The Sparse Grid Interpolant

John Burkardt
Department of Scientific Computing
Florida State University

.....

https://people.sc.fsu.edu/~jburkardt/presentations/sparse_interpolant_2013_fsu.pdf

February 14, 2024

Abstract

Smolyak's sparse grid construction is commonly used in a multidimensional setting to develop efficient quadrature rules. However, the method can be applied in a straightforward way to the interpolation problem as well. In this discussion, we outline a procedure that begins with a family of interpolants defined on a family of tensor product grids, and demonstrate how the Smolyak rule can be used to generate an interpolant of known precision.

1 Introduction

The sparse grid construction of Smolyak [1] is often presented as a method for producing a multidimensional quadrature rule which is as precise as a pure tensor product quadrature rule, but uses far fewer points. Typically, a Smolyak quadrature rule is constructed as a weighted sum of low order interpolatory quadrature rules associated with a family of (possibly nested) tensor product grids. But interpolatory quadrature rules are derived by integrating an underlying interpolation rules. The Smolyak procedure, applied to quadrature rules, produces its result from a simple linear combination of the quadrature rules.

This suggests that we might regard a Smolyak quadrature rule as the result of integration applied to an interpolation procedure constructed from the Smolyak principle. We can then imagine reversing the integration process, to recover the Smolyak interpolation procedure explicitly. We naturally begin by recalling the interpolatory rules associated with the component quadrature rules, and then supposing the linear coefficients associated with the Smolyak procedure are applied to the appropriate selection of these rules.

As part of the analysis, we specify the kind of interpolation problems that we want to consider. Our version of the interpolation problem will assume that there is an underlying but unknown function $F()$, which we are free to sample at an arbitrary set of finitely many interpolation points $\{X_i\}$ lying in some product space X. Our task is to construct a function $G()$ from some specific interpolation space, with the property that $G(X_i) = F(X_i)$ at the interpolation points.

Moreover, we desire the ability to consider a family of such interpolating schemes, so that we can choose the degree of polynomial precision in any coordinate direction; we might also wish to be able to pick a particular sequence of elements of such a family for which the set of interpolation points is nested.

The Smolyak procedure, when applied to the quadrature problem, assumes the existence of a family of 1D quadrature rules (or, more generally, a separate family of such rules for each coordinate dimension). We will correspondingly assume the existence of a procedure for producing a family of interpolation rules whose form is determined by the choice of interpolation nodes. In fact, for a given interpolation interval, and a given value of n, the number of interpolation nodes, we may decide to specify the choice of interpolation nodes, in which case the 1D procedure can be entirely specified by n.

To apply the Smolyak procedure to the interpolation problem, we need a family of multidimensional interpolants; we plan to construct these interpolants from product rules applied to 1D interpolants. Thus, it is necessary now to describe a typical such family and show how it can be used as the first step in this process.

2 1D Interpolation Rules

The 1D interpolation problem can be posed as follows: Given a set of interpolation nodes $\xi = \{\xi_i\}$ with $\xi_1 < \xi_2 < ... < \xi_n$ which lie in some interval $[a, b]$, and a corresponding set of interpolation values $\eta = (\eta_1, \eta_2, ..., \eta_n)$, determine a function $g(x)$ from some suitable function space $\mathcal{G}$, which can be evaluated for any $x \in [a, b]$, and which has the property that $g(\xi_i) = \eta_i$, for $i = 1, ..., n$.

Since we assume the interpolation points are distinct, we may, in particular, choose the interpolating function to be $p(n, \xi, \eta)(x)$, the unique polynomial of order n (or, equivalently, of degree $n - 1$) which satisfies the interpolation conditions. There are two well-known procedures for representing and computing $p(n, \xi, \eta)(x)$, either as a Newton divided difference polynomial, or as a weighted sum of Lagrange basis functions.

The important thing to note is that, if we have picked n, the number of interpolation points, and ξ, the set of interpolation points, then the form of the interpolating polynomial $p(n, \xi, \eta)(x)$ is automatically determined.

If we furthermore decide to prescribe the set of interpolation points ξ as a function of n, we will have arrived at a family of 1D interpolating functions which are parameterized by n, and which can solve the interpolation problem for an arbitrary set of data η. Since we are assuming that we are free to specify the interpolation points $\{xi_i\}$, it presumably follows that the values $\{\eta_i\}$ can be produced for us, presumably by an underlying, unknown, and possibly expensive function $f(x)$. Note that we do not expect to be asked to produce an interpolating function given at an arbitrary set of nodes; instead, our procedure may be thought of as proposing a family of polynomial interpolants, whose parametric order n can be increased as desired to approximate a function which we can query on the interpolation points.

Although a natural choice for interpolation points might appear to involve equally spaced values in the interval $[a, b]$, this is generally not the best choice, and so we will briefly list several possibilities for the choice of n interpolation points:

- **NCC**: *Newton-Cotes-Closed* uses equally spaced points, including the endpoints;
- **NCO**: *Newton-Cotes-Open* uses equally spaced points, excluding the endpoints;
- **CC**: *Clenshaw-Curtis* uses the extrema of the Chebyshev polynomial $T_{n-1}(x)$, mapped to [a,b];
- **F1**: *Fejer Type 1* uses the zeros of the Chebyshev polynomial $T_n(x)$, mapped to [a,b];
- **F2**: *Fejer Type 2* uses the extrema of the Chebyshev polynomial $T_{n+1}(x)$, mapped to [a,b] and omitting the endpoints.

Our default choice for the 1D interpolation points will be the "CC" points.

It is worth determining a representation of the 1D interpolating polynomial $p(n, \xi, \eta)(x)$. We choose to develop the Lagrange representation. Thus, with each interpolation node ξ_i we associate the i-th Lagrange basis polynomial $l(i, n, \xi)(x)$ defined by

$$l(i, n, \xi)(x) = \frac{\prod_{j \neq i} x - \xi_j}{\prod_{j \neq i} \xi_i - \xi_j}$$

It is easy to see that $l(i, n, \xi)(\xi_j) = \delta_{ij}$, and that each basis function is a polynomial of degree $n - 1$ and hence, by uniqueness, we can define the polynomial interpolant to the set of n data values ξ, η by:

$$p(n, \xi, \eta)(x) = \sum_{i=1}^{n} \eta_i \; l(i, n, \xi)(x)$$

because the quantity on the right clearly satisfies the interpolation conditions.

3 1D Interpolation Families

So far, we have seen a number of choices for 1D interpolation rules over a given interval. For any positive order n, each rule has a well-defined procedure for selecting the interpolation abscissas.

We now wish to create a family of interpolation rules, that is, an indexed selection of instances of an interpolation rule. The selected rules will be of increasing order, and, in general, we will be interested in guaranteeing that the rules are *nested*, that is, that all the abscissas of a given rule are included in the next one. Using a family of nested rules will have significant advantages when carrying out a sequence of calculations.

The selection process can be defined entirely in terms of the orders of the successive elements. For the rules we have considered so far, the following initial sequences of orders suggest how a nested family can be defined:

- **NCC**: 1, 3, 5, 9, 17, $2^j + 1$;
- **NCO**: 1, 3, 7, 15, 31, $2^{j-1} - 1$;
- **CC**: 1, 3, 5, 9, 17, $2^j + 1$;
- **F1**: 1, 3, 9, 27, 3^j;
- **F2**: 1, 3, 7, 15, 31, $2^{j-1} - 1$.

Once we have determined a family of interpolation rules, we can refer to particular elements by indexing the name of the family. Thus NCC^3 is the third element of the NCC family, the Newton Cotes Closed rule of order 5. When referring to a generic family, we will write F, with generic element F^i.

4 Product Interpolation in a Multi-dimensional Space

Now suppose that the interpolation problem is posed on an m-dimensional region which is the product of 1D intervals.

The m-dimensional product interpolation problem can be posed as follows: Given a set of interpolation nodes $\Xi = F^{i_1} \otimes F^{i_2} \ldots \otimes F^{i_m}$, which lie in some product interval $[A, B] = [a_1, b_1] \otimes [a_2, b_2] \ldots \otimes [a_m, b_m]$, and a corresponding set of interpolation values $H = \eta_1 \otimes \eta_2 \ldots \otimes \eta_m$, determine a function $G(X)$ from some suitable function space $\mathcal{G}$, which can be evaluated for any $X \in [A, B]$, and which has the property that $G(\Xi_i) = H_i$, for $i \in n_1 \otimes n_2 \ldots \otimes n_m$.

Assuming that F^i is a 1D interpolation family suitable for the interval $[a_i, b_i]$ GOD THIS IS HARD TO ABSTRACTIFY.

5 Sparse Grid Interpolation in a Multi-dimensional Space

Sparse grid interpolants are for a given spatial dimension m are indexed by a quantity ℓ called the *sparse grid level*, and a particular sparse grid interpolant can be represented by $\mathcal{I}(\ell, m)$. The Smolyak construction defines the value of $\mathcal{I}(\ell, m)$ in terms of a linear combination of product rules $\mathcal{P}^{\mathbf{i}}$ whose indices $\mathbf{i}$ satisfy a constraint:

$$\mathcal{I}(\ell, m)(x) = \sum_{\ell - m + 1 \leq |\mathbf{i}| \leq \ell} (-1)^{\ell - |\mathbf{i}|} \left(\begin{array}{c} m - 1 \\ \ell - |\mathbf{i}| \end{array} \right) \mathcal{P}^{\mathbf{i}}(\mathbf{x})$$

One way of interpreting this formula is to conclude that, given ℓ, m and a multidimensional point x, we should construct each product interpolant $\mathcal{P}^{\mathbf{i}}$ whose index $\mathbf{i}$ satisfies the criterion, evaluate the interpolant numerically and add the weighted value to a running sum. This might be termed the "external" or "nonintrusive" sparse grid interpolant.

Take, for instance, the case where we are interpolating a function of the form:

$$f(x,y) = (0.8\sqrt{x^2+y^2} + 0.35\sin(2.4\pi\sqrt{x^2+y^2}/\sqrt{2}))1.5\sin(1.3\arctan(y/x))$$

over $[0,+1]^2$, so that $m = 2$, and we have chosen $\ell = 1$. We wish to determine the interpolated value at $x = (0.4, 0.6)$. Suppose we are using the 1D interpolation family CC.

Then we will have:

$$\begin{aligned}\mathcal{P}^{(0,0)}(x) &= 0.4604\\ \mathcal{P}^{(1,0)}(x) &= 0.4785\\ \mathcal{P}^{(0,1)}(x) &= 0.5818\end{aligned}$$

and hence

$$\mathcal{I}(\ell,m)(x) = -1 * 1 * 0.4604 + 1 * 1 * 0.4785 + 1 * 1 * 0.5818 = 0.5999$$

Here, the exact value of $f(0.4, 0.6) = 0.5035$.

Alternatively, the formula may be interpreted in an "internal" or "intrusive" way as:

$$\mathcal{I}(\ell,m)(x) = \sum_{\ell-m+1\le|\mathbf{i}|\le\ell} (-1)^{\ell-|\mathbf{i}|} \left(\begin{array}{c} m-1 \\ \ell-|\mathbf{i}| \end{array} \right) I^{i_1}(x_1) \times \cdots \times I^{i_m}(x_m)???$$

that possibility, and several related ones that are There are standard techniques for producing a family of interpolants g(j)(x), such that, as we increase the index j, the interpolating process will exactly match any function f(x) which happens to be a polynomial of limited degree. The usual techniques for doing this are derived from tensor product interpolants, whose expense grows exponentially with the spatial dimension.

For simplicity, we'll assume that the argument space X is the unit hypercube, and that we have a family of 1D interpolation rules g(j)(), where the index j is an indication of the precision of the interpolation rule. A typical tensor product interpolant G(J)() can then be thought of as constructed by the product G(J)(X) = g(j1)(x1) * g(j2)(x2) * ... * g(jm)(xm). where, of course, J = (j1,j2,...jm) and X = (x1,x2,...,xm).

A procedure due to Smolyak, which is more typically applied to problems in quadrature, can also be used for the interpolation problem. The Smolyak interpolation rule of level L is defined by A(L,M) = sum (L-M+1 ¡= —J— ¡= L) C(—J—) * g(j1)(x1) * g(j2)(x2) * ... * g(jm)(xm). Here —J— = j1+j2+...+jm, and, for each —J—, the sum must be taken over all possible vectors J with nonnegative integer entries that sum to —J—.

Some improvements to this approach can be suggested. First, many of the coefficients C() may be zero, because the coefficient vector C for an M-dimensional sparse interpolant of level L will have at most M nonzero coefficients. Secondly, if the 1D interpolation family is chosen so that the interpolant points of successive members are nested, then it is possible to simplify the evaluation process greatly.

6 Smolyak's Procedure for Multi-dimensional Quadrature

A sparse grid is constructed as the weighted sum of a set of product rules. Each product rule has the form of a direct product of 1D quadrature rules. For each dimension j, the 1D quadrature rules come from an

indexed family. While it is possible to use a different family of rules for each dimension, for this document it is sufficient to assume that the same indexed family is used to supply 1D quadrature rules for every dimension. Thus, a typical element of the family of 1D quadrature rules might be denoted by $\mathcal{U}^j$. The rules of the quadrature family are indexed by the variable j, which is called the *1D level*. The index j begins at 0, and the corresponding first 1D quadrature rule is almost always a 1 point rule, such as the midpoint rule. It is expected that as the index j increase, so will the precision of the corresponding indexed rule.

We can specify a product rule for an M-dimensional space by creating a *level vector*, that is, an M-vector i, each of whose entries is the 1D level j which indexes the rule to be used in that spatial coordinate. The resulting product rule may be written as $\mathcal{U}^{j_1} \otimes \cdots \otimes \mathcal{U}^{j_M}$. We say that this product rule has a *product level* of $|\mathbf{i}| = \sum_{m=1}^{M} j_m$. Since the lowest value of a 1D level is taken to be 0, the lowest value of $|\mathbf{i}|$ is also 0. Thus, the product rule of product level 0 is formed by M factors of a 1 point rule, each having 1D level of 0.

A product rule is a quadrature rule for approximating an integral; a sparse grid is a refined method for approximating integrals that uses weighted combinations of product rules. A sparse grid rule can be indexed by a variable called the *sparse grid level*, here symbolized by L. The lowest value of L is taken to be 0. The sparse grid of sparse grid level L is formed by weighted combinations of those product rules whose product level $|\mathbf{i}|$ falls between $L - M + 1$ and L.

The first sparse grid, of sparse grid level $L = 0$, will be formed of all product grids with product levels between $1 - M$ and 0. The lowest possible product level is actually 0, and is attained only by the M-dimensional product of the midpoint rule (or whatever the first 1D quadrature rule is). So the sparse grid rule of sparse grid level 0 is equal to the product rule of product level 0.

Each sparse grid rule $\mathcal{A}(L, M)$ is a weighted sum of the selected product rules. The selection is based on the product levels, and each weighting coefficient is a power of -1 times a combinatorial coefficient.

References

[1] Sergey Smolyak, Quadrature and interpolation formulas for tensor products of certain classes of functions, Doklady Akademii Nauk SSSR, Volume 4, 1963, pages 240-243.