EFFICIENT BASIS CHANGE AND REGULARIZATION FOR SPARSE-GRID INTERPOLATING POLYNOMIALS

GREGERY T. BUZZARD

ABSTRACT. Sparse-grid interpolation provides good approximations to smooth functions in high dimensions based on relatively few function evaluations, but in standard form is expressed in Lagrange polynomials and requires function values at all points of a sparse grid. Here we give a block-diagonal factorization of the matrix for changing basis from a Lagrange polynomial formulation of a sparse-grid interpolant to a tensored orthogonal polynomial (or gPC) representation. For fixed maximum degree of interpolation, the resulting change of basis algorithm is linear in the number of points of evaluation as dimension increases. Additionally, we use this factorization with ℓ_1 and minimum Sobolev norm (MSN) regularization to provide good interpolants even when function values are not available at a significant fraction of points of the sparse grid or are subject to measurment error.

KEYWORDS: Sparse grid, polynomial interpolation, stochastic collocation, polynomial chaos, sparsity, ℓ_1 minimization.

1. INTRODUCTION

A common problem in many areas of computational mathematics is to approximate a given function based on a small number of functional evaluations or observations. This problem arises in numerical methods for PDE [29, 33], sensitivity analysis [28, 19, 9], uncertainty quantification [33], many areas of modeling [20, 26], and other settings. As a result, there are a large number of approaches to this problem, and the literature is large and growing quickly. In settings in which the points of evaluation may be chosen at will, two common approaches are generalized polynomial chaos (gPC) using cubature, and sparse grid collocation. In other settings in which the points of evaluation are given, common approaches include RS-HDMR, cut-HDMR, ANOVA decomposition, kriging, and moving least squares,

Sparse grid collocation has been used widely in recent years as a means of providing a reasonable approximation to a smooth function, f, defined on a hypercube in $\mathbb{R}^n$, based on relatively few function evaluations [33]. This method produces a polynomial interpolant using Lagrange interpolating polynomials based on function values at points in a union of product grids of small dimension [3, 30]. Using barycentric interpolation to evaluate the resulting polynomial [5], this method is a viable alternative to an

Date: September 8, 2011.

expansion of f in terms of a sum of products of one-dimensional orthogonal polynomials. This latter approach is known as generalized polynomial chaos (gPC) or spectral decomposition, and is obtained via standard weighted L^2 techniques. However, the orthogonality implicit in the gPC representation often provides many advantages over the Lagrange representation, particularly in applications to differential equations, in which the gPC representation is closely related to spectral methods. Other advantages of the gPC representation include the ability to estimate convergence as more points are added to the sparse grid, and the ability to estimate variance-based sensitivity coefficients quickly and accurately [19]. A common approach to obtain the gPC coefficients is to use numerical integration by applying a cubature rule. However, cubature rules to integrate the product of two polynomials up to the degree in the sparse grid interpolant typically require either more or different points than found in the grid itself.

In this paper we provide an efficient algorithm for converting from the Lagrange interpolating polynomial to an equivalent gPC polynomial using only the function values at the sparse grid points. The foundation of this algorithm is a matrix factorization based on the fact that the sparse grid is a union of small product grids. More precisely, let Φ be the matrix obtained by evaluating each of the gPC basis functions (one per column) at each of the sparse grid points (one per row). This matrix produces the gPC coefficients, c, from the function values, f, by solving $\Phi c = f$ for c. We show below that Φ^{-1} factors into a product of block diagonal matrices in which each block corresponds to one of the small product grids composing the full sparse grid. We show also how to adapt this factorization to apply ideas of ℓ_1 minimization and minimum Sobolev norm (MSN) to approximate these coefficients when a significant fraction of function values are missing or when the function values are corrupted by noise.

Methods for changing basis between sets of orthogonal polynomials are described in many places, including [1], [6], and [22]. Most of these results focus on the case of polynomials of one variable. As will be seen below, tensored versions of some of these methods could be applied in place of the matrix multiplication method described in section 4. However, as seen below, the use of such methods does not change the basic result on linear running time for fixed accuracy, and in practice they are not likely to provide a significant computational advantage over the basic methods described in Section 4 or Section 5. For further background, Boyd [8], Chapter 5 discusses the Matrix Multiplication Transform for converting between Lagrange and spectral representations, while Chapter 10 discusses the special case of the Chebyshev polynomial expansion using the FFT. A recent result of Carley [15] provides a method for interpolating a function on an arbitrary set of points using a specially constructed basis of orthogonal polynomials and for differentiating the resulting interpolating polynomial. However, little is known about the convergence of this method as a function of the number of points.

Contributions and organization: The primary contributions of this paper are (i) the block-diagonal factorization of the basis-change matrix Φ^{-1}, (ii) the resulting change-of-basis algorithm (which, for a fixed order of polynomial accuracy and increasing dimension, has run-time linear in the number of points), and (iii) the resulting algorithm for ℓ_1 and MSN regularization (which, under appropriate assumptions, has run-time linear in the total number of basis polynomials), using either exact interpolation or ℓ_2-approximation of function values when these values have a noise component.

Additionally, we note that numerical results show that the running time per point of evaluation is essentially constant up to a sparse grid depth (closely related to polynomial degree of accuracy) of about 8, that this algorithm may be applied to any set of orthogonal polynomials, including those whose interval of orthogonality is infinite or semi-infinite. Finally, the code developed here for evaluating the resulting gPC representation is typically a factor of 5 or more faster than the widely available spinterp package [25].

In Section 2, we provide some background into Smolyak's algorithm for sparse grid interpolation and then give the factorization of Φ^{-1} in Section 3. In section 4 we describe the algorithm to produce the gPC representation based on function values at the sparse grid points only and give an upper bound on the complexity of the algorithm. In section 5, we discuss variations of the basic algorithm in which ℓ_1 minimization and MSN regularization are used to produce reasonable interpolations even when not all function values are available. In section 6, we give numerical results on running time and accuracy of the algorithm.

This research is partially supported under NSF grant DMS-0900277. I am grateful to Brad Lucier and Dongbin Xiu for many helpful discussions.

2. Sparse grid interpolation

In this section we provide some background on Smolyak's algorithm for sparse grid interpolation. The discussion here is based on [3]. The foundation for sparse grid interpolation is interpolation in one dimension using Lagrange interpolation:

$$U^i(f) = \sum_{j=1}^{m_i} f(x_j^i) L_j^i, \tag{1}$$

where $i \in \mathbb{N}$, $x_j^i \in [-1, 1]$, and L_j^i is the Lagrange polynomial satisfying $L_j^i(x_k^i) = \delta_{jk}$. A common choice for sparse grid interpolation is to use the Chebyshev-Gauss-Lobatto (CGL) points, in which case $m_i = 2^{i-1} + 1$ and $x_j^i = -\cos((j-1)\pi/2^{i-1})$ for $i > 1$, since this choice provides a nesting property that is vital for efficient interpolation in higher dimensions. Moreover, $m_1 = 1$ and $x_1^1 = 0$ for this choice. Other choices, including Gauss-Patterson nodes and nodes on a (semi)-infinte interval as in [7], are also possible; the methods below apply unchanged for these choices. These

one dimensional formulas may be tensored in dimension $d > 1$ to yield

$$U^{\mathbf{i}}(f) = (U^{i_1} \otimes \cdots \otimes U^{i_d})(f) = \sum_{\mathbf{1} \le \mathbf{j} \le m_{\mathbf{i}}} f(x^{\mathbf{i}}_{\mathbf{j}}) L^{\mathbf{i}}_{\mathbf{j}}, \tag{2}$$

where $\mathbf{i}$ and $\mathbf{j}$ are multi-indices with componentwise partial order, $\mathbf{1}$ is the multi-index of all 1s, $x^{\mathbf{i}}_{\mathbf{j}} = (x^{i_1}_{j_1}, \ldots, x^{i_d}_{j_d})$, and $L^{\mathbf{i}}_{\mathbf{j}}(x) = L^{i_1}_{j_1}(x_1) \cdots L^{i_d}_{j_d}(x_d)$. This formula requires $m_{i_1} \cdots m_{i_d}$ function values sampled on a product grid. Note however, that when some of the i_k are 1, then this grid is of dimension less than d (since $m_1 = 1$).

Linear combinations of these formulas produce the Smolyak formulas. Let $U^0 = 0$ and $\Delta^i = U^i - U^{i-1}$ for $i \in \mathbb{N}$, and define $|\mathbf{i}| = i_1 + \cdots + i_d$. Then for $q \ge d$ we have

$$A(q, d) = \sum_{d \le |\mathbf{i}| \le q} \Delta^{\mathbf{i}}, \tag{3}$$

where $\Delta^{\mathbf{i}}$ is the tensor product of the Δ^{i_k}. A multinomial expansion and $q = d + k$ produces

$$A(d+k, d) = \sum_{k+1 \le |\mathbf{i}| \le d+k} (-1)^{d+k-|\mathbf{i}|} \begin{pmatrix} d-1 \\ |\mathbf{i}| - k - 1 \end{pmatrix} U^{\mathbf{i}}. \tag{4}$$

When $d > k$, as is common for large d, we may replace $k + 1 \le |\mathbf{i}|$ by $d \le |\mathbf{i}|$. An anisotropic version of this formula is described in [24]. The formula for this version is largely the same as (3) with the index set $d \le |\mathbf{i}| \le q$ replaced by a more general index set I, characterized by the property that if $\mathbf{i} \in I$ and $i_k > 1$, then $\mathbf{i} - \mathbf{e}_k \in I$, where $\mathbf{e}_k$ has 1 in the kth position and zeros elsewhere.

Important results from [3] include that $A(d + k, d)(P) = P$ for all polynomials of degree at most k, and that if f is in C^m, then

$$\|f - A(q, d)(f)\|_\infty \le \frac{c_{d,m}}{n^m} (\log n)^{(m+2)(d-1)+1} \|f\|_\infty, \tag{5}$$

where $n = n(q, d)$ is the number of points in the sparse grid, $X(q, d)$, for $A(q, d)$. There is a similar error estimate for a weighted L^2 norm of the Chebyshev expansion of $f - A(q, d)(f)$. They note also that $n(d + k, d)$ is asymptotic to $2^k d^k / k!$ as d tends to ∞, in the sense that the ratio tends to 1.

3. Change of basis matrix and factorization

In this section we describe the matrix factorization that is central to the efficient change to a gPC basis.

Let P_j, $j \ge 0$ be a set of polynomials, each of degree j, orthogonal on the interval $[-1, 1]$ using inner product $\int_{-1}^{1} fg\, w dx$, where w is a nonnegative weight function that is positive in the interior of the interval (straightforward modifications allow for unbounded intervals, but for clarity we restrict to the finite interval).

The expansion in (4) together with (1) gives a decomposition of $A(q,d)$ into a sum of products of Lagrange polynomials, with each $U^{\mathbf{i}}$ corresponding to a full product grid, $X_{\mathbf{i}}$ (although usually $X_{\mathbf{i}}$ is less than d-dimensional since each entry of $\mathbf{i}$ that is 1 produces only a single point for the corresponding dimension). On this grid, each nontrivial coordinate direction k has $i = i_k > 1$, and the number of points in this direction is $m_i = 2^{i-1} + 1$. Hence the degree of each L_j^i is $m_i - 1$, so we write each L_j^i as a linear combination of $P_0, \ldots, P_{m_i-1}$. Suppressing for the moment the dependence on $\mathbf{i}$, we have

$$L_j = \sum_{m=0}^{m_i-1} C_{j,m} P_m. \tag{6}$$

Taking tensor products as in (1), we obtain a basis in terms of tensor products of P_n for interpolation on the product grid $X_{\mathbf{i}}$. Taking a union over the $X_{\mathbf{i}}$ that appear in (4), we obtain a basis for interpolation on the entire sparse grid. The problem then is to determine the (gPC) coefficients in this basis for specified function values, $\mathbf{f}$, at the sparse grid nodes.

Conceptually, the simplest approach to finding the gPC coefficients starts by evaluating each basis function, P, at each point in $X = X(q,d)$. These values form a matrix, Φ, with each basis function corresponding to one column and each point in X corresponding to one row.

Definition 3.1. Let P_j, $j = 1, \ldots, n$ be the gPC basis polynomials for the sparse grid, X, having points $x_1, \ldots, x_n$. Then the matrix Φ is the matrix with entries $\Phi_{i,j} = P_j(x_i)$.

With this definition, and a gPC coefficient vector, c, the product Φc gives the value at each point in X of the polynomial determined by the coefficients in c. Hence one can solve the interpolation problem by solving $\Phi c = \mathbf{f}$ for c. Of course, simply constructing this matrix takes $O(n^2)$ operations, while solving the system in this form using elimination takes $O(n^3)$ operations.

The alternative presented here is to give a factorization of Φ^{-1} based on (4).

Theorem 3.2. *The change-of-basis matrix Φ^{-1} has a factorization*

$$\Phi^{-1} = \pi^T \hat{D}_w \hat{\Phi}^{-1} \pi,$$

where $\hat{\Phi}$ is block-diagonal, with each block corresponding to a sub-grid, $X_{\mathbf{i}}$; $\hat{D}_w$ is diagonal; and π is a matrix that includes $\mathbb{R}^n$ into $\mathbb{R}^N$ for some $N > n$.

Proof. To start, we order the columns of Φ so that for each multi-index, $\mathbf{i}$, the set of column indices of basis functions for interpolation on $X_{\mathbf{i}}$ from (6) are the same as the set of row indices for points in $X_{\mathbf{i}}$. Let n be the number of points in X and $n_{\mathbf{i}}$ be the number in $X_{\mathbf{i}}$. Let $\pi_{\mathbf{i}}$ be the projection from $\mathbb{R}^n$ to $\mathbb{R}^{n_{\mathbf{i}}}$ obtained by restricting to the indices corresponding to $X_{\mathbf{i}}$. In this case, $\Phi_{\mathbf{i}} = \pi_{\mathbf{i}} \Phi \pi_{\mathbf{i}}^T$ is the matrix obtained by evaluating basis functions at points, with the basis functions and points restricted to those associated

with $X_{\mathbf{i}}$. Hence, given $\mathbf{f} = f(X)$, we can interpolate f on $X_{\mathbf{i}}$ by solving $\Phi_{\mathbf{i}} c_{\mathbf{i}} = \pi_{\mathbf{i}} \mathbf{f}$ for $c_{\mathbf{i}}$.

Since $U^{\mathbf{i}}(f)$ in (1) also interpolates f on $X_{\mathbf{i}}$, we have

$$U^{\mathbf{i}}(f)(X_{\mathbf{i}}) = \Phi_{\mathbf{i}} c_{\mathbf{i}} = \pi_{\mathbf{i}} \Phi \pi_{\mathbf{i}}^T c_{\mathbf{i}}. \tag{7}$$

Since the gPC basis functions associated with $X_{\mathbf{i}}$ were obtained by changing basis from the Lagrange representation of $U^{\mathbf{i}}$, we see that $U^{\mathbf{i}}(f)$ and the polynomial with gPC coefficients $\pi_{\mathbf{i}}^T c_{\mathbf{i}}$ are the same polynomial. Hence we may extend the equality in (7) to all of X by replacing $X_{\mathbf{i}}$ by X on the left and dropping $\pi_{\mathbf{i}}$ on the right to obtain

$$U^{\mathbf{i}}(f)(X) = \Phi \pi_{\mathbf{i}}^T c_{\mathbf{i}}. \tag{8}$$

Let $w_{\mathbf{i}}$ denote the weight for $U^{\mathbf{i}}$ in (4). Using (8) with (4), we have

$$\begin{aligned} \mathbf{f} = A(f)(X) &= \sum_{\mathbf{i} \in I} w_{\mathbf{i}} U^{\mathbf{i}}(f)(X) \\ &= \sum_{\mathbf{i}} w_{\mathbf{i}} \Phi \pi_{\mathbf{i}}^T c_{\mathbf{i}} \\ &= \Phi \left(\sum_{\mathbf{i}} w_{\mathbf{i}} \pi_{\mathbf{i}}^T c_{\mathbf{i}} \right). \end{aligned}$$

Recalling that $c_{\mathbf{i}} = \Phi_{\mathbf{i}}^{-1} \pi_{\mathbf{i}} \mathbf{f}$, we have

$$\Phi^{-1} \mathbf{f} = \sum_{\mathbf{i} \in I} w_{\mathbf{i}} \pi_{\mathbf{i}}^T \Phi_{\mathbf{i}}^{-1} \pi_{\mathbf{i}} \mathbf{f}. \tag{9}$$

Let $\mathbf{i}_1, \ldots, \mathbf{i}_m$ be an enumeration of the indices in I, let π be the matrix obtained by vertical concatenation of $\pi_{\mathbf{i}_1}, \ldots, \pi_{\mathbf{i}_m}$, let $\hat{\Phi}$ be the block diagonal matrix with $\Phi_{\mathbf{i}_1}, \ldots, \Phi_{\mathbf{i}_m}$ as blocks, and let $\hat{D}_w$ be the diagonal matrix with diagonal entries $w_{\mathbf{i}_1}, \ldots, w_{\mathbf{i}_m}$, each repeated according to the size of the corresponding block. Then (9) gives the factorization

$$\Phi^{-1} = \pi^T \hat{D}_w \hat{\Phi}^{-1} \pi. \tag{10}$$

□

4. Algorithm and complexity

Based on the previous section, the algorithm to find the coefficient vector, c, to represent the sparse-grid interpolating polynomial in gPC basis is simply to use (10) to find $c = \Phi^{-1} \mathbf{f}$. We avoid the matrix inverse by defining $\hat{\mathbf{f}} = \pi \mathbf{f}$, solving $\hat{\Phi} \hat{c} = \hat{\mathbf{f}}$ for $\hat{c}$, and then using $c = \pi^T \hat{D}_w \hat{c}$.

In this section we show that for fixed k in (4) and large dimension, d, the complexity of this algorithm is linear in the number of points of evaluation. We frame this result as a corollary to the following proposition, which gives a bound on the total number of operations needed to compute on each subgrid, $X_{\mathbf{i}}$, in turn, as needed to solve $\hat{\Phi} \hat{c} = \hat{\mathbf{f}}$ for $\hat{c}$.

In both of the following results, $n(d+k,d)$ is the number of points in the sparse grid associated with $A(d+k,d)$.

Proposition 4.1. *Let $m \geq 1$ and $k \geq 1$. Then there is $c_{k,m} > 0$ so that for all $d > k$*

$$\sum_{d \leq |\mathbf{i}| \leq d+k} |X_{\mathbf{i}}|^m \leq c_{k,m} n(d+k,d).$$

In this proposition, we use the fact mentioned above that when $d > k$, then the sparse grid X is a union of $X_{\mathbf{i}}$ over $\mathbf{i}$ as indicated in the sum given here. Given this result, the following theorem is nearly immediate.

Theorem 4.2. *For fixed k, there is $c_k > 0$ so that for $d > k$, the coefficients in the gPC expansion of $A(d+k,d)(f)$ may be found using (10) with computation time bounded by $c_k n(d+k,d)$. That is, for fixed k, the running time is linear in the number of grid points.*

We have not attempted to calculate the best possible $c_{m,k}k$ in the proof of this result. In fact, numerical results given below show that in some cases, the coefficient c_k actually decreases as k increases due to the spread of fixed overhead costs over a small number of points when k and d are small. Additionally, the numerical results show that the time per point evaluated is roughly constant through $k = 8$.

Note that this algorithm computes the gPC coefficients of the interpolating polynomial rather than the original function itself. However, for a C^m function, f, with $m \geq 1$, the error estimate from (5) implies that the interpolating function converges uniformly to f as the depth, k, increases. Since the gPC coefficients are obtained by weighted integration of f against the orthogonal polynomials, the gPC coefficients for the interpolating polynomials converge to the gPC coefficients for f, and (5) provides a means to estimate the error in these coefficients based on the set of orthogonal polynomials and their corresponding weights.

Proof of Proposition 4.1. Given $|\mathbf{i}| = d+\kappa$ between d and $d+k$, consider the number of points in the corresponding grid $X_{\mathbf{i}}$. Let $r = r(\mathbf{i})$ be the number of nontrivial entries in $\mathbf{i}$ (that is, entries larger than 1). Suppose without loss that the first r entries of $\mathbf{i}$ are nontrivial with entries $i_j = \kappa_j + 1 \geq 2$. Then the number of points in this grid is $\prod_{j=1}^{r}(2^{\kappa_j}+1)$. If κ_1 is the only entry larger than 1, then $\kappa_1 = \kappa - (r-1)$, so the number of points is exactly $3^{r-1}(2^{\kappa-r+1}+1)$. A simple counting argument implies that any other distribution of nontrivial entries produces no more grid points, so

$$|X_{\mathbf{i}}| \leq 3^{r-1}2^{\kappa-r+2}.$$

To construct $\mathbf{i}$ with $|\mathbf{i}| = d+\kappa$ and r nontrivial entries, we select r out of d nontrivial entries, then distribute κ elements to these entries, so that each entry gets at least one element. There are $\binom{d}{r}\binom{\kappa-1}{r-1}$ ways to do this.

Summing from $r = 1$ to κ, and using some crude upper bounds, we have

$$\sum_{|\mathbf{i}|=d+\kappa} |X_{\mathbf{i}}|^m \leq \sum_{r=1}^{\kappa} \binom{d}{r}\binom{\kappa-1}{r-1}(3^m)^{r-1}(2^m)^{\kappa-r+2}$$
$$\leq (2^m)^{\kappa+1} d \sum_{q=0}^{\kappa-1} \binom{\kappa-1}{q} \frac{(3^m)^q d^q}{(2^m)^q q!}.$$

The summation on the right hand side is the Laguerre polynomial of degree $\kappa - 1$ evaluated at $-(3^m)d/(2^m)$. For κ and m fixed, this is asymptotic to $(3^m d/2^m)^{\kappa-1}/(\kappa-1)!$ as d increases. Using this in place of the summation above, summing over κ, and again bounding the resulting polynomial by the highest degree term gives

$$\sum_{\kappa=1}^{k} (2^m)^2 d \frac{(3^m d)^{\kappa-1}}{(\kappa-1)!} \leq c \frac{(3^m)^{k-1}}{(k-1)!} d^k = \frac{ck}{2}\left(\frac{3^m}{2}\right)^{k-1} \frac{2^k d^k}{k!}, \tag{11}$$

for fixed k and large d. Note that $n(d+k, d)$ is asymptotically equal to $2^k d^k/k!$ for large d. Hence, increasing c to account for the asymptotic approximation of the Laguerre polynomial and of the number of points in the sparse grid, we obtain the desired result. □

Proof of Theorem 4.2. Given the sparse grid, X, associated with $A(d+k, k)$ and the vector of values, $\mathbf{f}$, we need first to form the matrices π, $\hat{D}_w$, and $\hat{\Phi}$ as in (9). From the construction of π, it has exactly one nonzero entry per row and is $n \times N$, where $n = n(d, k)$ is asymptotic to $2^k d^k/k!$ as d increases [3], and $N = \sum_{d \leq |\mathbf{i}| \leq d+k} |X_{\mathbf{i}}|$. Hence π and the diagonal matrix, $\hat{D}_w$ may be constructed and applied in time $O(N)$.

A given block of $\hat{\Phi}$ is obtained by evaluating products of 1-dimensional polynomials on the points of a product grid, $X_{\mathbf{i}}$. For a nontrivial entry $\mathbf{i}_j = 1 + \kappa_j$, the polynomials in x_j have degree at most 2^{κ_j}. Using a three-term recurrence, these polynomials may be evaluated at a given point in $O(2^{\kappa_j})$. Since the projection of $X_{\mathbf{i}}$ to the x_j coordinate has $2^{\kappa_j} + 1$ points, these polynomials are evaluated in time $O(2^{\kappa_j+1})$. This is repeated for each nontrivial κ_j for a total of $O(k2^{\kappa_j+1})$, which is (very crudely) $O(k|X_{\mathbf{i}}|)$. Then for each point in $X_{\mathbf{i}}$, we multiply at most k polynomials, which is again $O(k|X_{\mathbf{i}}|)$. Hence $\hat{\Phi}$ may be constructed in time $O(kN)$.

As noted above, to find the gPC coefficients, c, we define $\hat{\mathbf{f}} = \pi\mathbf{f}$, solve $\hat{\Phi}\hat{c} = \hat{\mathbf{f}}$ for $\hat{c}$, and then use $c = \pi^T \hat{D}_w \hat{c}$. Since $\hat{\Phi}$ has blocks of size $|X_{\mathbf{i}}| \times |X_{\mathbf{i}}|$, we may solve each block in time at most $O(|X_{\mathbf{i}}|^3)$. Combining this with Proposition 4.1 and the estimates $O(N)$ and $O(kN)$ above, we obtain the theorem. □

A practical point is that many of the blocks in $\hat{\Phi}$ are identical up to a permutation of the rows. Hence for each of these blocks, we may use a single LU decomposition and appropriate permutations of the entries in $\mathbf{f}$ to

reduce the total running time (although not, of course, the linear dependence described in the theorem).

We note also that the algorithm given here is compatible with the anisotropic adaptive sparse grids of [24]. The analysis given here shows that the time per point depends only on the maximum size of a block in the factorization of Φ^{-1}, and this size is bounded by the maximum depth, k, in both the isotropic and the anisotropic cases.

5. Regularization

A limitation of the standard sparse grid method is that function values must be available at all points at a given level of refinement in order to construct a useful interpolant. This problem is overcome to some extent in anisotropic sparse grids [24], although even in this case all points must be available at each of the full product subgrids. On the other hand, much recent work in compressed sensing focuses on the fact that in the case of a sparse or nearly sparse signal, the signal may be recovered (up to some given frequency) with far fewer than the classically expected number of samples [13, 10, 12, 21].

The idea of near sparsity is relevant here when the function, f, to be interpolated is smooth, in which case the coefficients in the gPC expansion of f decay fairly rapidly. The rate of decay of these coefficients is quantified in the periodic case by inclusion in a Sobolev space. That is, if g has a representation $g(x) = \sum_{\mathbf{k}} c_{\mathbf{k}} \exp(2\pi i \mathbf{k} \cdot x)$ with $\mathbf{k}$ a d-dimensional multi-index and $x \in \mathbb{R}^d$, then g is in W_s^2 exactly when $\sum_{\mathbf{k}} |c_{\mathbf{k}}|^2 (1 + |\mathbf{k}|^2)^s$ is finite. Of course, larger s corresponds to smoother g and faster decay of $|c_{\mathbf{k}}|$. By minimizing this sum over all functions g which interpolate f on a given set of points, we obtain a minimum sobolev norm (MSN) interpolant for f on this set of points.

A generalized version of this problem and various convergence results are discussed in [18]. In particular, Theorems 2.1 and 2.2 of that paper address the existence and convergence of the resulting minimizing function. In that paper, the authors use a random set of points for interpolation and a basis of functions, $\mathbb{H}_N^d$ consisting of functions $g(x)$ as above, where the sum is over $\|\mathbf{k}\|_1 < N$ and N is chosen based on the set of interpolation points. Taking $\hat{g}(\mathbf{k}) = c_{\mathbf{k}}$ they define $g^{(s)}$ to be the function with Fourier coefficients $(1 + \|\mathbf{k}\|_2^2)^{s/2} c_{\mathbf{k}}$. Then (roughly) for $p \geq 1$ and $s > d$, there exists $c > 0$ so that given $f \in W_s^p$, and a fixed set, Y, of interpolation points, the problem of finding P^* in $\mathbb{H}_N^d$ to minimize $\|P^{(s)}\|_p$ subject to $P(Y) = f(Y)$ has a solution, P^* that satisfies $\|P^*\|_{W_s^p} < c\|f\|_{W_s^p}$. Additionally, there is a uniform bound on $|P^* - f|$ in any subset in which Y is sufficiently dense.

The application of this method to polynomial interpolation is illustrated in [16]. In this approach, a polynomial is represented in terms of a gPC expansion using a basis of tensored Chebyshev polynomials of the first kind,

T_m, normalized to be orthonormal with respect to the standard weight function. We continue to write T_m for the normalized polynomial. Then

$$P(x) = \sum_{\mathbf{k}\in K} c_{\mathbf{k}} T_{\mathbf{k}}(x), \tag{12}$$

where K is a finite index set and $T_{\mathbf{k}}(x) = T_{k_1}(x_1)\cdots T_{k_d}(x_d)$. The functions T_m satisfy $T_m(\cos x) = \cos(mx)$, so that (12) gives a Fourier cosine expansion of $P(\cos\theta_1, \ldots \cos\theta_d)$. The square of the norm of this periodic function in the Sobolev space W_s^2 is $\sum_{\mathbf{k}\in K} |c_{\mathbf{k}}|^2(1+|\mathbf{k}|^2)^s$.

To avoid overly cumbersome notation, we identify P as in (12) with the corresponding periodic function $P(\cos\theta_1, \ldots, \cos\theta_2)$, with a corresponding identification for $\hat{P}$, $P^{(s)}$, and $\widehat{P^{(s)}}$. Hence $\hat{P}(\mathbf{k}) = c_{\mathbf{k}}$, $P^{(s)}(x) = \sum_{\mathbf{k}\in K} c_{\mathbf{k}}(1+|\mathbf{k}|^2)^{s/2} T_{\mathbf{k}}(x)$, and $\widehat{P^{(s)}}(\mathbf{k}) = c_{\mathbf{k}}(1+|\mathbf{k}|^2)^{s/2}$.

Applying the MSN framework to the current setting produces the following minimization problem.

MSN ℓ_2: Given a fixed sparse grid, X, on I^d with corresponding tensored Chebyshev polynomial basis $\{T_{\mathbf{k}} : k \in K\}$, a function $f \in W_s^2(I^d)$ for some $s > 0$, a subset $\xi \subset X$, and function values $\mathbf{f}_\xi = f(\xi)$, minimize

$$\|P^{(s)}\|_2^2 = \sum_{\mathbf{k}\in K} |c_{\mathbf{k}}|^2(1+|\mathbf{k}|^2)^s$$

subject to $P(x) = f(x)$ for all $x \in \xi$.

Note that $\|P^{(s)}\|_2^2 = \|\widehat{P^{(s)}}\|_2^2$. Hence, as noted in [18], the minimization problem above is nothing but a weighted ℓ_2 minimization on coefficients of P. The weights $(1+|\mathbf{k}|^2)^s$ serve to penalize high degree (frequency) terms more than low degree terms, with the penalty increasing with s. In one sense, this is an *a priori* assumption about the size of coefficients as a function of degree and in another sense is an assumption about the smoothness class of the given function. This kind of weighting is similar in spirit to the idea of reweighting in ℓ_1 minimization [14]. In that setting, basis elements are weighted more heavily (penalized) based on the results of an unweighted ℓ_1 minimization; this often produces much better recovery than unweighted minimization alone.

Given the fact that weighted ℓ_1 minimization is an effective method for recovering a sparse signal and that the MSN approach provides a weighting to enforce sparseness (decay) based on smoothness, we propose the following minimization problem for interpolation of smooth functions.

MSN ℓ_1: With the same setting as **MSN** ℓ_2, minimize

$$\|\widehat{P^{(s)}}\|_1 = \sum_{\mathbf{k}\in K} |c_{\mathbf{k}}|(1+|\mathbf{k}|^2)^{s/2}$$

subject to $P(x) = f(x)$ for all $x \in \xi$.

For a discussion of a similar weighted derivative penalty approach in the context of multivariate splines, see Section 3.4 of [31].

Note that for large s, the weights $(1 + |\mathbf{k}|^2)^s$ may be very large, which can make the the MSN ℓ_1 and ℓ_2 problems ill-conditioned. An approach to minimize the ill-conditioning is discussed in [17]. In the current paper, we restrict to moderate s to avoid such technical difficulties. Note that the power of s in MSN ℓ_2 versus $s/2$ in MSN ℓ_1 implies that the range of reasonable s for ℓ_2 is smaller than for ℓ_2.

Convergence: As a special case of more general results, the authors of [18] prove the following convergence result for MSN ℓ_2 on arbitrary sets of interpolation points. First, given s sufficiently large, f in W_s^2, and increasing interpolation sets, ξ_j, (with basis K_j increasing accordingly), find P_j. If $x_j \in \xi_j$ with x_j converging to x_0, then $P_j(x_0)$ converges to $f(x_0)$. Using fairly straightforward estimates on derivatives, one can prove the same result for the solutions of MSN ℓ_1. In [18] there are also bounds on the rate of convergence for these solutions; we do not attempt to address this issue for MSN ℓ_1.

MSN via factorization: Each of the two versions of MSN problems given above may be reformulated in terms of the factorization given earlier. The original form of each problem is

$$\min \|D_s c\|_p \text{ subject to } \Phi c = \mathbf{f}, \tag{13}$$

where Φ is the matrix given in section 3 using as basis the tensored Chebyshev polynomials of the first kind, p is 1 or 2, and D_s is a diagonal matrix giving the weights $(1+|c_{\mathbf{k}}|^2)^{s/2}$. When Φ and f are obtained by using all the points in a sparse grid, then $\Phi c = f$ is fully determined, and the minimization problem is trivial. However, when not all function values are available or if we replace the condition $\Phi c = f$ by $\|\Phi c - f\|_2 < \sigma$ for $\sigma > 0$, then (13) is a nontrivial problem.

We may reframe this problem in terms of the factorization of section 3 as

$$\min \|D_s \pi^T \hat{D}_w \hat{c}\|_p \text{ subject to } \hat{\Phi}\hat{c} = \hat{\mathbf{f}}, \tag{14}$$

where $\hat{D}_w$, $\hat{c}$, $\hat{\Phi}$, and $\hat{\mathbf{f}}$ are all as in that section, except $\hat{\Phi}$ and $\hat{\mathbf{f}}$ may be obtained by using less than the full set of sparse grid points. There are two potential problems with this formulation. In the case in which some function values are missing, the solution to (14) may not interpolate the given points. The reason for this is that the missing values may appear more than once in the full vector $\hat{\mathbf{f}}$. That is, a given point in X may appear in more than one subgrid $X_{\mathbf{i}}$. In order to ensure that the solution to (14) interpolates the given point, we need to impose a consistency condition on the coefficients $\hat{c}$: for each missing point that appears in more than one subgrid $X_{\mathbf{i}}$, the values at that point obtained by each possible $\Phi_{\mathbf{i}} c_{\mathbf{i}}$ must all agree. This may be enforced by constructing a matrix, $\hat{Z}$, in which each row of $\hat{Z}$ is obtained from $\hat{\Phi}$ on the full set of points by taking one row of $\hat{\Phi}$ and subtracting

another row of $\hat{\Phi}$ corresponding to the same point in a different subgrid. This leads to

$$\min \|D_s \pi^T \hat{D}_w \hat{c}\|_p \text{ subject to } \hat{\Phi}\hat{c} = \hat{\mathbf{f}}, \hat{Z}\hat{c} = 0.$$

Another problem with this formulation is that for large s, the matrix D_s produces large differences in the sensitivity of the objective function to the various entries in $\hat{c}$. While this ill-conditioning cannot be removed entirely, the routine we use for solving this problem (NESTA [4]) converges more quickly when we change variables to move the scaling to the equality constraint.

Based on the structure of π, with a single 1 in each row with all other entries 0, there is a diagonal matrix $\hat{D}_s$ so that $D_s \pi^T = \pi^T \hat{D}_s$. Let $\hat{v} = \hat{D}_s \hat{D}_w \hat{c}$, $\hat{\Phi}_{sw} = \hat{\Phi}\hat{D}_s^{-1}\hat{D}_w^{-1}$, and $\hat{Z}_{sw} = \hat{Z}\hat{D}_s^{-1}\hat{D}_w^{-1}$. This gives

$$\min \|\pi^T \hat{v}\|_p \text{ subject to } \hat{\Phi}_{sw}\hat{v} = \hat{\mathbf{f}}, \hat{Z}_{sw}\hat{v} = 0. \tag{15}$$

When $p = 2$, this problem is a standard constrained quadratic programming problem, although the fact that π^T is many-to-one means that many standard techniques are not applicable. When $p = 1$, this is an ℓ_1-analysis problem in the terminology of [4], which describes the algorithm NESTA for solving such problems. NESTA is based on Nesterov's method [27], which is a fast, first-order method for finding the approximate solution to the minimizer of a nonsmooth convex function by using an appropriate smooth function in its place. NESTA extends this idea by using continuation in the smoothing parameter. This algorithm provides an efficient solution to problems of the form

$$\min \|D^T x\|_1 \text{ subject to } Ax = b.$$

The convergence rate of Nesterov's method as applied by NESTA is proportional to the operator norm of the matrix D. This provides the motivation, mentioned above, for moving the scaling due to D_s from the objective function into the constraint.

An important caveat is that this algorithm (or at least the efficiency of this algorithm) relies on the assumption that A is an orthogonal projector: $A^T A = I$. In practice, this requirement may be met by factoring $A^T = QR$ and replacing $Ax = b$ by $Q^T x = (R^T)^{-1} b$.

In the case of (15), this means that we must find the QR factorization of the matrix $[\hat{\Phi}_{sw}^T \ \hat{Z}_{sw}^T]$. Since s introduces a row scaling in this matrix, and since the QR factorization is not stable under extreme row scaling, this limits the range of possible s. Additionally, since $\hat{Z}_{sw}^T$ is not block diagonal, this means that the run time, in general, is no longer linear in the number of points. However, in the case that none of the missing points appears in more than one subgrid, then the matrix Z is redundant, so we may eliminate it and retain the block diagonal structure in the QR decomposition and the linear run time. In terms of the structure of a sparse grid, this means that

only the highest level, or terminal, subgrids may have points missing. In this case, the block diagonal structure and Proposition 4.1 imply that the QR factorization is again linear in the number of points in the sparse grid. While the more general case is not prohibitive, in the numerical examples given below, we restrict to the case of empty Z.

Noisy case: When there is error in the function values, we have the formulation

$$\min \|\pi^T \hat{v}\|_p \text{ subject to } \|\hat{\Phi}_{sw}\hat{v} - \hat{\mathbf{f}}\|_2 < \sigma, \hat{Z}_{sw}\hat{v} = 0. \tag{16}$$

In this case, the inequality constraint implies that the matrix Z must include all points that appear more than once in the full vector $\hat{\mathbf{f}}$, even if only terminal points are missing. Although NESTA can solve problems of the form

$$\min \|D^T x\|_1 \text{ subject to } \|Ax - b\|_2 \leq \sigma \tag{17}$$

when A is not an orthogonal projector it requires the singular value decomposition of A. Since $[\hat{\Phi}_{sw}^T \ \hat{Z}_{sw}^T]$ is not block diagonal, this can be very expensive. In addition, the combination of equality and inequality constraints introduces additional numerical difficulties.

Instead, we reformulate once again to combine the ℓ_1 and ℓ_2 constraints:

$$\min \|\pi^T \hat{v}\|_p + \lambda\|\hat{\Phi}_{sw}\hat{v} - \hat{\mathbf{f}}\|_2 \text{ subject to } \hat{Z}_{sw}\hat{v} = 0. \tag{18}$$

For the correct choice of λ, this is equivalent to the previous formulation, although in general there is no direct method for determining λ from σ (see e.g., [32]). Here we do not focus on the most efficient methods for determining λ from σ but use a simple line search method to determine λ. In order to use this formulation, we need a smooth version of $\|\cdot\|_2$ that fits the framework required for Nesterov's algorithm. Such a smooth version is given in [2], which gives another algorithm for solving problems of the form considered here. We note also that in this case, the matrix $\hat{Z}_{sw}$ guarantees consistency across multiple entries in $\hat{\mathbf{f}}$, so we may eliminate multiple entries (rows) from $\hat{\Phi}_{sw}$ and $\hat{\mathbf{f}}$. This implies that the same value of σ may be used to give the ℓ_2 penalty using the original points in $\mathbf{f}$ or the remaining points in $\hat{\mathbf{f}}$.

6. Numerical results

In this section we provide results of numerical experiments using the revised algorithm described in the previous section. All computations were performed in Matlab 7.7.0 on a Dell Precision PWS690 with an Intel Xeon running at 3GHz with 3GB of RAM.

Efficiency: To evaluate the running time of the conversion algorithm, we used the test function labeled Oscillatory in [3],

$$f_1(x) = \cos\left(2\pi w_1 + \sum_{i=1}^{d} c_i x_i\right),$$

where c_i and w_1 are chosen at random as indicated in [23]. The domain of definition is $[0, 1]$. The sparse grid for a given dimension and depth was created using the Matlab package spinterp, version 5.1.1 [25]. The resulting functional values were then used as input to the revised conversion algorithm using the Legendre polynomials as basis.

In Figure 1, we plot the running time of the conversion algorithm for fixed order of accuracy, k, and increasing dimension. The plot on the left clearly shows the linear dependence on the number of points evaluated. The plot on the right shows the same data as a function of the number of dimensions. Here the nonlinear increase for k bigger than 1 is due to the nonlinear dependence of the number of points of evaluation on dimension. Nevertheless, for $k = 2$, the algorithm is reasonably fast even up to 100 dimensions.

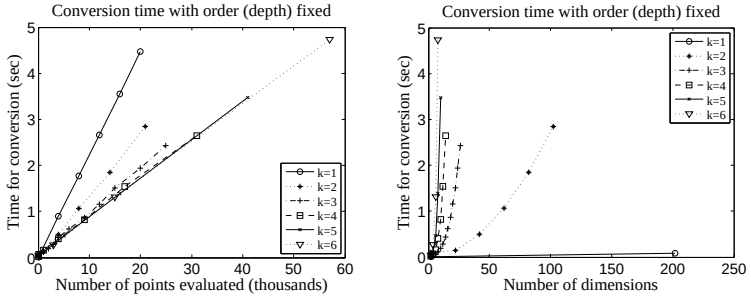

FIGURE 1. Time for conversion to Legendre basis as a function of number of points and number of dimensions. Each trace corresponds to a fixed order of accuracy (or depth, k) with the number of dimensions increasing. Left: Time versus number of points showing linear dependence on the number of points evaluated. Right: The same data as a function of number of dimensions (except $k = 1$ is truncated for scaling reasons). Given the linear relationship on the right, this is essentially a scaled plot of the number of points as a function of the number of dimensions.

In Figure 2, we plot the running time of the conversion algorithm for fixed dimension and increasing order of accuracy, k. Here the (relatively crude) bound in (11) implies that the running time is bounded by $c_1 c_2^k$ per point for some constants c_1 and c_2. However, the plot on the left shows that the deviation from nonlinear is relatively small for small k. This is made

more precise in the plot on the right, which shows the time per point as a function of k. Perfect linear dependence would imply that the traces for different dimensions would coincide. While this is essentially true for $k \geq 5$ in the data presented, there are significant deviations for small k and d. This is due to fixed overhead time that must averaged over fewer points in these cases, which gives rise to the decrease in time per point as dimension increases when $k \leq 4$. Somewhat more surprisingly, the time per point (after discounting the fixed overhead) is essentially constant up to $k = 8$.

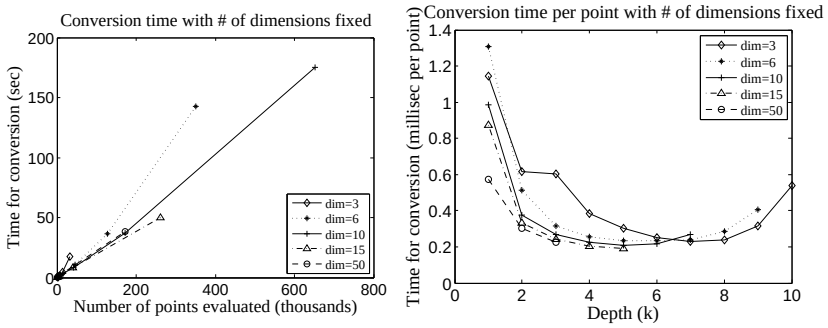

FIGURE 2. Time for conversion to Legendre basis as a function of number of points and order of accuracy (or depth, k). Each trace corresponds to a fixed number of dimensions with the depth increasing. Left: Time versus number of points showing nonlinear dependence on k, particularly in dimensions 3 and 6, for which the maximum depth is 10 and 9, respectively. Right: The same data scaled to give time per point of evaluation and shown as a function of k. Perfect linear scaling of time with number of points would imply that the time per point is the same for a given value of k, independent of dimension. Apart from fixed overhead costs that skew the results for small k, the time per point is nearly constant over a large range of feasible values of k.

We note that the interpolant in gPC form using our algorithm agrees with the Lagrange interpolant from the spinterp package up to a maximum difference of 10^{-10} at randomly selected points in the examples we've studied. Moreover, our implementation of the gPC interpolant is significantly faster per point than the interpolation in spinterp. In Figure 3, we plot the time for interpolation per point for dimension 5 with $k = 6$ and dimension 30 with $k = 3$. Not counting the time for conversion, our method is at least 10 times faster per point than spinterp. With conversion, our method is faster in these examples (and many others) when interpolating more than about 100 points. For both methods, the precise timing depends heavily on the implementation, and we do not claim that the interpolation in gPC form is intrinsically faster than barycentric interpolation. However, for applications

involving many interpolations, our method has a clear advantage over the spinterp implementation.

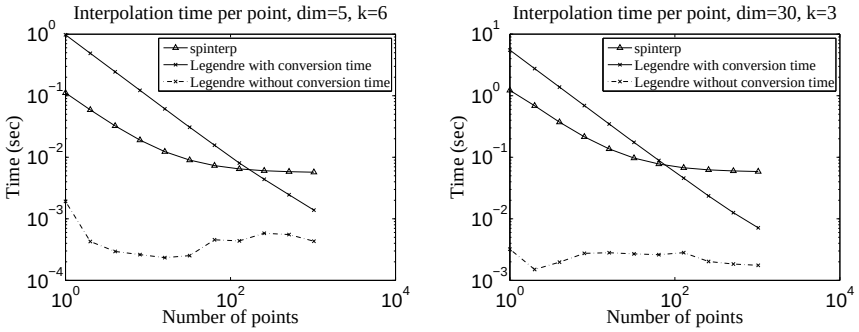

FIGURE 3. Time for interpolation per point as a function of number of points to interpolate. Left: Dimension 5, $k = 6$. Right: Dimension 30, $k = 3$. In both cases, interpolation per point (after conversion) in gPC form is much faster than interpolation using spinterp.

Regularization: To evaluate the effectiveness of the MSN ℓ_1 regularization approach, we used the test function Oscillatory as above, and also the function labeled Product Peak in [3],

$$f_2(x) = \Pi_{i=1}^{d}(c_i^{-2} + (x_i - w_i)^2)^{-1},$$

using the selection of random parameters c and w as given in [23]. However, to make comparison of error more meaningful across dimensions, we used a linear scaling of both of these functions to produce a sample mean of 0 and sample variance of 1.

We show plots of the error between these test functions and interpolating polynomials obtained by various choices of dimension, adaptive versus non-adaptive sparse grids, Sobolev exponent s, and noise. In each case, we fix a test function and dimension and plot an estimate of the L^∞ norm of the difference between the test function and an interpolating (or approximating) function, as a function of the number of points interpolated. For the method presented here, we need also to specify a sparse grid and specify a subset of points in this sparse grid for sampling. In the plots shown, we first select a set of points for evaluation, construct an interpolant, and then add points and repeat the process. Since the results depend on the points chosen, we repeat this process 8 times and plot the mean, maximum, and minimum errors. The original sparse grid may be either isotropic or anisotropic, but in either case, as noted above, we focus here on the case in which the missing points are terminal points (appear in only one subgrid).

In Figure 4, we show the (estimated) L^∞ error between interpolating polynomials and the function Oscillatory. Both plots show the expected

decrease in error as more points are sampled, using either the nonadaptive (isotropic) grid or the adaptive (anisotropic) grids of [25]. For the figure on the left, the isotropric grid of maximum depth ($k = 4$) was used to determine the basis polynomials and the allowable points for sampling in the MSN ℓ_1 problem. Points were removed at random from this grid and the MSN ℓ_1 interpolating polynomial determined as described above. This procedure was repeated 8 times with different choices of missing points. The plot shows the mean, max, and min L^∞ errors over these 8 trials. The plot on the right was constructed in the same way, using the adaptive grid to determine the basis polynomials and the allowable points for sampling.

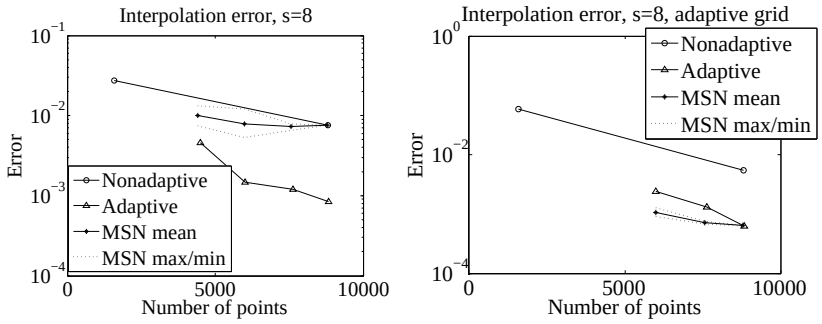

FIGURE 4. Estimated L^∞ error between interpolating polynomial and the function Oscillatory in dimension 10. Each plot shows the error obtained from the standard nonadaptive grid and from the adaptive grid as well as error from the solution to MSN ℓ_1 with $s = 8$. Left: Nonadaptive grid with $k = 4$ used to determine the basis and set of points for MSN. Right: Adapative grid used to determine basis and points.

The importance of the Sobolev exponent, s, is shown in Figure 5. The plot on the left shows the same procedure as that applied to obtain the plot on the left in Figure 4, only now $s = 0$, so that no penalty is applied to high degree/frequency terms. In this case, the error is much greater than that in the case of $s = 8$. This is consistent with the fact that this test function is analytic and hence the coefficients in the gPC expansion should decay rapidly with increasing degree. To display the robustness of these results with increasing dimension, we show on the right the same procedure in dimension 50, with $k = 2$ and $s = 8$. Again the MSN ℓ_1 procedure performs very well, in this case even better than the adaptive grid, even though the points for the MSN grid were taken from the nonadaptive grid.

As with any numerical method, this one does not perform equally well with all functions. In Figure 6, we show plots analogous to those in Figure 4, only now using the function Product Peak. In this case, the errors using the MSN procedure decrease more slowly than might be expected from the previous case.

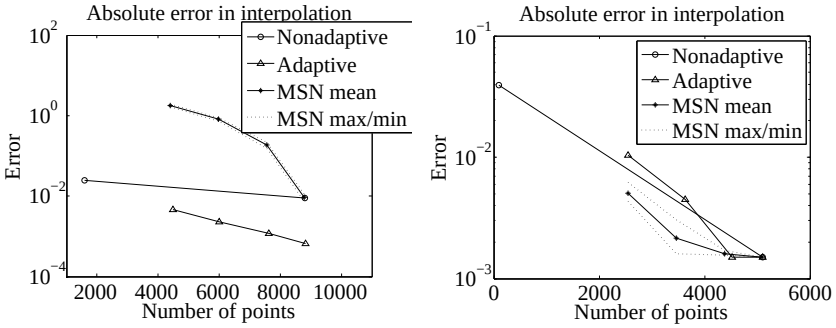

FIGURE 5. Estimated L^∞ error between interpolating polynomial and the function Oscillatory. Left: Nonadaptive grid in dimension 10 with $k = 4$ and $s = 0$. Right: Nonadaptive grid in dimension 50 with $k = 2$ and $s = 8$.

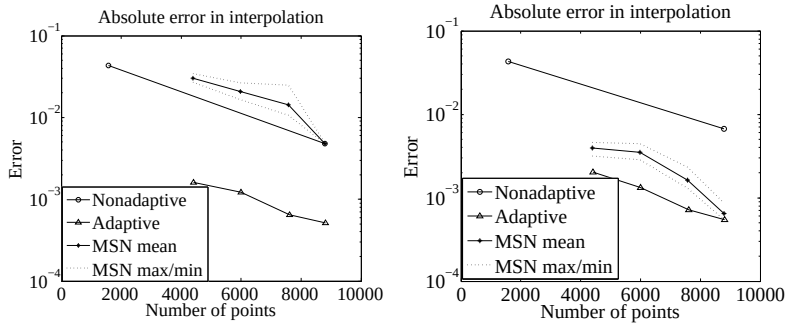

FIGURE 6. Estimated L^∞ error between interpolating polynomial and the function Product Peak. Left: Nonadaptive grid in dimension 10 with $k = 4$ and $s = 8$. Right: Adaptive grid in dimension 10 with $k = 4$ and $s = 8$.

D-Restricted Isometry Property: The paper [11] considers the problem in (17) in the case when $b = Ax + z$, with z a random variable modeling measurement error and noise. Let x^* denote the minimizer of (17). Theorem 1.4 of [11] gives a bound on the $\|x^* - x\|_2$ under the assumption that the matrix A satisfies a particular form of the D-Restricted Isometry Property (D-RIP). The D-RIP condition with constant δ_c is

$$(1 - \delta_c)\|v\|_2^2 \leq \|Av\|_2^2 \leq (1 + \delta_c)\|v\|_2^2$$

for all v obtained as a linear combination of at most c columns of D (here we use δ_c in place of the more standard δ_s to avoid confusion with the Sobolev exponent, s). Theorem 1.4 of [11] then states that if D is a tight frame, and $\delta_{7c} < 0.6$ for A, then the minimizer, x^*, of (17) satisfies

$$\|x^* - x\|_2 \leq C_0 \sigma\sigma + C_1 \frac{\|D^T x - (D^T x)_c\|_1}{\sqrt{s}},$$

where C_0 and C_1 depend only on δ_{2s}.

This theorem does not apply directly to the problem at hand because D is not a tight frame. However, as mentioned in [11], this assumption is made for ease of analysis; a version of the results still apply without this assumption. A more serious obstacle is that in the setting of (16), the set of allowable vectors, $\hat{v}$, is constrained by $\hat{Z}_{sw}\hat{v} = 0$ and additionally by the nonuniform probability of nonzero coefficients in the gPC expansion: low-degree polynomials are more likely to have large coefficients than are high-degree polynomials. For this reason, instead of estimating δ_c by selecting uniformly a random subset of columns of Π' and taking a uniform random linear combination of these columns, we instead select a random subset of columns of Π' with smaller probability for high degree terms and take the intersection of the span of these columns with the null space of $\hat{Z}_{sw}$. Since this is a computationally expensive procedure, and since the motivation for this nonuniform sampling is only heuristic at present, we did not do extensive tests. However, for dimension 5, with $k = 3$, and 20% of points missing, estiamtes for δ_c are typically in the range of 0.3, even for c close to maximal, suggesting reasonable bounds on the recovery error. Obtaining more rigorous bounds is an area for future research.

Noisy case: Because of the need to determine λ in (18), and because λ may depend quite sensitively on σ in (17), the problem of estimating the coefficients using (18) is computationally fairly expensive. Moreover, to emphasize the role of sparsity, we did not use the test functions given above directly. Instead, we computed the gPC expansion of the interpolating polynomial at a given depth and then set all gPC coefficients below a threshold to 0. We did this with the function Oscillatory (normalized to have mean 0 and std dev 1) and then added Gaussian noise with std dev 0.2.

We removed some of the sparse grid points and then solved (17) for various values of λ to approximate (18) with $\sigma = 0.2 * sqrt(n + sqrt(2 * n))/2$, where n is the number of sampled points.

We performed this procedure in dimension 2 with depth = 5 and 13 missing points out of 65 and in dimension 5 with depth = 3 and 49 missing points out of 241. For comparison, we computed the gPC expansion of the function that interpolates the given function values (with noise) exactly. The plots in Figure 7 show the original sparse coefficients on the left and the difference between these original coefficients and each of the coefficients obtained with (18) and obtained by interpolating the noisy values exactly. While the recovery is far from perfect, the coefficients from (17) in general show significantly lower error than the coefficients from exact interpolation with noise.

7. Conclusion

We have described a block-diagonal factorization of the matrix for changing basis from Lagrange polynomials to orthogonal polynomials based on

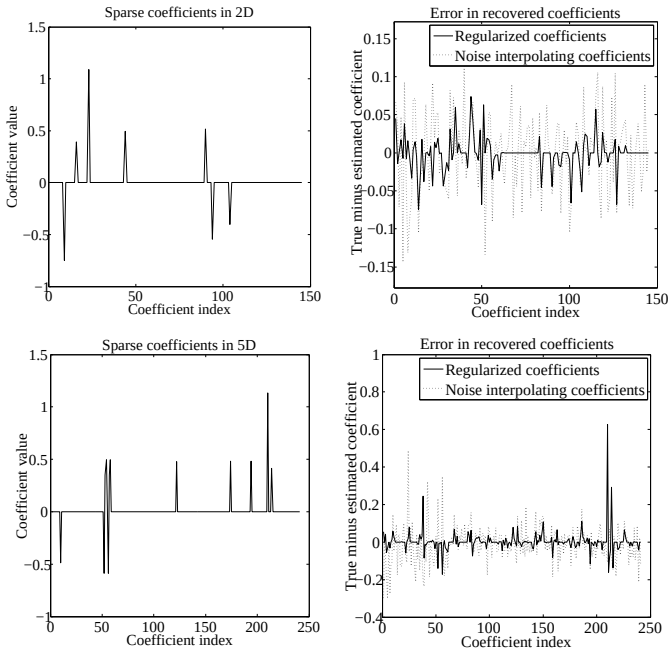

FIGURE 7. Recovery of sparse coefficients with missing values and noise in function values. Top left: gPC coefficients for target function in 2 dimensions. Top right: error in recovered coefficients (solid line), and coefficients for polynomial that interpolates the given noisy function values exactly (dotted line). Bottom left: gPC coefficients for target function in 5 dimensions. Bottom right: recovered coefficients (solid), and coefficients for exact interpolation (dotted).

function values at a Smolyak sparse grid. This factorization leads to an efficient algorithm for converting an interpolating polynomial from Lagrange form to gPC form. For a fixed degree of accuracy and increasing dimension, this algorithm is linear in the number of points of evaluation. Moreover, for fixed dimension, the time per point of evaluation is nearly constant as the degree of accuracy increases up to about $k = 8$. We also showed how to use this factorization together with ℓ_1 minimization via the algorithm NESTA to provide a good approximation to the original interpolating polynomial, even when function values are not available at some points of the sparse grid and/or when the values are corrupted by noise. Together, these results provide a significant extension to the power and flexibility provided by sparse grid interpolation.

REFERENCES

[1] Bradley K. Alpert and Vladimir Rokhlin. A fast algorithm for the evaluation of legendre expansions. *SIAM J. Sci. Stat. Comput.*, 12:158–179, January 1991.

[2] N. S. Aybat and G. Iyengar. A First-order smoothed penalty method for compressed sensing. *SIAM J. Optimization*, 21(1):287–313, 2011.

[3] V. Barthelmann, E. Novak, and K. Ritter. High dimensional polynomial interpolation on sparse grids. *Adv. Comput. Math.*, 12(4):273–288, 2000.

[4] S. Becker, J. Bobin, and E.J. Candès. Nesta: A fast and accurate first-order method for sparse recovery. *SIAM J. on Imaging Sciences*, 4(1):1–39, 2011.

[5] J.-P. Berrut and L.N. Trefethen. Barycentric lagrange interpolation. *SIAM Rev.*, 46:501–517, 2004.

[6] Alin Bostan, Bruno Salvy, and Éric Schost. Power series composition and change of basis. In *Proceedings of the twenty-first international symposium on Symbolic and algebraic computation*, ISSAC '08, pages 269–276, New York, NY, USA, 2008. ACM.

[7] John P. Boyd. Exponentially convergent fourier-chebshev quadrature schemes on bounded and infinite intervals. *Journal of Scientific Computing*, 2:99–109, 1987. 10.1007/BF01061480.

[8] J.P. Boyd. *Chebyshev and Fourier Spectral Methods*. Dover, New York, 2 edition, 2001.

[9] G.T. Buzzard and D. Xiu. Variance-based global sensitivity analysis via sparse grid interpolation and cubature. *Comm. Comp. Phys.*, 9:542–567, 2011.

[10] E. J. Candès, J. Romberg, and T. Tao. Robust uncertainty principles: exact signal reconstruction from highly incomplete frequency information. *Information Theory, IEEE Transactions on*, 52(2):489–509, February 2006.

[11] Emmanuel J. Candès, Yonina C. Eldar, Deanna Needell, and Paige Randall. Compressed sensing with coherent and redundant dictionaries. *Applied and Computational Harmonic Analysis*, 31:59–73, 2010.

[12] T. Candès, E. J.and Tao. Near optimal signal recovery from random projections: Universal encoding strategies? *IEEE Trans. Inform. Theory*, 52:5406–5425, 2006.

[13] Emmanuel Cands and Justin Romberg. Sparsity and incoherence in compressive sampling. *Inverse Problems*, 23(3):969, 2007.

[14] E. J. Cands, M. Wakin, and S. Boyd. Enhancing sparsity by reweighted ℓ_1 minimization. *J. Fourier Anal. Appl.*, 14:877–905, 2007.

[15] Michael Carley. Moving least squares via orthogonal polynomials. *SIAM J. Sci. Comput.*, 32:1310–1322, May 2010.

[16] S. Chandrasekaran, K. R. Jayaraman, J. Moffitt, H. N. Mhaskar, and S. Pauli. Minimum Sobolev Norm Schemes and Applications in Image Processing. In Truchetet, F and Laligant, O, editor, *Wavelet applications in industrial processing VII*, volume 7535 of *Proceedings of SPIE*. Soc Imaging Sci & Technol (IS&T); SPIE, 2010. Conference on Wavelet Applications in Industrial Processing VII, San Jose, CA, JAN 18-19, 2010.

[17] S. Chandrasekaran, K.R. Jayaraman, M. Gu, H.N. Mhaskar, and J. Mofftt. Higher order numerical discretization methods with sobolev norm minimization. *Procedia Computer Science*, 4:206 – 215, 2011. Proceedings of the International Conference on Computational Science, ICCS 2011.

[18] S. Chandrasekaran and H. N. Mhaskar. A construction of linear bounded interpolatory operators on the torus, 2010.

[19] Thierry Crestaux, Olivier Le Maitre, and Jean-Marc Martinez. Polynomial chaos expansion for sensitivity analysis. *Reliability Engineering & System Safety*, 94(7):1161 – 1172, 2009. Special Issue on Sensitivity Analysis.

[20] M.M. Donahue, G.T. Buzzard, and A.E. Rundell. Experiment design through dynamical characterisation of non-linear systems biology models utilising sparse grids. *Systems Biology, IET*, 4(4):249 –262, july 2010.

[21] D.L. Donoho. Compressed sensing. *IEEE Trans. Inform. Theory*, 52:1289–1306, 2006.

[22] W. Gander. Change of basis in polynomial interpolation. *Numerical Linear Algebra with Applications*, 12(8):769–778, 2005.

[23] A.C. Genz. A package for testing multiple integration subroutines. In P. Keast and G. Fairweather, editors, *Numerical Integration : recent developments, software, and applications*, pages 337–340. Kluwer, Dordrecht, 1987.
[24] Thomas Gerstner and Michael Griebel. Dimension-adaptive tensor-product quadrature. *Computing*, 71:2003, 2003.
[25] Andreas Klimke and Barbara Wohlmuth. Algorithm 847: Spinterp: piecewise multilinear hierarchical sparse grid interpolation in matlab. *ACM Trans. Math. Softw.*, 31:561–579, December 2005.
[26] Genyuan Li, Herschel Rabitz, Jishan Hu, Zheng Chen, and Yiguang Ju. Regularized random-sampling high dimensional model representation (rs-hdmr). *Journal of Mathematical Chemistry*, 43:1207–1232, 2008.
[27] Y. Nesterov. Smooth minimization of non-smooth functions. *Math. Program., Serie A*, 103:127–152, 2005.
[28] A. Saltelli, S. Tarantola, and K. P.-S. Chan. A quantitative model-independent method for global sensitivity analysis of model output. *Technometrics*, 41(1):39–56, 1999.
[29] Jie Shen and Li-Lian Wang. Sparse spectral approximations of high-dimensional problems based on hyperbolic cross. *SIAM J. Num. Analysis*, 48(3):1087–1109, 2010.
[30] S.A. Smolyak. Quadrature and interpolation formulas for tensor products of certain classes of functions. *Soviet Math. Dokl.*, 4:240–243, 1963.
[31] Curtis B. Storlie, Laura P. Swiler, Jon C. Helton, and Cedric J. Sallaberry. Implementation and evaluation of nonparametric regression procedures for sensitivity analysis of computationally demanding models. *Reliability Engineering & System Safety*, 94:1735–1763, 2009.
[32] Ewout van den Berg and Michael P. Friedlander. Probing the pareto frontier for basis pursuit solutions. *SIAM J. Sci. Comput.*, 31:890–912, November 2008.
[33] Dongbin Xiu and Jan S. Hesthaven. High-order collocation methods for differential equations with random inputs. *SIAM Journal on Scientific Computing*, 27(3):1118–1139, 2005.

Department of Mathematics, Purdue University, West Lafayette, IN 47907