta[p] = p, \qquad \text{for all} \quad p \in \mathcal{P}_{\Lambda^q(\Theta)}, \qquad \text{where} \quad \Lambda^q(\Theta) = \bigcup_{\boldsymbol{i} \in \Theta} \{\boldsymbol{j} : \boldsymbol{j} \le \boldsymbol{q}(\boldsymbol{i})\}^*. \tag{17}$$

Thus, $\Lambda^m(\Theta)$ and $\Lambda^q(\Theta)$ define the polynomial spaces associated with G_Θ.

Let $C^0(\Gamma)$ be the space of all continuous functions $f : \Gamma \to \mathbb{R}$ imbued with sup (or L^∞) norm $\|f\|_{C^0(\Gamma)} = \max_{\boldsymbol{x} \in \Gamma} |f(\boldsymbol{x})|$. The point-wise approximation error of a sparse grid is bounded by

$$\|f - G_\Theta[f]\|_{C^0(\Gamma)} \le \left(1 + \|G_\Theta\|_{C^0(\Gamma)}\right) \inf_{p \in \mathcal{P}_{\Lambda^m(\Theta)}} \|f - p\|_{C^0(\Gamma)}, \tag{18}$$

where $\|G_\Theta\|_{C^0(\Gamma)}$ is the operator norm of G_Θ (also called the Lebesgue constant)

$$\|G_\Theta\|_{C^0(\Gamma)} = \sup_{g \in C^0(\Gamma)} \frac{\|G_\Theta[g]\|_{C^0(\Gamma)}}{\|g\|_{C^0(\Gamma)}} = \max_{\boldsymbol{x} \in \Gamma} \sum_{\boldsymbol{j} \in X(\Theta)} |\psi_{\boldsymbol{j}}(\boldsymbol{x})|.$$

For the nested case $\psi_{\boldsymbol{j}}(\boldsymbol{x})$ are defined in (12) and (14), and for the non-nested case $\psi^{\boldsymbol{i}}_{\boldsymbol{j}}(\boldsymbol{x})$ are defined in (15) with the repeated points grouped together. The error in quadrature approximation is bounded as

$$\left| \int_\Gamma f(\boldsymbol{x}) \rho(\boldsymbol{x}) d\boldsymbol{x} - G_\Theta[f] \right| \le \left(\int_\Gamma \rho(\boldsymbol{x}) d\boldsymbol{x} + \sum_{\boldsymbol{j} \in X(\Theta)} |w_{\boldsymbol{j}}| \right) \inf_{p \in \mathcal{P}_{\Lambda^q(\Theta)}} \|f - p\|_{C^0(\Gamma)}, \tag{19}$$

and for the non-nested case the sum becomes $\sum_{(\boldsymbol{i},\boldsymbol{j}) \in X(\Theta)} |t_{\boldsymbol{i}} w^{\boldsymbol{i}}_{\boldsymbol{j}}|$ where weights corresponding to the same points are grouped together before taking the absolute value. Note, even if the one dimensional rule inducing the sparse grid has positive quadrature weights, since $t_{\boldsymbol{i}}$ can be negative, some of $w_{\boldsymbol{j}}$ can be negative.

The classical approach for sparse grids construction is to pre-define Θ according to some formula. Let $\boldsymbol{\xi}, \boldsymbol{\eta} \in \mathbb{R}^d$ be anisotropic weight vectors such that $\xi_k > 0$ for all $1 \le k \le d$, and let L indicate the "level" of the sparse grid approximation (the word "level" here is used loosely as the value of L has meaning only relative to $\boldsymbol{\xi}$). The classical anisotropic case takes

$$\Theta^{\boldsymbol{\xi}}(L) = \{\boldsymbol{i} \in \mathbb{N}^d : \boldsymbol{\xi} \cdot \boldsymbol{i} \le L\}^\dagger, \tag{20}$$

log-corrected or *curved* selection[46]

$$\Theta^{\boldsymbol{\xi},\boldsymbol{\eta}}(L) = \{\boldsymbol{i} \in \mathbb{N}^d : \boldsymbol{\xi} \cdot \boldsymbol{i} + \boldsymbol{\eta} \cdot \log(\boldsymbol{i} + \mathbf{1}) \le L\}^\ddagger, \tag{21}$$

*as with $\boldsymbol{m}(\boldsymbol{i})$ we take $\boldsymbol{q}(\boldsymbol{i}) = \bigotimes_{k=1}^d q(i_k)$

†Here $\cdot$ indicates the standard vector dot product $\boldsymbol{i} \cdot \boldsymbol{j} = \sum_{k=1}^d i_k j_k$.

‡Here $\log(\boldsymbol{i}) = \bigotimes_{k=1}^d \log(i_k)$

hyperbolic cross section

$$\Theta^{\boldsymbol{\xi}}(L) = \{\boldsymbol{i} \in \mathbb{N}^d : (\boldsymbol{i}+\mathbf{1})^{\boldsymbol{\xi}} \leq L\}. \tag{22}$$

Alternatively, the multi-index set Θ can be selected as the smallest lower set that results in a $\Lambda^m(\Theta)$ (or $\Lambda^q(\Theta)$) that includes a desired polynomial space (see [46] for details). Total degree space

$$\{\boldsymbol{j} \in \mathbb{N}^d : \boldsymbol{\xi}\cdot\boldsymbol{j} \leq L\} \subset \Lambda^m(\Theta), \quad \Rightarrow \quad \Theta^{\boldsymbol{\xi},m}(L) = \{\boldsymbol{i} \in \mathbb{N}^d : \boldsymbol{\xi}\cdot\boldsymbol{m}(\boldsymbol{i}-\mathbf{1}) \leq L\}^*, \tag{23}$$

or using a log-correction

$$\begin{aligned} & \{\boldsymbol{j} \in \mathbb{N}^d : \boldsymbol{\xi}\cdot\boldsymbol{j} + \boldsymbol{\eta}\cdot\log(\boldsymbol{j}+\mathbf{1}) \leq L\} \subset \Lambda^m(\Theta), \quad \Rightarrow \\ \Theta^{\boldsymbol{\xi},\boldsymbol{\eta},m}(L) \;=\; & \{\boldsymbol{i} \in \mathbb{N}^d : \boldsymbol{\xi}\cdot\boldsymbol{m}(\boldsymbol{i}-\mathbf{1}) + \boldsymbol{\eta}\cdot\log(\boldsymbol{m}(\boldsymbol{i}-\mathbf{1})+\mathbf{1}) \leq L\}, \end{aligned} \tag{24}$$

or hyperbolic cross section space

$$\{\boldsymbol{j} \in \mathbb{N}^d : (\boldsymbol{j}+\mathbf{1})^{\boldsymbol{\xi}} \leq L\} \subset \Lambda^m(\Theta), \quad \Rightarrow \Theta^{\boldsymbol{\xi},m}(L) = \{\boldsymbol{i} \in \mathbb{N}^d : (\boldsymbol{m}(\boldsymbol{i}-\mathbf{1})+\mathbf{1})^{\boldsymbol{\xi}} \leq L\}. \tag{25}$$

Tensor selection types (23), (24) and (25) target corresponding polynomial spaces associated with point-wise approximation, the corresponding quadrature formulas use q in place of m, i.e., for total degree space

$$\{\boldsymbol{j} \in \mathbb{N}^d : \boldsymbol{\xi}\cdot\boldsymbol{j} \leq L\} \subset \Lambda^q(\Theta), \quad \Rightarrow \quad \Theta^{\boldsymbol{\xi},m}(L) = \{\boldsymbol{i} \in \mathbb{N}^d : \boldsymbol{\xi}\cdot\boldsymbol{q}(\boldsymbol{i}-\mathbf{1})+\mathbf{1} \leq L\}^\dagger, \tag{26}$$

or using a log-correction

$$\begin{aligned} & \{\boldsymbol{j} \in \mathbb{N}^d : \boldsymbol{\xi}\cdot\boldsymbol{j} + \boldsymbol{\eta}\cdot\log(\boldsymbol{j}+\mathbf{1}) \leq L\} \subset \Lambda^q(\Theta), \quad \Rightarrow \\ \Theta^{\boldsymbol{\xi},\boldsymbol{\eta},q}(L) \;=\; & \{\boldsymbol{i} \in \mathbb{N}^d : \boldsymbol{\xi}\cdot(\boldsymbol{q}(\boldsymbol{i}-\mathbf{1})+\mathbf{1}) + \boldsymbol{\eta}\cdot\log(q(\boldsymbol{i}-\mathbf{1})+\mathbf{2}) \leq L\}, \end{aligned} \tag{27}$$

or hyperbolic cross section space

$$\{\boldsymbol{j} \in \mathbb{N}^d : (\boldsymbol{j}+\mathbf{1})^{\boldsymbol{\xi}} \leq L\} \subset \Lambda^q(\Theta), \quad \Rightarrow \Theta^{\boldsymbol{\xi},m}(L) = \{\boldsymbol{i} \in \mathbb{N}^d : (\boldsymbol{q}(\boldsymbol{i}-\mathbf{1})+\mathbf{1})^{\boldsymbol{\xi}} \leq L\}. \tag{28}$$

For example, $\Theta^{\mathbf{1},q}(L)$ constructed according to (26) will result in $G_{\Theta^{\mathbf{1},q}(L)}$ that integrates exactly all polynomials of total degree up to and including L. Similarly, $\Theta^{\mathbf{1},-\frac{\mathbf{1}}{\mathbf{2}},m}(L)$ will result in the dominant polynomial space defined in Proposition 8 and equation (8) in [5]. For more information about optimal and quasi-optimal polynomial approximation see [46] and references therein.

2.3 Global Grids: Sequence Grid

A sequence grid is constructed from a one dimensional nested rule with $m(l) = l+1$. The theoretical properties, i.e., (18) and (19), are identical to the global grid, however, the sequence grid uses representation in terms of Newton (as opposed to Lagrange) polynomials. Let

$$\phi_1(x) = 1, \quad \text{for } j > 1, \quad \phi_j(x) = \prod_{i=1}^{j-1} \frac{x - x_i}{x_j - x_i}, \quad \text{and for } \boldsymbol{j} \in \mathbb{N}^d, \quad \phi_{\boldsymbol{j}}(\boldsymbol{x}) = \prod_{k=1}^{d} \phi_{j_k},$$

*Here for notational convenience we assume that $m(-1) = 0$.

†Here for notational convenience we assume that $q(-1) = -1$.

where each ϕ_{j_k} is evaluated at the corresponding k-th component of $\boldsymbol{x}$. Then $G_\Theta[f]$ can be written as

$$G[f](\boldsymbol{x}) = \sum_{\boldsymbol{j} \in X(\Theta)} s_{\boldsymbol{j}} \phi_{\boldsymbol{j}}(\boldsymbol{x}), \tag{29}$$

where the surplus coefficients $s_{\boldsymbol{j}}$ satisfy the linear system of equation

$$\sum_{\boldsymbol{1} \le \boldsymbol{j} \le \boldsymbol{i}} s_{\boldsymbol{j}} \phi_{\boldsymbol{j}}(\boldsymbol{x}_{\boldsymbol{i}}) = f(\boldsymbol{x}_{\boldsymbol{i}}), \quad \text{for every } \boldsymbol{i} \in X(\Theta). \tag{30}$$

Note that all sparse grids induced by nested one dimensional rules can be written in the Newton form above, but Tasmanian implements sequence grids only for the case when $m(l) = l + 1$.

Computing and storing the coefficients $s_{\boldsymbol{j}}$ is more expensive then the weights $t_{\boldsymbol{i}}$, especially when $f(\boldsymbol{x})$ is a vector valued function where each output dimension of $f(\boldsymbol{x})$ requires a separate set of coefficients. However, computing the surpluses is a one time cost, followup evaluations of a sequence approximation are much cheaper since Newton polynomials are easier to construct. Thus, sequence grids are faster when a large number of evaluations of $G_\Theta[f]$ are desired.

2.4 Global Grids: Refinement

Global and sequence grids implemented in Tasmanian support two types of refinement based on surpluses and anisotropic coefficient decay. Given $G_\Theta[f]$ for some index set Θ, the goal of a refinement procedure is to produce an updated $\hat{\Theta}$ (with $\Theta \subset \hat{\Theta}$) such that $G_{\hat{\Theta}}[f]$ is more accurate and the additional indexes included in $\hat{\Theta}$ are "optimal" with respect to properties of $f(\boldsymbol{x})$ that are "inferred" from $G_\Theta[f]$. Note that refinement is supported only for grids induced by nested rules.

The surplus refinement is implemented only for grids induced by rules with $m(l) = l + 1$ (sequence and global grids alike). In that case $X(\Theta) = \Theta + \mathbf{1}$ and the refinement strategy considers the hierarchical surpluses (30). The set Θ is then expanded with indexes that are "close" to the indexes associated with large relative surpluses. Specifically:

$$\hat{\Theta} = \Theta \bigcup \left(\bigcup_{\boldsymbol{j} \in X(\Theta), |s_{\boldsymbol{j}}| > \epsilon \cdot f_{\max}} \left\{ \boldsymbol{i} \in \mathbb{N}^d : \sum_{k=1}^{d} |i_k - j_k - 1| = 1 \right\} \right), \tag{31}$$

where $f_{\max} = \max_{\boldsymbol{j} \in X(\Theta)} |f(\boldsymbol{x}_{\boldsymbol{j}})|$ and $\epsilon > 0$ is user specified tolerance. In the case when $f(\boldsymbol{x})$ has multiple outputs, if using a global grids (i.e., with Lagrange representation) then the user must specify one output to be used by the refinement criteria. The surpluses and $f_{\max}$ will be computed only for that one output. In contrast, a sequence grid computes and stores the surpluses for all outputs, thus, refinement can be easily done with either one output or all outputs simultaneously, in which case we refine for those $\boldsymbol{j} \in X(\Theta)$ such that $|s_{\boldsymbol{j}}| > \epsilon \cdot f_{\max}$ for any of the outputs. Here the purpose of the $f_{\max}$ is used to normalize the surpluses in case a vector valued function has outputs with significantly different scaling.

The second type of refinement is labeled *anisotropic*, and it is a two stage process. First, $G_\Theta[f]$ is expresses in terms of orthogonal multivariate Legendre polynomials, then anisotropic weights $\boldsymbol{\xi}$ and $\boldsymbol{\eta}$ are inferred from the decay rate of the coefficients. The refinement set $\hat{\Theta}$ is constructed according to (23) or (24) so that $G_{\hat{\Theta}}$ includes a desired minimum number of new points, where the minimum number of new points

exploits parallelism in computing the values of $f(\boldsymbol{x_j})$. Legendre expansion is computationally expensive, hence grids induced by rules with growth $m(l) = l + 1$ use hierarchical surpluses in place of the Legendre coefficients. As before, when $f(\boldsymbol{x})$ has multiple outputs, sequence and global grids can focus on a single output, and sequence grids can considers the largest normalized surplus, i.e., largest $|s_{\boldsymbol{j}}|/f_{\max}$ among all outputs. For more details on this type of refinement, see [46].

2.5 Global Grids: One dimensional rules

2.5.1 Chebyshev rules

Roots and extrema of Chebyshev polynomials are a common choice of one dimensional interpolation and integration rules and Tasmanian implements several Chebyshev based rules. The non-nested Chebyshev points are placed at the roots of the polynomials and the growth is either $m(l) = l+1$ or $m(l) = 2l+1$. The Clenshaw-Curtis[10] and Clenshaw-Curtis-zero (latter assumes the $f(\boldsymbol{x})$ is zero at $\partial\Gamma$) use only the nested Chebyshev points and $m(l)$ grows exponentially. The nested Fejer type 2[14] points use the extrema of the Chebyshev polynomials and also have exponential $m(l)$.

In addition, the library includes the more recently developed $\mathcal{R}$-Leja points[8]. Define $\{\theta_j\}_{j=1}^{\infty}$ as

$$\theta_1 = 0, \quad \theta_2 = \pi, \quad \theta_3 = \frac{\pi}{2}, \quad \text{for } j > 3,\ \theta_j = \begin{cases} \theta_{j-1} + \pi, & j \text{ is odd} \\ \frac{1}{2}\theta_{\frac{j}{2}+1}, & j \text{ is even} \end{cases} \tag{32}$$

then the $\mathcal{R}$-Leja points are given by $x_j = \cos(\theta_j)$ and the centered $\mathcal{R}$-Leja points start at $x_1 = 0$, $x_2 = 1$, $x_3 = -1$, and $x_j = \cos(\theta_j)$ for $j > 3$. The growth of the $\mathcal{R}$-Leja rule is $m(l) = l + 1$ and the centered rule allows for multiple definitions, namely odd rules $m(l) = 2l + 1$, the $\mathcal{R}$-Leja double-2 growth defined by

$$m(0) = 1, \quad m(1) = 3, \quad \text{for } l > 1, \quad m(l) = 2^{\lfloor \frac{l}{2} \rfloor + 1} \left(1 + \frac{l}{2} - \left\lfloor \frac{l}{2} \right\rfloor\right) + 1, \tag{33}$$

and the $\mathcal{R}$-Leja double-4 rule defined by

$$m(l) = 1, \quad m(l) = 3, \quad \text{for } l > 1, \quad m(l) = 2^{\lfloor \frac{l-2}{4} \rfloor + 2} \left(1 + \frac{l-2}{4} - \left\lfloor \frac{l-2}{4} \right\rfloor\right) + 1, \tag{34}$$

where $\lfloor x \rfloor = \max\{z \in \mathbb{Z} : z \leq x\}$ is the *floor* function, see [46] for more details.

Tasmanian also includes a shifted $\mathcal{R}$-Leja sequence defined by

$$x_1 = -\frac{1}{2}, \quad x_2 = \frac{1}{2}, \quad \text{for } j > 2, \quad x_j = \begin{cases} \sqrt{\frac{1+x_{(j+1)/2}}{2}}, & j \text{ is odd} \\ -x_{j-1}, & j \text{ is even} \end{cases} \tag{35}$$

which comes with growth $m(l) = l + 1$ or $m(l) = 2(l + 1)$. Table 1, summarizes all Chebyshev rules.

2.5.2 Gauss rules

The roots of orthogonal polynomials are a common choice for points for numerical integration due to the high level of precision. Orthogonality is defined with respect to a specific integration weight that often

Points	$m(l)$	$q(l)$	Note:
Chebyshev: Non-nested Chebyshev roots	$m(l) = l + 1$	$q(l) = l - 1 + (l \mod 2)$	very low Lebesgue constant
Clenshaw-Curtis: Nested Chebyshev roots	$m(0) = 1, m(l) = 2^l + 1$	$q(l) = m(l)$	very low Lebesgue constant
Clenshaw-Curtis-Zero: Nested Chebyshev roots	$m(l) = 2^{l+1} - 1$	$q(l) = 2^l$	assumes $f(\boldsymbol{x}) = 0$ at $\partial\Gamma$
Fejer type 2: Nested Chebyshev extrema	$m(l) = 2^{l+1} - 1$	$q(l) = 2^l$	no points at $\partial\Gamma$
$\mathcal{R}$-Leja: See (32)	$m(l) = l + 1$	$q(l) = l - 1 + (l \mod 2)$	see [8, 46]
$\mathcal{R}$-Leja odd: Centered $\mathcal{R}$-Leja	$m(l) = 2l + 1$	$q(l) = m(l)$	see [8, 46]
$\mathcal{R}$-Leja double 2: Centered $\mathcal{R}$-Leja	see (33)	$q(l) = m(l)$	see [8, 46]
$\mathcal{R}$-Leja double 4: Centered $\mathcal{R}$-Leja	see (34)	$q(l) = m(l)$	see [8, 46]
$\mathcal{R}$-Leja shifted: See (35)	$m(l) = l + 1$	$q(l) = m(l) - 1$	see [9]
$\mathcal{R}$-Leja shifted even: See (35)	$m(l) = 2(l + 1)$	$q(l) = 2l + 1$	see [9]

Table 1. Summary of the available Chebyshev rules.

times requires additional parameters α and/or β. The Gauss rules also include the Hermite and Laguerre polynomials that assume unbounded domain. Gauss rules are usually non-nested, have growth $m(l) = l+1$, and precision $q(l) = 2l + 1$. Odd versions of the rules use growth $m(l) = 2l + 1$ and $q(l) = 4l + 1$, and when coupled with *qpcurved* or *qptotal* tensor selection the odd versions of the Gauss rules usually result in sparse grids with fewer points.

Gauss-Patterson[37] points are a notable exception in most ways. The Patterson construction uses the Legendre orthogonal polynomials and imposes the additional requirement that the points are nested, which leads to a rule with growth $m(l) = 2^{l+1} - 1$ and precision $q(l) = \frac{3}{2}m(l) - \frac{1}{2} = 3 \cdot 2^l - 2$. Note that the construction of the Gauss-Patterson points and weights is a computationally expensive and ill-conditioned problem, Tasmanian does not include code that computes the point and weight, instead the first 9 levels are hard-coded into the library. The 9 levels should give sufficient precision for most applications, while the custom rule capabilities of the library can be used to extend beyond that limit, assuming the user provides Gauss-Patterson points and weights for higher levels. Summary of all Gauss rules is listed in Table 2.

2.5.3 Greedy rules

Tasmanian implements a number of rules using sequences of points that are based on greedy optimization. The most well known rule uses the Leja points[11], where

$$x_1 = 0, \qquad \text{for } j > 1 \quad x_{j+1} = \operatorname*{argmax}_{x \in [-1,1]} \prod_{i=1}^{j} |x - x_i|. \tag{36}$$

Name	Generalized Integral	Notes
Gauss-Patterson:	$\int_a^b f(x)dx$	The only nested rule Canonical: $a=-1, b=1$
Gauss-Legendre:	$\int_a^b f(x)dx$	Highest 1-D exactness Canonical: $a=-1, b=1$
Gauss-Chebyshev type 1:	$\int_a^b f(x)(b-x)^{-0.5}(x-a)^{-0.5}dx$	Canonical: $a=-1, b=1$
Gauss-Chebyshev type 2:	$\int_a^b f(x)(b-x)^{0.5}(x-a)^{0.5}dx$	Canonical: $a=-1, b=1$
Gauss-Gegenbauer:	$\int_a^b f(x)(b-x)^{\alpha}(x-a)^{\alpha}dx$	Must specify α Canonical: $a=-1, b=1$
Gauss-Jacobi:	$\int_a^b f(x)(b-x)^{\alpha}(x-a)^{\beta}dx$	Must specify α, β Canonical: $a=-1, b=1$
Gauss-Laguerre:	$\int_a^{\infty} f(x)(x-a)^{\alpha}e^{-b(x-a)}dx$	Must specify α Canonical: $a=0, b=1$
Gauss-Hermite:	$\int_{-\infty}^{\infty} f(x)(x-a)^{\alpha}e^{-b(x-a)^2}dx$	Must specify α Canonical: $a=0, b=1$

Table 2. Summary of the available Gauss rules.

Similar construction can be done using the extrema of the Lebesgue function

$$x_1 = 0, \qquad \text{for } j > 1 \quad x_{j+1} = \underset{x\in[-1,1]}{\text{argmax}} \sum_{j'=1}^{j} \prod_{i=1, i\neq j'}^{j} \left|\frac{x - x_i}{x_{j'} - x_i}\right|. \tag{37}$$

We can greedily minimize the norm of $\mathcal{U}^{m(j+1)}$, where $x_1 = 0$ and for $j > 1$

$$x_{j+1} = \underset{x\in[-1,1]}{\text{argmin}} \max_{y\in[-1,1]} \prod_{i=1}^{j} \left|\frac{y - x_i}{x - x_i}\right| + \sum_{j'=1}^{j} \left|\frac{y - x}{x_{j'} - x}\right| \prod_{i=1, i\neq j'}^{j} \left|\frac{y - x_i}{x_{j'} - x_i}\right| \tag{38}$$

or minimizing the norm of the surplus operator $\Delta^{m(j+1)}$, where $x_1 = 0$ and for $j > 1$

$$x_{j+1} = \underset{x\in[-1,1]}{\text{argmin}} \max_{y\in[-1,1]} \left(1 + \sum_{i=1}^{j} \prod_{j'=1, j'\neq i}^{j} \left|\frac{x - x_{j'}}{x_i - x_{j'}}\right|\right) \prod_{j'=1}^{j} \left|\frac{y - x_{j'}}{x - x_{j'}}\right|. \tag{39}$$

In all cases the growth can be set to $m(l) = l + 1$ or $m(l) = 2l + 1$. However, unlike the $\mathcal{R}$-Leja points, the odd rules here do not result in symmetric distribution of the points, hence $q(l) = m(l) - 1$ (and $q(0) = 1$). For a numerical survey of the properties of interpolants constructed from the above sequences, see [46]. Note that quadrature rules using the above sequences can potentially result in zero weights (i.e., $w_j = 0$ for some j), Tasmanian does NOT automatically check if the weights are zero. The greedy rules are intended for interpolation purposes and are not the best rules to use for numerical integration. A list of the greedy rules is given in Table 3.

Name	Points	$m(l)$
Leja: **Leja odd:**	See (36)	$m(l) = l + 1$ $m(l) = 2l + 1$
Max-Lebesgue: **Max-Lebesgue odd:**	See (37)	$m(l) = l + 1$ $m(l) = 2l + 1$
Min-Lebesgue: **Min-Lebesgue odd:**	See (38)	$m(l) = l + 1$ $m(l) = 2l + 1$
Min-Delta: **Min-Delta odd:**	See (39)	$m(l) = l + 1$ $m(l) = 2l + 1$

Table 3. Summary of the available greedy sequence rules.

2.6 Fourier Grids

For cases where the interpolant of $f(\boldsymbol{x})$ must be periodic in derivatives as well as function values, Tasmanian implements sparse interpolation with a Fourier basis, for more details see [20, 24, 42]. The one-dimensional (nested) rule assumes a canonical domain of $[0, 1]$ with the nodes

$$x_1 = 0, \qquad x_j = \frac{\left\lfloor \frac{3}{2}\left(j - 1 - 3^{\lfloor \log_3(j-1) \rfloor}\right)\right\rfloor}{3^{\lfloor \log_3(j-1)+1 \rfloor}}, \quad \text{for } j > 1.$$

Note that each level contains 3^l nodes, which allows us to preserve the nested structure, have levels with complete exponent (see below) and use radix-3 Fast-Fourier-Transform (FFT) algorithm. We define the exponential functions

$$\phi_j(x) = \exp\left(2\pi\mathcal{I}x(-1)^{j+1}\lfloor j/2 \rfloor\right),$$

where $\mathcal{I}$ is the unit complex number, i.e., $\mathcal{I}^2 = -1$. The interpolant is real values, which means that the effective basis functions at level l are

$$\left\{ \cos(2\pi\omega x), \sin(2\pi\omega x) \ : \ \omega \in \mathbb{Z}, \ |\omega| \leq \frac{3^l - 1}{2} \right\},$$

and the coefficients of the basis functions are computed using FFT.

Let $f : [0, 1] \to \mathbb{R}$ and consider the 1-D Fourier interpolant at level l, i.e., the interpolant $\mathcal{U}^{3^l}[f](x)$ matching $f(x)$ at nodes x_s for $s \leq 3^l$. The interpolant is given by

$$\mathcal{U}^{3^l}[f](x) = \sum_{j=1}^{3^l} \Re\left(\hat{f}_j \phi_j(x)\right)$$

where $\Re$ indicates the real part of a number and $\hat{f}_j$ is a special reordering of the discrete Fourier coefficients of reordered sequence $f(x_j)$ (i.e., the index has to be reordered twice). First, we define $f_i = f(x_s)$ such that for $i = 1, 2, \cdots, 3^l$ the corresponding x_s values are in ascending order, i.e., we f_i are the function values reordered in domain space according to x_s. Second, we take the discrete Fourier transform (using FFT algorithm) and obtain Fourier coefficients $\hat{f}_i$. Finally, the Fourier coefficients are reordered again to match the basis, i.e.,

$$\hat{f}_j = \hat{f}_i, \qquad \text{where } i = \begin{cases} \lfloor j/2 \rfloor, & (-1)^j < 0, \\ 3^l - \lfloor j/2 \rfloor, & (-1)^j > 0. \end{cases}$$

Internally, Tasmanian implements the re-indexing inline with negligible overhead, but it is noteworthy that there is no strong connection between the coefficient associated with $\phi_j(x)$ and the spacial node x_j, i.e., unlike other types of grids, the coefficient cannot be interpreted as hierarchical surplus.

Extending the one dimensional interpolant to multidimensional context is done analogously to the sparse grids construction with Global rules. We define

$$\boldsymbol{x_j} = \bigotimes_{k=1}^{d} x_{j_k}, \qquad \boldsymbol{\phi_j}(\boldsymbol{x}) = \prod_{k=1}^{d} \phi_{j_k}, \qquad \mathcal{U}^{\boldsymbol{i}} = \bigotimes_{k=1}^{d} \mathcal{U}^{3^{i_k}},$$

and a sparse interpolant

$$G_\Theta[f](\boldsymbol{x}) = \sum_{\boldsymbol{i} \in \Theta} t_{\boldsymbol{i}} \mathcal{U}^{\boldsymbol{i}}[f](\boldsymbol{x}),$$

where Θ is some lower set and the weights $t_{\boldsymbol{i}}$ are the same as in (11).

The function space follows logic similar to that of Global grids with trigonometric frequency in place of polynomial order. The theoretical results for the target polynomial space are similar, i.e., functions of finite differentiability require hyperbolic-cross-section space, while analytic functions use total degree space. One note is that a single trigonometric frequency (greater than 0), requires two points and two basis functions to capture both $\sin(\cdot)$ and $\cos(\cdot)$ components. This relation is handled automatically when building the grid with types `type_iptotal` and `type_iphyperbolic`, i.e., Tasmanian automatically uses the correct interpretation. Finally, adaptive refinement can be performed in the Fourier context in the same way as Global grids[32].

2.7 Local Polynomial Grids: Hierarchical interpolation rule

Local polynomial grids are constructed from equidistant points and use functions with support restricted to a neighborhood of each point. The local support of the functions allow the employment of locally adaptive strategies and thus local grids are suitable for approximating functions with sharp behavior, e.g., large fluctuation of the gradient. Similar to the global grids, local grids are constructed from tensors of points and functions in one dimension. In contrast to global grids, local grids use functions with local support and very strict hierarchy. For in depth analysis of the properties of the local grids see [19, 31, 30, 43].

Let $\{x_j\}_{j=0}^{\infty} \in [-1, 1]$ be a sequence of nodes (w.l.o.g., we assume that we are working on the canonical domain $[-1, 1]$) and let $\{\Delta x_j\}_{j=0}^{\infty}$ indicate the "resolution" of our approximation at point x_j, i.e., the support of the associated function. In addition, we have the hierarchy defined by the *parents* and *children* sets

$$\begin{aligned} P_j &= \{i \in \mathbb{N} : x_i \text{ is a parent of } x_j\}, \\ O_j &= \{i \in \mathbb{N} : x_i \text{ is a child (offspring) of } x_j\}, \end{aligned}$$

where P_j can have more than one element. For a particular example of such hierarchies, see Section 2.9. We assume that P_j and O_j define a partial order of the points and let $h : \mathbb{N} \to \mathbb{N}$ map each point to a place in the hierarchy also called *level*, i.e.,

$$h(j) = \begin{cases} 0, & P_j = \emptyset \\ h(i) + 1, & \text{for any } i \in P_j \end{cases}$$

We define the ancestry set A_j

$$A_j = \{i \in \mathbb{N} : h(i) \leq h(j) \quad \text{and} \quad (x_i - \Delta x_i, x_i + \Delta x_i) \cap (x_j - \Delta x_j, x_j + \Delta x_j) \neq \emptyset\}$$

In order to construct the basis functions, for each x_j we consider the set of p nearest ancestors

$$F_j^{(p)} = \underset{F \subset A_j, \#F = p}{\operatorname{argmin}} \sum_{i \in F} |x_i - x_j|,$$

where $\#F$ indicates the number of elements of F. Note that $F_j^{(p)}$ is defined only for $p \leq \#A_j$.

The functions associated with a hierarchy can have various polynomial order $p \geq 0$. For constant functions

$$\phi_j^{(0)}(x) = \begin{cases} 1, & x \in (x_j - \Delta x_j, x_j + \Delta x_j) \\ 0, & x \notin (x_j - \Delta x_j, x_j + \Delta x_j) \end{cases}$$

For linear functions

$$\phi_j^{(1)}(x) = \begin{cases} 1 - \frac{|x - x_j|}{\Delta x_j} & x \in (x_j - \Delta x_j, x_j + \Delta x_j) \\ 0, & x \notin (x_j - \Delta x_j, x_j + \Delta x_j) \end{cases}$$

and functions of arbitrary order $p > 1$

$$\phi_j^{(p)}(x) = \begin{cases} \prod_{i \in F_j^{(p)}} \frac{x - x_i}{x_j - x_i}, & x \in (x_j - \Delta x_j, x_j + \Delta x_j) \\ 0, & x \notin (x_j - \Delta x_j, x_j + \Delta x_j) \end{cases}$$

Note that a function can have order p only if the corresponding $F_j^{(p)}$ exists, i.e., $h(j)$ is large enough. Tasmanian constructs local polynomial grids by automatically using the largest p available for each $\phi_j^{(p)}(x)$, optionally the library can be restricted p to a maximum user defined value. In the rest of this discussion, we would omit p.

We extend the one dimensional hierarchy to a d-dimensional context using multi-index notation*

$$\boldsymbol{x_j} = \bigotimes_{k=1}^{d} x_{j_k}, \qquad \phi_{\boldsymbol{j}}(\boldsymbol{x}) = \prod \phi_{j_k}, \qquad supp\{\phi_{\boldsymbol{j}}(\boldsymbol{x})\} = \bigotimes_{k=1}^{d} (x_{j_k} - \Delta x_{j_k}, x_{j_k} + \Delta x_{j_k}),$$

where each $\prod \phi_{j_k}$ is evaluated at the corresponding k-th entry of $\boldsymbol{x}$ and $supp\{\phi_{\boldsymbol{j}}(\boldsymbol{x})\}$ indicate the support of $\phi_{\boldsymbol{j}}(\boldsymbol{x})$. Parents and children are associated with different directions

$$P_j^{(k)} = \{\boldsymbol{i} \in \mathbb{N}^d : \boldsymbol{i} \underset{k}{=} \boldsymbol{j}^{\dagger} \text{ and } i_k \in P_{j_k}\} \qquad O_j^{(k)} = \{\boldsymbol{i} \in \mathbb{N}^d : \boldsymbol{i} \underset{k}{=} \boldsymbol{j} \text{ and } i_k \in O_{j_k}\}$$

and the level of a multi-index is $h(\boldsymbol{j}) = \sum_{k=1}^{d} h(j_k)$. The multidimensional ancestry set is

$$A_{\boldsymbol{j}} = \left\{\boldsymbol{i} \in \mathbb{N}^d : h(\boldsymbol{i}) \leq h(\boldsymbol{j}) \quad \text{and} \quad supp\{\phi_{\boldsymbol{i}}(\boldsymbol{x})\} \bigcap supp\{\phi_{\boldsymbol{j}}(\boldsymbol{x})\} \neq \emptyset\right\}$$

*Similar to the global grids, $\mathbb{N}$ indicates the set of non-negative integers, and $W, F, A, P, O, B, X \subset \mathbb{N}^d$ denote sets of multi-indexes.

†Here by $\boldsymbol{i} \underset{k}{=} \boldsymbol{j}$ we mean that $\boldsymbol{i}$ and $\boldsymbol{j}$ have the same components in all but the k-th direction

For $f : \Gamma \to \mathbb{R}$, a multi-dimensional interpolant of $f(\boldsymbol{x})$ is defined by a set of points X so that

$$G_X[f] = \sum_{\boldsymbol{j} \in X} s_{\boldsymbol{j}} \phi_{\boldsymbol{j}}(\boldsymbol{x}),$$

where the surplus coefficients $s_{\boldsymbol{j}}$ are chosen such that $G_X[f](\boldsymbol{x_i}) = f(\boldsymbol{x_i})$ for all $\boldsymbol{i} \in X$, specifically, by definition of $\phi_{\boldsymbol{j}}(\boldsymbol{x})$

$$s_{\boldsymbol{j}} = f(\boldsymbol{x_j}) - \sum_{\boldsymbol{i} \in A_{\boldsymbol{j}}} s_{\boldsymbol{i}} \phi_{\boldsymbol{i}}(\boldsymbol{x_j}). \tag{40}$$

In the case when $f(\boldsymbol{x})$ is a vector valued function, a separate set of surplus coefficients is computed for each output. When Tasmanian first creates a local polynomial grid, the set of points is chosen so that

$$X = \{\boldsymbol{j} \in \mathbb{N}^d : h(\boldsymbol{j}) \leq L\}, \tag{41}$$

for some use specified L.

2.8 Local Polynomial Grids: Adaptive refinement

Locally adaptive grids are best utilized with an appropriate refinement strategy. Suppose we have constructed $G_X[f]$ for some X and consider an updated $\hat{X}$ so that new points are added only in the region of Γ where $G_X[f]$ sharply deviates from $f(\boldsymbol{x})$. The surpluses $s_{\boldsymbol{j}}$ are a good local error indicator, and thus we define $\hat{X}$ that contains only indexes that are parents or children of indexes $\boldsymbol{j}$ associated with large $s_{\boldsymbol{j}}$.

First, we define the set of large surpluses

$$B = \left\{ \boldsymbol{j} \in X : \frac{|s_{\boldsymbol{j}}|}{f_{\max}} > \epsilon \right\},$$

where $\epsilon > 0$ is desired tolerance and $f_{\max} = \max_{\boldsymbol{i} \in X} |f(\boldsymbol{x_i})|$. When $f(\boldsymbol{x})$ is a vector valued function, an index $\boldsymbol{j}$ is included in B if any of the outputs has normalized surpluses larger than ϵ. Tasmanian implements 4 different refinement strategies, where $\hat{X}$ is selected by including parents and/or children of $\boldsymbol{j} \in B$ in different directions. This is done based on consideration of "orphan" directions and directional surpluses.

For each index in $\boldsymbol{j}$, we define the "orphan" directions

$$T_{\boldsymbol{j}} = \left\{ k \in \{1, 2, \ldots, d\} : P^{(k)}_{\boldsymbol{j}} \not\subset X \right\},$$

thus, $T_{\boldsymbol{j}}$ contains the directions where we have missing parents. We also consider directional surpluses, let

$$W^{(k)}_{\boldsymbol{j}} = \left\{ \boldsymbol{i} \in X : \boldsymbol{i} \underset{k}{=} \boldsymbol{j} \right\}, \qquad G_{W^{(k)}_{\boldsymbol{j}}}[f] = \sum_{\boldsymbol{i} \in W^{(k)}_{\boldsymbol{j}}} c^{(k)}_{\boldsymbol{i}} \phi_{\boldsymbol{i}}(\boldsymbol{x}),$$

where we have a set of the one directional surpluses $c^{(k)}_{\boldsymbol{i}}$ associated with each index $\boldsymbol{j}$, however, we focus our attention only to $c^{(k)}_{\boldsymbol{j}}$. The set of large one directional surpluses is

$$C_{\boldsymbol{j}} = \left\{ k \in \{1, 2, \ldots, d\} : \frac{\left| c^{(k)}_{\boldsymbol{j}} \right|}{f_{\max}} > \epsilon \right\}.$$

The classical refinement strategy constructs $\hat{X}$ by adding the children of $\boldsymbol{j} \in B$, i.e.,

$$\hat{X} = X \bigcup \left(\bigcup_{\boldsymbol{j} \in B} \bigcup_{k \in \{1,2,\ldots,d\}} O_{\boldsymbol{j}}^{(k)} \right). \tag{42}$$

However, the classical strategy can lead to instability around orphan points, hence, the parents-first approach adds parents before the children

$$\hat{X} = X \bigcup \left(\bigcup_{\boldsymbol{j} \in B} \left(\bigcup_{k \in T_{\boldsymbol{j}}} P_{\boldsymbol{j}}^{(k)} \right) \bigcup \left(\bigcup_{k \notin T_{\boldsymbol{j}}} O_{\boldsymbol{j}}^{(k)} \right) \right). \tag{43}$$

Large surplus signifies large local error, however, refinement doesn't have to be done in all directions, thus, the directional refinement uses $k \in C_{\boldsymbol{j}}$, i.e.,

$$\hat{X} = X \bigcup \left(\bigcup_{\boldsymbol{j} \in B} \bigcup_{k \in C_{\boldsymbol{j}}} O_{\boldsymbol{j}}^{(k)} \right). \tag{44}$$

Combining the parents-first and directional approach leads to the family-direction-selective (FDS) method

$$\hat{X} = X \bigcup \left(\bigcup_{\boldsymbol{j} \in B} \left(\bigcup_{k \in C_{\boldsymbol{j}} \cap T_{\boldsymbol{j}}} P_{\boldsymbol{j}}^{(k)} \right) \bigcup \left(\bigcup_{k \in C_{\boldsymbol{j}} \setminus T_{\boldsymbol{j}}} O_{\boldsymbol{j}}^{(k)} \right) \right). \tag{45}$$

For more details about the four refinement strategies see [43].

2.9 Local Polynomial Grids: One dimensional rules

Tasmanian implements three specific one dimensional hierarchical rules: standard rule with Δx_j decreasing by 2 at each level, a semi-local rule where global basis is used for levels 0 and 1, and a modified rule that assumes $f(\boldsymbol{x}) = 0$ at $\partial\Gamma$.

The standard local rule is given by

$$x_0 = 0, \qquad x_1 = -1, \qquad x_2 = 1, \qquad \text{for } j > 2 \quad x_j = (2j-1) \times 2^{-\lfloor \log_2(j-1) \rfloor} - 3, \tag{46}$$

where $\lfloor x \rfloor = \max\{z \in \mathbb{Z} : z \leq x\}$ is the *floor* function. The parent sets are

$$P_0 = \emptyset, \qquad P_1 = \{0\}, \qquad P_2 = \{0\}, \qquad P_3 = \{1\}, \qquad \text{for } j > 3 \quad P_j = \left\{ \left\lfloor \frac{j+1}{2} \right\rfloor \right\},$$

and the offspring sets are

$$O_0 = \{1, 2\}, \qquad O_1 = \{3\}, \qquad O_2 = \{4\}, \qquad \text{for } j > 2 \quad O_j = \{2j-1,\, 2j\}.$$

The level function is

$$h(j) = \begin{cases} 0, & j = 0, \\ 1, & j = 1, \\ \lfloor \log_2(j-1) \rfloor + 1, & j > 1, \end{cases}$$

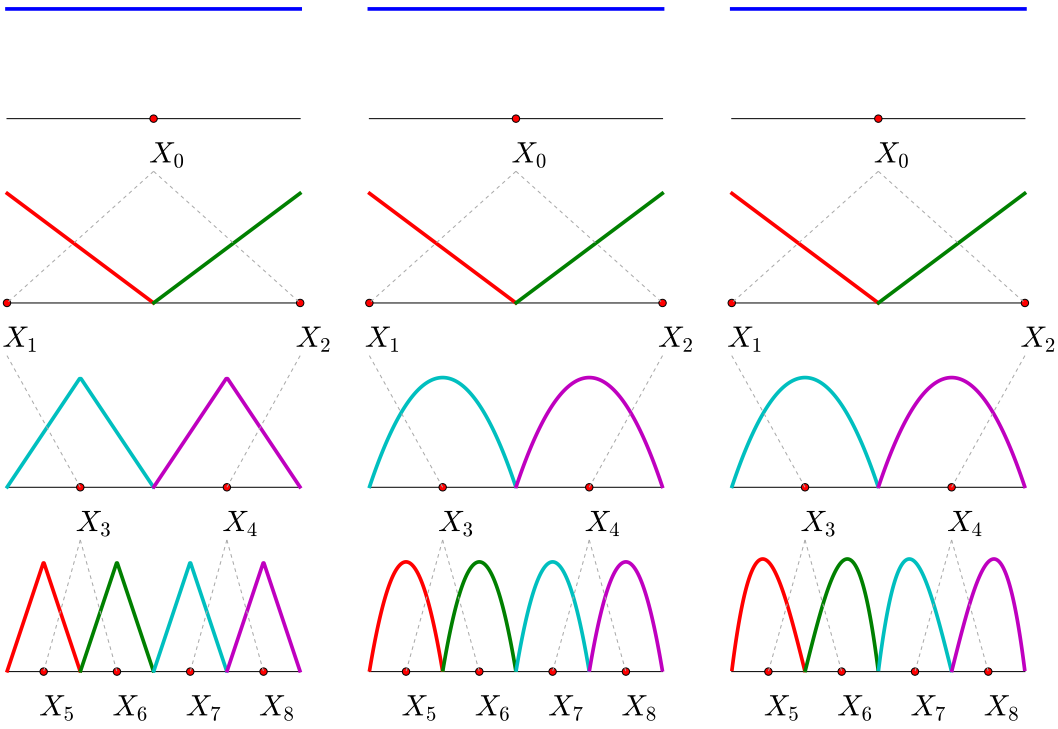

Figure 1. Local polynomial points (*rule_localp*) and functions, left to right: linear, quadratic, and cubic functions.

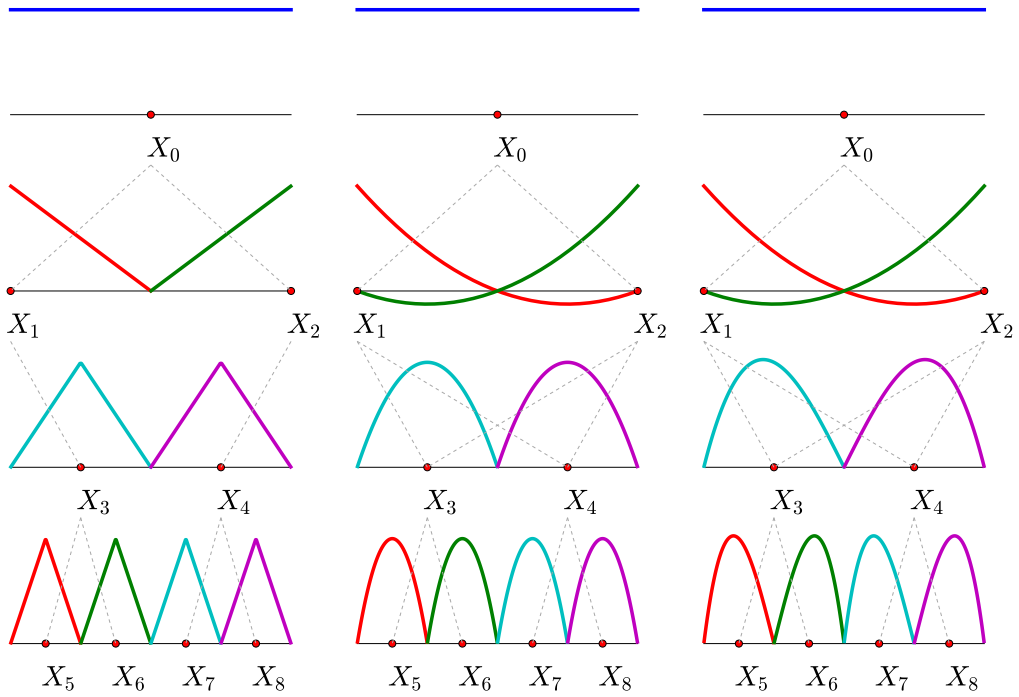

Figure 2. Semi-local polynomial points (*rule_semilocalp*) and functions, left to right: linear, quadratic, and cubic functions.

and the resolution Δx_j is given by $\Delta x_0 = 1$ and for $j > 0$ we have $\Delta x_j = 2^{-h(j)+1}$. Figure 1 shows the first four levels of the linear, quadratic, and cubic functions.

A modification to the standard rule uses the same points, however, functions at level $l = 1$ with degree higher than linear will have global support, i.e., if $p > 1$ then $\Delta x_1 = \Delta x_2 = 2$. In addition, for the purpose of parents refinement (43) and (45) we use $P_3 = P_4 = \{1, 2\}$. The modified rule sacrifices resolution and gains higher polynomial order, thus, the semi-local approach is better suited for functions with "smoother" behavior. Figure 2 shows the linear, quadratic, and cubic semi-local functions. Note: there is no difference between the linear versions of the local and semi-local rules.

An alternative local rule does not put points on the boundary and implicitly assumes that $f(\boldsymbol{x}) = 0$ at $\partial\Gamma$. The hierarchy is defined as

$$x_0 = 0, \qquad \text{for } j > 0 \quad x_j = (2j + 3) \times 2^{-\lfloor \log_2(j+1) \rfloor} - 3, \tag{47}$$

The parent sets are

$$P_0 = \emptyset, \qquad \text{for } j > 0 \quad P_j = \left\{ \left\lfloor \frac{j-1}{2} \right\rfloor \right\},$$

and the offspring sets are $O_j = \{2j + 1, 2j + 2\}$. The level function is $h(j) = \lfloor \log_2(j + 1) \rfloor$ and the

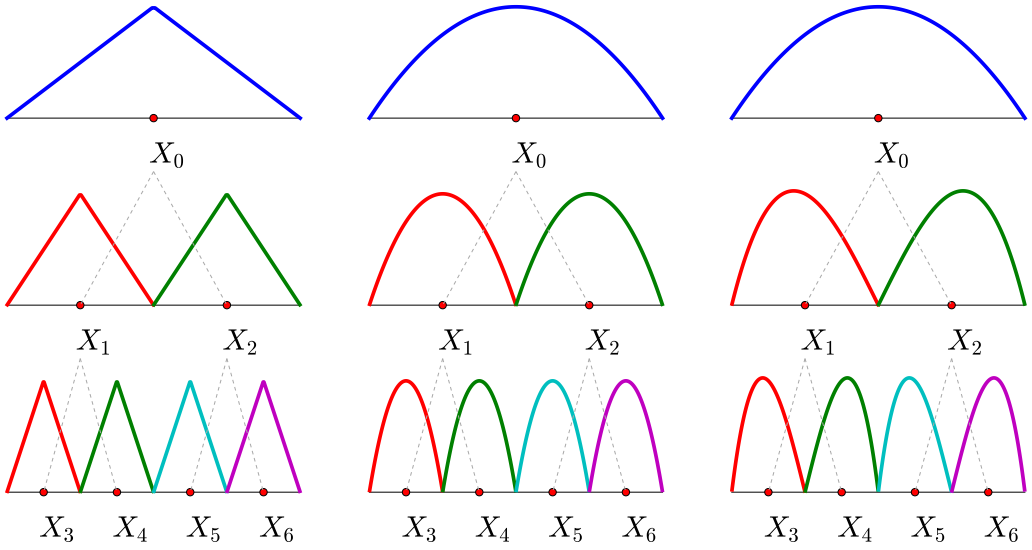

Figure 3. Semi-local polynomial points (*rule_localp0*) and functions, left to right: linear, quadratic, and cubic functions.

resolution Δx_j is given by $\Delta x_0 = 2^{-h(j)}$. Figure 3 shows the first three levels of the linear, quadratic, and cubic functions.

A rule with piece-wise constant (and discontinuous) basis is also provided within Tasmanian. Figure 4 shows the first four levels and the associated parents-offspring relations, see [44] for details.

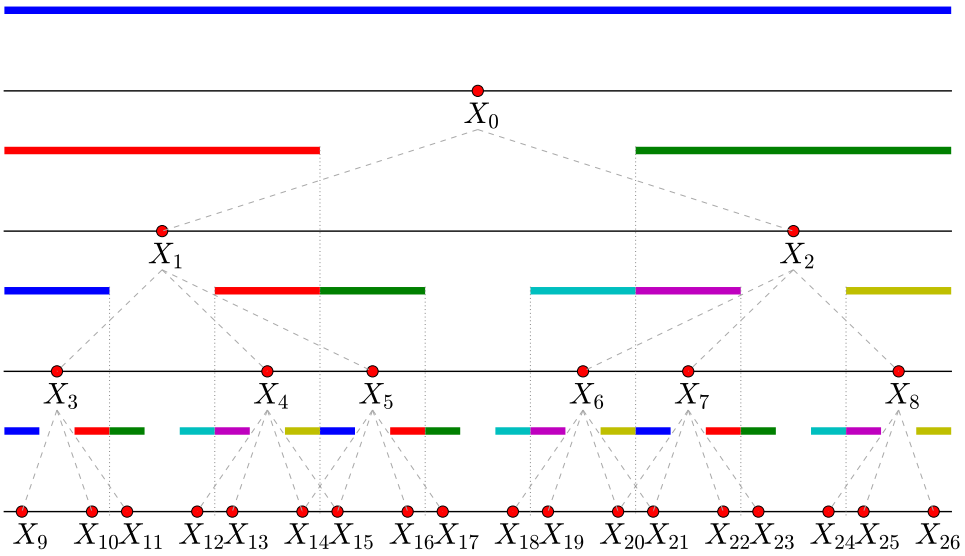

Figure 4. Semi-local polynomial points (*rule_localp*) rule with order 0.

2.10 Wavelets

Tasmanian, in addition to the local polynomial rules, also implements wavelet rules with order 1 and 3. The hierarchy followed by the wavelets as well as the refinement strategies are very similar to the local grids. The differences are as follows:

- The zeroth levels of wavelet rules of order 1 and 3, have 3 and 5 points respectively. This is a sharp contrast to the single point of of the polynomial rules, since level 0 wavelet grid has 3^d (or 5^d) points in d-dimensions (as opposed to a single point). See Figure 5.

- Wavelet rules have larger Lebesgue constant, which is due to the large magnitude of the boundary wavelet functions. This can lead to instability of the wavelet interpolant around the boundary of the domain.

- The linear system of equations associated with the wavelet surpluses is not triangular, hence a sparse matrix has to be inverted every time values are loaded into the interpolant. This leads to a significantly higher computational cost in manipulating the wavelet grids, especially in loading values and performing direction selective refinement.

- Wavelets form a Riesz basis, which over-simplistically means that the wavelet surpluses are much sharper indicators of the local error and hence wavelet based refinement strategy "could" generate a grid that is more accurate and has fewer points. The quotations around the word "could" relate to the point about the Lebesgue constant.

- For more details about wavelets, see [19, 22, 48].

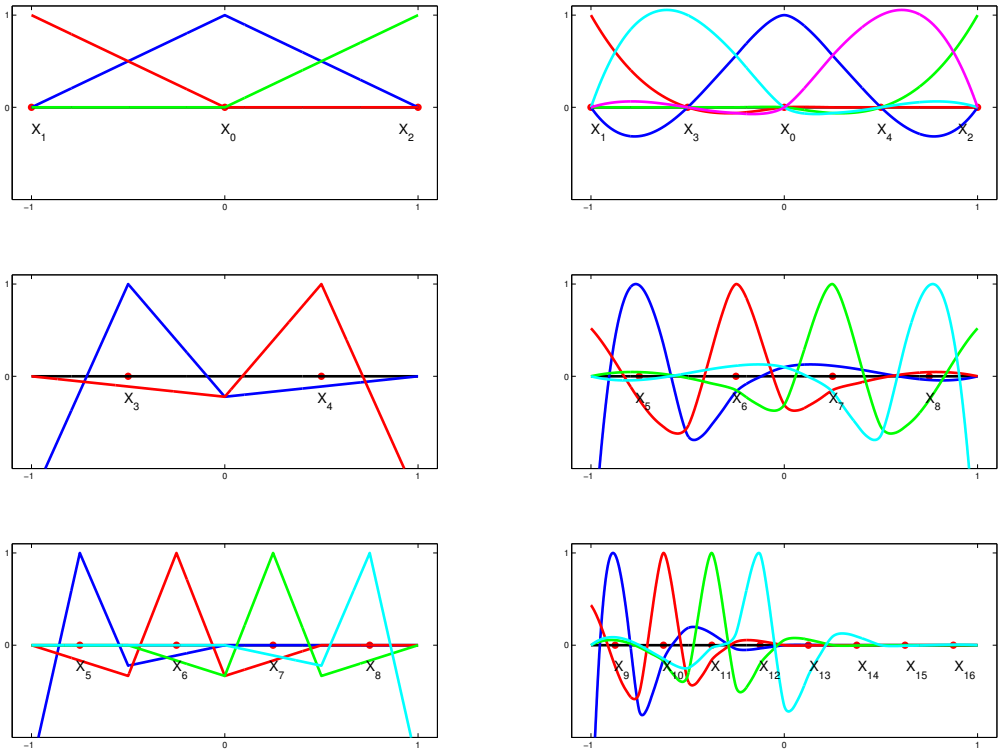

Figure 5. The first three levels for wavelets of order 1 (left) and 3 (right). The functions associated with x_{13}, x_{14}, x_{15}, and x_{16} are purposely omitted to reduce the clutter on the plot, since the funcitons are mirror images of the those associated with x_{12}, x_{11}, x_{10}, and x_9 respectively.

2.11 Domain Transformation

Sparse grids are build on canonical 1D domain $[-1, 1]$, with the exception of Gauss-Laguerre and Gauss-Hermite rules that use $[0, \infty)$ and $(-\infty, \infty)$ respectively. Linear transformation can be applied to translate $[-1, 1]$ to an arbitrary interval $[a, b]$, for unbounded domain we can apply shift a and scaling b. This simple linear transformation will not affect the properties of the grid, i.e., function space spanned by the basis or the Lebesgue constant. Thus, the a and b parameters are used to simplify implementation and generate a grid on a domain consistent with the range of the input of an arbitrary function $f(\boldsymbol{x})$. However, non-linear transformation can also be used with the goal of accelerating convergence.

2.11.1 Conformal Map

For simplicity, assume that $f(x)$ is a one dimensional function defined on $[-1, 1]$. Then a conformal map is any monotonic strictly increasing $g(x)$ such that

$$g : [-1, 1] \to [-1, 1], \qquad g(-1) = -1, \qquad \text{and} \quad g(1) = 1,$$

Then, instead of constructing a sparse grids rule that integrates or interpolates $f(x)$, we construct a rule for $f(g(x))$, with the hope that the composed function will be easier to approximate, e.g., have larger region of analyticity[28, 2].

In case of a quadrature rule, we note that

$$\int f(x)dx = \int f(g(x))g'(x)dx,$$

thus if we have a quadrature rule $\int f(g(x))dx \approx \sum_{\boldsymbol{i} \in X(\theta)} \omega_{\boldsymbol{i}} f(g(x_{\boldsymbol{i}}))$, then

$$\int f(x)dx \approx \sum_{\boldsymbol{i} \in X(\theta)} \omega_{\boldsymbol{i}} g'(\boldsymbol{x}_{\boldsymbol{i}}) f(g(x_{\boldsymbol{i}})) = \sum_{\boldsymbol{i} \in X(\theta)} \hat{\omega}_{\boldsymbol{i}} f(\hat{x}_{\boldsymbol{i}}).$$

The transformed quadrature nodes are $\hat{x}_{\boldsymbol{i}} = g(\boldsymbol{x}_{\boldsymbol{i}})$ and the corresponding quadrature weights are $\hat{\omega}_{\boldsymbol{i}} = \omega_{\boldsymbol{i}} g'(\boldsymbol{x}_{\boldsymbol{i}})$. Similarly, if $G_\theta[f \circ g](x) \approx f(g(x))$, then

$$f(x) = f(g(g^{-1}(x))) \approx G_\theta[f \circ g](g^{-1}(x)),$$

and the sparse grids nodes associated with $f(x)$ are again $\hat{x}_{\boldsymbol{i}} = g(\boldsymbol{x}_{\boldsymbol{i}})$. Note, in the interpolation case the function basis used to approximate $f(x)$ is a composition between the standard basis (polynomials or wavelets) and $g^{-1}(x)$.

Appropriate choice of $g(x)$ can significantly accelerate convergence, but a wrong choice can severely deteriorate accuracy. As an experimental feature, Tasmanian allows for non-linear transformation of the integration/interpolation domain with $g(x)$ based on the truncated Maclaurin series of $\arcsin(x)$. Different degree of truncation can be chosen in each direction and conformal mapping can be composed with standard linear a-b transformation to obtain optimal rule over any arbitrary domain. Note: this feature will not work with unbounded rules, such as Gauss-Laguerre and Gauss-Hermite.

2.12 Alternative coefficient construction

The sparse grids approximation can be generalized as

$$G_\Theta[f](\boldsymbol{x}) = \sum_{i=j}^{n} c_j \phi_j(\boldsymbol{x}),$$

where c_j is a set of scalar or vector coefficients depending whether $f(\boldsymbol{x})$ has scalar or vector output, and ϕ_j is a set basis functions. The approximation is related to the best fit of $f(\boldsymbol{x})$ in the span of $\phi_j(\boldsymbol{x})$ with a penalty constant (e.g., Lebesgue constant). Here, for simplicity, we suppress the multi-index notation and assume linear ordering of the nodes and basis functions. In the standard SG algorithms, the n coefficients c_j are derived from n samples of $f(\boldsymbol{x}_j)$ collected at specially chosen nodes $\boldsymbol{x}_j$. The choice of $\boldsymbol{x}_j$ is performed in a way that minimizes the penalty, but it also leads to a significant drawback, i.e., the target function $f(\boldsymbol{x})$ must be evaluated at exactly the selected set of nodes. In some applications, this is either impractical or even infeasible, e.g., the domain of $f(\boldsymbol{x})$ is not a hypercube but rather a blob of some shape contained within a hypercube. In order to utilize the flexible function spaces associated with sparse grids and in order to take advantage of the advanced adaptive approximation algorithms, a different approach is needed to construct c_j from an arbitrary set of realizations of $f(\boldsymbol{x})$.

Let $\{f(\boldsymbol{s}_i)\}_{i=1}^m$ indicate m samples of $f(\boldsymbol{x})$ for an arbitrary set of sample points $\boldsymbol{s}_i$, where for simplicity we assume that $f(\boldsymbol{x})$ is scalar valued. Define the basis matrix A and data vector f

$$A = \{a_{i,j}\} \in \mathbb{R}^{m \times n}, \quad \text{where} \quad a_{i,j} = \phi_j(\boldsymbol{s}_i), \qquad \boldsymbol{f} = \{f_i\}, \quad \text{where} \quad f_i = f(\boldsymbol{s}_i).$$

Similarly, we can arrange the coefficients c_j in a vector $\boldsymbol{c}$, and we seek $\boldsymbol{c}$ such that

$$A\boldsymbol{c} = \boldsymbol{f}. \tag{48}$$

In the case of standard sparse grids construction with a nested rule, (48) has exact solution, i.e., either $\boldsymbol{c} = \boldsymbol{f}$ for global grids or $\boldsymbol{c}$ are the hierarchical coefficients of the sequence or local grids. In the case of non-nested grids, the coefficients $\boldsymbol{c}$ have a more complex nature and (48) is not satisfied for all rows, but the "solution" $\boldsymbol{c}$ is found according to the direct sum of tensors formula. In the general case, when the samples come from an arbitrary set, an exact solution cannot be found and since $m \neq n$ the system of equations is either under or over determined.

3 Random Sampling

3.1 DREAM: General algorithm

Let $\Gamma \subset \mathbb{R}^d$ and $\rho : \Gamma \to \mathbb{R}^+$ be a non-negative function with

$$\int_\Gamma \rho(\boldsymbol{x}) d\boldsymbol{x} < \infty,$$

then scaling $\rho(\boldsymbol{x})$ gives us a probability density function and the goal of the random sampling algorithm is to generate points $\{\boldsymbol{x}_i\}$ with the said distributions.

Standard Metropolis-Hastings algorithm creates a chain of samples, by iteratively proposing a new sample followed by an accept/reject test. In short, given $\boldsymbol{x}_i$, we obtain a random perturbation $\boldsymbol{g}_i$ (with distribution symmetric around $\mathbf{0}$) and we set

$$\boldsymbol{x}_{i+1} = \begin{cases} \boldsymbol{x}_i + \boldsymbol{g}_i, & \frac{\rho(\boldsymbol{x}_i + \boldsymbol{g}_i)}{\rho(\boldsymbol{x}_i)} \geq u_i, \\ \boldsymbol{x}_i & \text{othwerwise,} \end{cases}$$

where u_i is a random sample from uniform distribution over $[0, 1]$. Regardless of the initial $\boldsymbol{x}_0$, in the limit as $i \to \infty$, the distribution of $\boldsymbol{x}_i$ matches the one defined by the pdf $\rho(\boldsymbol{x})$. In practice, a finite set of $\boldsymbol{x}_i$ are computed and an initial batch of samples is discarded (a process called the burn-up).

The DiffeRential Evolution Adaptive Metropolis (DREAM) algorithm simultaneously evolves a number of chains and the probability distribution for the correction is informed by all samples in the chain. Specifically, the chain state is

$$\{\boldsymbol{x}_{1,i},\, \boldsymbol{x}_{2,i},\, \ldots,\, \boldsymbol{x}_{C,i}\},$$

where C is the total number of chains. Each chain is updated according to the accept/reject criteria

$$\boldsymbol{x}_{c,i+1} = \begin{cases} \boldsymbol{x}_{c,i} + \boldsymbol{g}_{c,i}, & \frac{\rho(\boldsymbol{x}_i + \boldsymbol{g}_{c,i})}{\rho(\boldsymbol{x}_i)} \geq u_i, \\ \boldsymbol{x}_{c,i} & \text{othwerwise.} \end{cases} \tag{49}$$

The updates are chosen as

$$\boldsymbol{g}_{c,i} = \gamma(\boldsymbol{x}_{c_1,i} - \boldsymbol{x}_{c_2,i}) + \boldsymbol{r}_{c,i}, \tag{50}$$

where c_1 and c_2 are random integers in the range $[1, C]$, γ is a jump scale constant (usually in $[0, 1]$), and $\boldsymbol{r}_{c,i}$ is a small correction sampled from a distribution that is symmetric around $\mathbf{0}$. The on-line manual refers to γ and $\boldsymbol{r}_{c,i}$ as the *differential* and *independent* updates.

Compared to the standard Metropolis-Hastings method, the DREAM algorithm has several practical advantages

- Using a large number of chains allows better initial coverage of Γ, which limits the dependence on the initial guess.
- The proposal is constantly updated based on the current chain state, which accelerates convergence.
- In the single chain algorithm, once the state reaches a high-probability regions, it is very unlikely that the chain would jump out of that region and reach a second one. Thus, Metropolis-Hastings struggles when dealing with multi-modal distributions. In contrast, when DREAM uses a sufficiently large number of chains some chains will reach every high probability region.

Distribution	Domain	Density	Defining parameters
Uniform	$[a, b]$	$\frac{1}{b-a}$	a, b
Gaussian	$(-\infty, \infty)$	$\frac{1}{\sqrt{2\pi\sigma}} \exp\left(-\frac{1}{2\sigma}(x-\mu)^2\right)$	σ, μ
Truncated Gaussian	$[a, b]$	$\frac{\exp\left(-\frac{1}{2\sigma}(x-\mu)^2\right)}{\tilde{C}}$	σ, μ, a, b
Exponential	$[a, \infty)$	$\lambda \exp\left(-\lambda(x-a)\right)$	λ, a
Beta	$[a, b]$	$\frac{\Gamma(\alpha+\beta)}{\Gamma(\alpha)\Gamma(\beta)} \frac{(x-a)^{\alpha-1}(b-x)^{\beta-1}}{(b-a)^{\alpha+\beta-2}}$	a, b, α, β
Gamma	$[a, \infty)$	$\frac{\beta^\alpha}{\Gamma(\alpha)} x^{\alpha-1} \exp\left(-\beta(x-a)\right)$	a, α, β

Table 4. Probability distributions included in Tasmanian. The normalization constant for the Truncated Gaussian distribution is $\tilde{C} = \left(\sqrt{2\pi\sigma} - \int_{(-\infty,a)\cup(b,\infty)} \exp\left(-\frac{1}{2\sigma}(\xi-\mu)^2\right) d\xi\right)$, and $\Gamma(\alpha)$ us the gamma funciton.

- Metropolis-Hastings can handle multiple modes if the high probability region of the update distribution is sufficiently large; however, this is seldom practical as wide spread of the updates leads to very low acceptance rate, which in turn leads to poor mixing.* DREAM largely circumvents this limitation and can handle multiple modes without sacrificing acceptance rate.
- Evolving multiple chains simultaneously allows the use of batched evaluations of $\rho(\boldsymbol{x})$, which can be performed much more efficiently than sequential evaluations.

Note that setting $\gamma = 0$ reduces the DREAM algorithm to multiple independent chains of standard Metropolis-Hastings.

3.2 Supported probability distributions

Tasmanian includes 6 probability distributions that can be used as priors in a context of Bayesian inference (e.g., see 1.2). The pdfs and the associated parameters are listed in Table 4. In addition, Tasmanian implements Gaussian likelihood of form (5), where the covariance could be diagonal with constant or non-constant diagonal entries, or a general dense matrix.

*Mixing is a numerical phenomena where multiple iterates of the same chain have identical values, which is not desirable when the chains are used for statistical analysis.

References

[1] S. ACHARJEE AND N. ZABARAS, *A non-intrusive stochastic galerkin approach for modeling uncertainty propagation in deformation processes*, Computers & structures, 85 (2007), pp. 244–254. 2

[2] B. ADCOCK AND R. B. PLATTE, *A mapped polynomial method for high-accuracy approximations on arbitrary grids*, SIAM Journal on Numerical Analysis, 54 (2016), pp. 2256–2281. 23

[3] N. AGARWAL AND N. R. ALURU, *A domain adaptive stochastic collocation approach for analysis of mems under uncertainties*, Journal of Computational Physics, 228 (2009), pp. 7662–7688. 2

[4] V. BARTHELMANN, E. NOVAK, AND K. RITTER, *High dimensional polynomial interpolation on sparse grids*, Advances in Computational Mathematics, 12 (2000), pp. 273–288. 2

[5] J. BECK, F. NOBILE, L. TAMELLINI, AND R. TEMPONE, *Convergence of quasi-optimal stochastic galerkin methods for a class of pdes with random coefficients*, Computers & Mathematics with Applications, 67 (2014), pp. 732–751. 8

[6] G. E. BOX AND G. C. TIAO, *Bayesian inference in statistical analysis*, vol. 40, John Wiley & Sons, 2011. 4

[7] H.-J. BUNGARTZ AND M. GRIEBEL, *Sparse grids*, Acta Numer., 13 (2004), pp. 147–269. 2

[8] M. A. CHKIFA, *On the Lebesgue constant of Leja sequences for the complex unit disk and of their real projection*, Journal of Approximation Theory, 166 (2013), pp. 176–200. 10, 11

[9] ——, *On the lebesgue constant of a new type of $\mathcal{R}$-leja sequences*, tech. rep., ORNL/TM-2015/657, Oak Ridge National Laboratory., 2015. 11

[10] C. W. CLENSHAW AND A. R. CURTIS, *A method for numerical integration on an automatic computer*, Numerische Mathematik, 2 (1960), pp. 197–205. 10

[11] S. DE MARCHI, *On Leja sequences: some results and applications*, Applied Mathematics and Computation, 152 (2004), pp. 621–647. 11

[12] M. D'ELIA, E. PHIPPS, A. RUSHDI, AND M. EBEIDA, *Surrogate-based ensemble grouping strategies for embedded sampling-based uncertainty quantification*, arXiv preprint arXiv:1705.02003, (2017). 2

[13] M. ELDRED, C. WEBSTER, AND P. CONSTANTINE, *Evaluation of non-intrusive approaches for wiener-askey generalized polynomial chaos*, in Proceedings of the 10th AIAA Non-Deterministic Approaches Conference, number AIAA-2008-1892, Schaumburg, IL, vol. 117, 2008, p. 189. 2

[14] L. FEJÉR, *On the infinite sequences arising in the theories of harmonic analysis, of interpolation, and of mechanical quadratures*, Bulletin of the American Mathematical Society, 39 (1933), pp. 521–534. 10

[15] D. GALINDO, P. JANTSCH, C. G. WEBSTER, AND G. ZHANG, *Accelerating stochastic collocation methods for partial differential equations with random input data*, SIAM/ASA Journal on Uncertainty Quantification, 4 (2016), pp. 1111–1137. 2

[16] D. GAMERMAN AND H. F. LOPES, *Markov chain Monte Carlo: stochastic simulation for Bayesian inference*, CRC Press, 2006. 4

[17] T. GERSTNER AND M. GRIEBEL, *Numerical integration using sparse grids*, Numerical algorithms, 18 (1998), pp. 209–232. 2

[18] ——, *Dimension–adaptive tensor–product quadrature*, Computing, 71 (2003), pp. 65–87. 2

[19] M. GRIEBEL, *Adaptive sparse grid multilevel methods for elliptic pdes based on finite differences*, Computing, 61 (1998), pp. 151–179. 2, 14, 21

[20] M. GRIEBEL AND J. HAMAEKERS, *Fast Discrete Fourier Transform on Generalized Sparse Grids*, Lecture Notes in Computational Science and Engineering, Springer International Publishing Switzerland, 2014, ch. 4, pp. 75–107. 13

[21] M. GUNZBURGER, C. TRENCHEA, AND C. WEBSTER, *A generalized stochastic collocation approach to constrained optimization for random data identification problems*, Tech. Rep. ORNL/TM-2012/185, Oak Ridge National Laboratory, 2012. 2

[22] M. GUNZBURGER, C. WEBSTER, AND G. ZHANG, *An adaptive wavelet stochastic collocation method for irregular solutions of partial differential equations with random input data*, Tech. Rep. ORNL/TM-2012/186, Oak Ridge National Laboratory, 2012. 2, 21

[23] M. D. GUNZBURGER, C. G. WEBSTER, AND G. ZHANG, *Stochastic finite element methods for partial differential equations with random input data*, Acta Numer., 23 (2014), pp. 521–650. 2

[24] K. HALLATSCHEK, *Fouriertransformation auf dünnen gittern mit hierarchischen basen*, Numerische Mathematik, 63 (1992), pp. 83–97. 13

[25] J. D. JAKEMAN, R. ARCHIBALD, AND D. XIU, *Characterization of discontinuities in high-dimensional stochastic problems on adaptive sparse grids*, J. Comput. Phys., 230 (2011), pp. 3977–3997. 2

[26] J. D. JAKEMAN, A. NARAYAN, AND D. XIU, *Minimal multi-element stochastic collocation for uncertainty quantification of discontinuous functions*, J. Comput. Phys., 242 (2013), pp. 790–808. 2

[27] J. D. JAKEMAN AND S. G. ROBERTS, *Local and dimension adaptive stochastic collocation for uncertainty quantification*, in Sparse Grids and Applications, Springer, 2012, pp. 181–203. 2

[28] P. JANTSCH AND C. G. WEBSTER, *Sparse grid quadrature rules based on conformal mappings*, Sparse Grids and Applications, - (to appear), pp. –. 23

[29] V. KHAKHUTSKYY AND M. HEGLAND, *Spatially-dimension-adaptive sparse grids for online learning*, in Sparse Grids and Applications – Stuttgart 2014, Springer, 2016, pp. 133–162. 2

[30] A. KLIMKE AND B. WOHLMUTH, *Algorithm 847: Spinterp: piecewise multilinear hierarchical sparse grid interpolation in matlab*, ACM Transactions on Mathematical Software (TOMS), 31 (2005), pp. 561–579. 2, 14

[31] X. MA AND N. ZABARAS, *An adaptive hierarchical sparse grid collocation algorithm for the solution of stochastic differential equations*, Journal of Computational Physics, 228 (2009), pp. 3084–3113. 2, 14

[32] Z. MORROW AND M. STOYANOV, *A method for dimensionally adaptive sparse trigonometric interpolation of periodic functions*, arXiv preprint arXiv:1908.10672, (2019). 14

[33] A. NARAYAN AND J. D. JAKEMAN, *Adaptive Leja sparse grid constructions for stochastic collocation and high-dimensional approximation*, SIAM J. Sci. Comput., 36 (2014), pp. A2952–A2983. 2

[34] F. NOBILE, L. TAMELLINI, AND R. TEMPONE, *Convergence of quasi-optimal sparse-grid approximation of Hilbert-space-valued functions: application to random elliptic PDEs*, Numer. Math., 134 (2016), pp. 343–388. 2

[35] F. NOBILE, R. TEMPONE, AND C. G. WEBSTER, *An anisotropic sparse grid stochastic collocation method for partial differential equations with random input data*, SIAM Journal on Numerical Analysis, 46 (2008), pp. 2411–2442. 2

[36] F. NOBILE, R. TEMPONE, AND C. G. WEBSTER, *A sparse grid stochastic collocation method for partial differential equations with random input data*, SIAM Journal on Numerical Analysis, 46 (2008), pp. 2309–2345. 2

[37] T. C. PATTERSON, *The optimum addition of points to quadrature formulae*, Mathematics of Computation, 22 (1968), pp. 847–856. 11

[38] D. PFLÜGER, *Spatially adaptive refinement*, in Sparse Grids and Applications, Springer, 2012, pp. 243–262. 2

[39] D. PFLÜGER, B. PEHERSTORFER, AND H.-J. BUNGARTZ, *Spatially adaptive sparse grids for high-dimensional data-driven problems*, J. Complexity, 26 (2010), pp. 508–522. 2

[40] E. PHIPPS, M. D'ELIA, H. C. EDWARDS, M. HOEMMEN, J. HU, AND S. RAJAMANICKAM, *Embedded ensemble propagation for improving performance, portability, and scalability of uncertainty quantification on emerging computational architectures*, SIAM Journal on Scientific Computing, 39 (2017), pp. C162–C193. 2

[41] S. A. SMOLYAK, *Quadrature and interpolation formulas for tensor products of certain classes of functions*, Dokl. Akad. Nauk SSSR, 4 (1963), pp. 240–243 (English translation). 2

[42] F. SPRENGEL, *A class of periodic function spaces and interpolation on sparse grids*, Numerical Functional Analysis and Optimization, 21 (2000), pp. 273–293. 13

[43] M. STOYANOV, *Hierarchy-direction selective approach for locally adaptive sparse grids*, tech. rep., ORNL/TM-2013/384, Oak Ridge National Laboratory., 2013. 2, 3, 14, 17

[44] M. STOYANOV, *Adaptive sparse grid construction in a context of local anisotropy and multiple hierarchical parents*, in Sparse Grids and Applications-Miami 2016, Springer, 2018, pp. 175–199. 2, 20

[45] M. STOYANOV, P. SELESON, AND C. WEBSTER, *Predicting fracture patterns in simulations of brittle materials under variable load and material strength*, in 19th AIAA Non-Deterministic Approaches Conference, 2017, p. 1326. 2

[46] M. K. STOYANOV AND C. G. WEBSTER, *A dynamically adaptive sparse grid method for quasi-optimal interpolation of multidimensional analytic functions*, arXiv preprint arXiv:1508.01125, (2015). 2, 3, 7, 8, 10, 11, 12

[47] ——, *A dynamically adaptive sparse grids method for quasi-optimal interpolation of multidimensional functions*, Computers & Mathematics with Applications, 71 (2016), pp. 2449–2465. 2

[48] W. SWELDENS AND P. SCHRÖDER, *Building your own wavelets at home*, in Wavelets in the Geosciences, Springer, 2000, pp. 72–107. 21

[49] J. A. VRUGT, C. TER BRAAK, C. DIKS, B. A. ROBINSON, J. M. HYMAN, AND D. HIGDON, *Accelerating markov chain monte carlo simulation by differential evolution with self-adaptive randomized subspace sampling*, International Journal of Nonlinear Sciences and Numerical Simulation, 10 (2009), pp. 273–290. 4

[50] J. A. VRUGT, C. J. TER BRAAK, M. P. CLARK, J. M. HYMAN, AND B. A. ROBINSON, *Treatment of input uncertainty in hydrologic modeling: Doing hydrology backward with markov chain monte carlo simulation*, Water Resources Research, 44 (2008). 4

[51] G. ZHANG AND M. GUNZBURGER, *Error analysis of a stochastic collocation method for parabolic partial differential equations with random input data*, SIAM Journal on Numerical Analysis, 50 (2012), pp. 1922–1940. 2

[52] G. ZHANG, M. GUNZBURGER, AND W. ZHAO, *A sparse grid method for multi-dimensional backward stochastic differential equaitons*, Journal of Computational Mathematics, 31 (2013), pp. 221–248. 2

[53] G. ZHANG, D. LU, M. YE, M. GUNZBURGER, AND C. WEBSTER, *An adaptive sparse-grid high-order stochastic collocation method for bayesian inference in groundwater reactive transport modeling*, Water Resources Research, 49 (2013), pp. 6871–6892. 4