Slow Exponential Growth for Clenshaw Curtis Sparse Grids

John Burkardt
Interdisciplinary Center for Applied Mathematics
& Information Technology Department
Virginia Tech
.....
http://people.sc.fsu.edu/∼burkardt/presentations/sgmga_ccs.pdf

April 4, 2014

Abstract

When Clenshaw Curtis rules are used to form a sparse grid, the orders of the underlying 1D factor rules form an exponential series. Even for a relatively low level, the 1D order growth is unnecessary, and is reflected in a noticeable cost in the order of the resulting sparse grid. We consider the effect of using the Clenshaw Curtis rules in a way that maintains the nestedness but delays their exponential growth so effectively that it becomes essentially linear. This restraint, in turn, brings down the cost of the sparse grids while entirely meeting the desired precision levels.

1 Introduction

The sparse grid construction of Smolyak [5] is a method for producing a multidimensional quadrature rule as a weighted sum of product rules, which are in turn typically formed by the product of 1D rules selected from an indexed family. A common choice for the indexed family is the Clenshaw Curtis quadrature rule. The rules in this family form a nested set, so that all the abscissas from one rule are included in the next. The nestedness requires that the order of the rules in the indexed family grows exponentially.

The sparse grid construction produces a family of sparse grids, which are indexed in a way that is related to the indexes of the underlying 1D rules. The exponential growth in the order of the Clenshaw Curtis 1D family affects the order of the sparse grid, although it is not so severe as to cause the sparse grid family order also to increase exponentially. The high order of the later 1D rules is also reflected in the ability of the sparse grid to exactly integrate certain high degree monomials; however, this extra accuracy does not extend the total precision of the sparse grid, and so, at least in the asymptotic sense, it is wasted, excessive precision.

The effect is particularly noticeable for low dimensional sparse grids. The usual sparse grid construction expects that any sparse grid of Clenshaw Curtis factors, of sparse grid level $\mathbf{l}$, will achieve a precision of $2\,\mathbf{l}+1$. But if we formally construct a 1D sparse grid out of the Clenshaw Curtis family, this simply selects successive members of the family, and so we greatly exceed our accuracy requirement by achieving accuracy of $2^{\mathbf{l}}+1$, and we do so at a corresponding cost of a high order.

It turns out that this straightforward construction is not the only way to build a sparse grid that takes advantage of the nestedness of the Clenshaw Curtis family. Smolyak's construction does not require that successive members of the family be distinct. It is really only important that the representative rule chosen for a given level achieves the desired precision level. If we construct a Clenshaw Curtis family of sparse grids with this precision requirement in mind, we can come close to achieving linear growth in the order of the 1D factors. This economy is then reflected in the resultant sparse grid, while maintaining the desired precision, per sparse grid level, of $2\,\mathbf{l}+1$.

In the following discussion, we describe the standard Clenshaw Curtis family of 1D rules with exponential growth, which we will denoted by "CC_E", and then indicate how a related family, denoted by "CC_SE" can be chosen so that it exhibits "slow exponential growth".

We will then compare the sparse grids that can be constructed with these two related families, and exhibit the resulting order growth.

2 "CC_E": Clenshaw Curtis Exponential Growth Family

Clenshaw Curtis quadrature is a simple family of rules for estimating integrals. By using unequally spaced points, it avoids the instability that makes Newton Cotes quadrature unusable beyond the first few orders. The values of the abscissas and weights can be easily computed, and because of symmetry, a rule of odd order **n** has degree of precision **n**, that is, it can integrate exactly the **n+1** monomials $1, x, x^2, \ldots, x^n$.

The relative merits of Clenshaw Curtis and Gauss Legendre quadrature rules are often compared [6]. Gauss Legendre quadrature has the marked advantage that a rule of order **n** attains degree of precision **2*n-1**. On the other hand, the form of the Clenshaw Curtis rules allows the construction of a nested family of increasing order. Such nested families can offer efficiency for certain kinds of calculations, because of the reuse of function values. It also means that when computing an integral estimate, one automatically has available the integral estimates from all the lower rules in the family, so that convergence estimates are exceptionally cheap.

The Clenshaw Curtis rule is usually developed over the interval [-1,1]. The classical nested family of Clenshaw Curtis rules begins with the midpoint rule, followed by a rule of order 3 which adds the interval endpoints, and then by rules which successively add points between each pair of points in the preceding rule. Let us index this family by a variable **lcc**, denoting the *level* of a rule, with the midpoint rule having level 0. Moreover, let **occ** denote the *order* of a rule, that is the number of points or abscissas. Finally, let **pcc** denote the *polynomial degree of precision*, that is, the value such that all monomials up to and including degree **pcc** are integrated exactly by the rule. Then the order as a function of level is:

$$\mathbf{occ}(\mathbf{lcc}) = \begin{cases} 1, & \text{if } \mathbf{lcc} = 0; \\ 2^{\mathbf{lcc}} + 1, & \text{otherwise;} \end{cases}$$

while the precision suggests why odd-order rules are preferred:

$$\mathbf{pcc}(\mathbf{lcc}) = \begin{cases} \mathbf{occ}(\mathbf{lcc}), & \text{if } \mathbf{occ}(\mathbf{lcc}) \text{ is odd;} \\ \mathbf{occ}(\mathbf{lcc}) - 1, & \text{if } \mathbf{occ}(\mathbf{lcc}) \text{ is even;;} \end{cases}$$

This means that the classical nested family of Clenshaw Curtis rules provides a list of rules whose order and precision increases exponentially with the level index. Such a growth pattern is not uncommon; for instance, the Gauss Patterson rules behave in almost the same way. The rapid growth in order does mean, however, that the family becomes unusable for relatively low values of **lcc**; for instance, a rule of level 20 would require about a million points, and the corresponding claim of a polynomial precision of a million has become a numerical fiction. See Figure 1.

3 "CC_SE": Clenshaw Curtis Slow Exponential Growth Family

When we constructed our family of 1D Clenshaw Curtis rules, we did so by seeking a nested family indexed in such a way that each increased in the level resulted in a (nested) rule of higher order, with the higher order, in turn, guaranteeing an increase in accuracy. Thie makes it possible to use the family in an adaptive calculation, in which increasing the level guarantees a more accurate estimate. The cost of this arrangement is incurred in the way that the order of the rules increases exponentially.

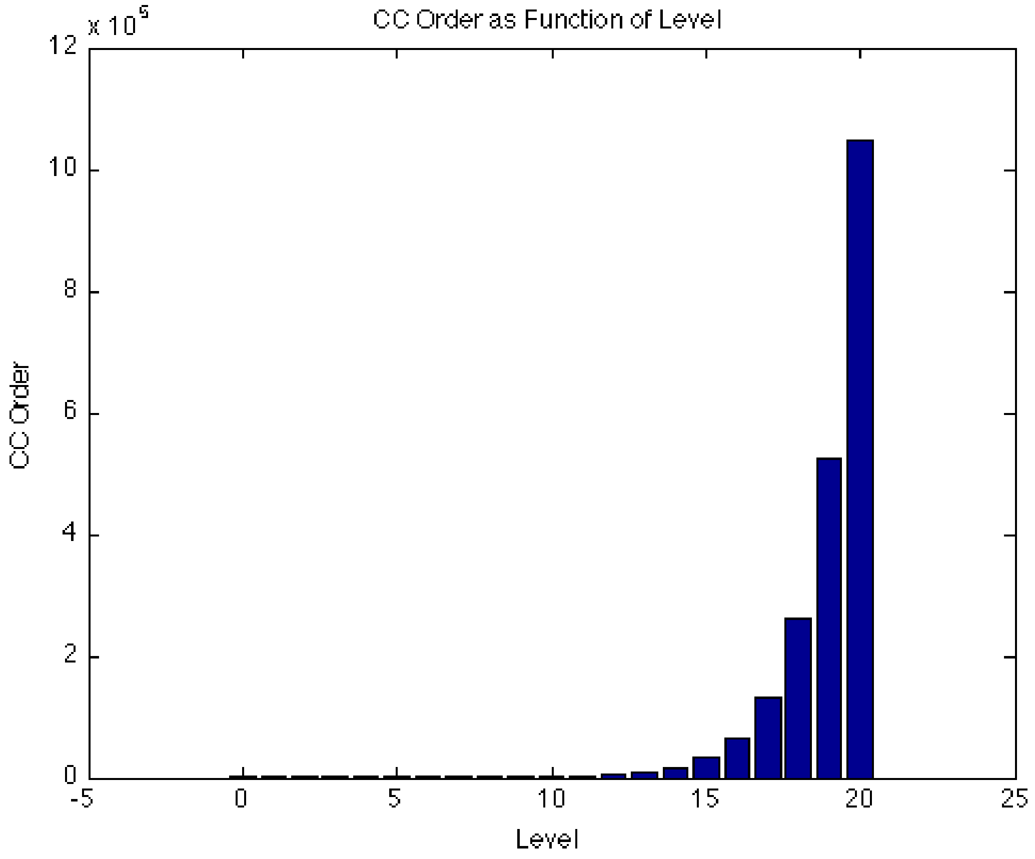

Figure 1: **Classical Nested Clenshaw Curtis Family**
The classical nested CC_E family has an exponential growth in the order.
At level 10, the order is 1,024 and at level 20, it is 1,048,576.

But this is not the only way to set up an adaptive calculation based on Clenshaw Curtis rules (or indeed other nested families). Suppose, instead, we consider each level of the adaptive procedure to be characterized by the requirement that a specific minimum polynomial precision be reached. If we count our levels beginning at 0, and if we make the typical choice of increasing precision by 2 each time, then the relationship between the level **l** and the minimum precision requirement $\mathbf{p_{min}}$ would have the simple linear form:

$$\mathbf{p_{min}}(\mathbf{l}) = 2\,l + 1 \tag{1}$$

If on level **l** we employ the **l**-th element of the classical nested Clenshaw Curtis family, then of course we easily meet (and generally greatly exceed) this precision requirement. But now that we have asserted that a linear increase in accuracy is our main concern, we can turn to the question of choosing the lowest order rule in our Clenshaw Curtis family that achieves that specific precision requirement.

For levels 0 through 4, the 1D rules selected by the CC_E family have precisions 1, 3, 5, 9 and 17. The corresponding minimum precision requirements are 1, 3, 5, 7 and 9. Thus, we can see that at level 3, the CC_E(3) rule exceeds the accuracy level required, and, moreover, is still good enough to meet the accuracy requirement at level 4. So if guaranteeing accuracy is our concern, then there is really no good reason to advance the quadrature rule from CC_E(3) to CC_E(4) when we move to the precision requirement of level 4.

Thus, we can now consider the construction of a related family of quadrature rules, the "CC_SE" family, which is defined as follows: CC_SE(l), the quadrature rule for level **l**, is to be chosen as the lowest order member of the CC_E family which satisfies the linear precision requirement, Equation 1.

This dramatically slows down the growth of the order as a function of level. In particular, for the CC_E formulation, the rule of order 65 (and precision 65) occurs at level 6, though the requested accuracy is only 13. For the CC_SE formulation, level 6 uses the rule of order 17 (and precision 17). Moreover, that same rule continues in use through level 8, only then jumping to the rule of order 33, which holds tuntil level 16, then jumping to the rule of 65, which is maintained through level 32. It should be clear that, for the CC_SE grid, the order growth becomes essentially linear, rather than exponential, as a function of the level L.

Here, we will write **lcc(lccs)** to indicate the level, in the standard family, of the rule which is to be used at level **lccs** in the "slow" family, while **occs(lccs)** indicates the order of the CC_SE rule and **pccs(lccs)** indicates the precision. Now that we consider the minimum precision to be the guiding value in defining what happens at a particular level, we have the following formulas:

$$\begin{aligned}
\mathbf{p_{min}}(\mathbf{lccs}) =& 2\,\mathbf{lccs} + 1\\
\mathbf{lcc}(\mathbf{lccs}) =& \text{ceil}(\log_2(\mathbf{lccs}) + 1)\\
\mathbf{occs}(\mathbf{lccs}) =& 2^{\mathbf{lcc}(\mathbf{lccs})} + 1\\
\mathbf{pcc}(\mathbf{lccs}) =& \mathbf{occs}(\mathbf{lccs})
\end{aligned}$$

These facts can be used to compare the minimum precision requirement to the acheived precision:

$$\begin{aligned}
\mathbf{p_{min}}(\mathbf{lccs}) =& 2\,\mathbf{lccs} + 1\\
=& 2^{\log_2(\mathbf{lccs})+1} + 1\\
\le& 2^{\text{ceil}(\log_2(\mathbf{lccs})+1)} + 1\\
=& 2^{\mathbf{lcc}(\mathbf{lccs})} + 1\\
=& \mathbf{pccs}(\mathbf{lccs}).
\end{aligned}$$

and hence we have that the CC_SE element of level **lccs** will achieve the desired precision $2\,\mathbf{lccs}+1$ or better, and by definition, it is the lowest order rule in the nested family that will match this precision requirement.

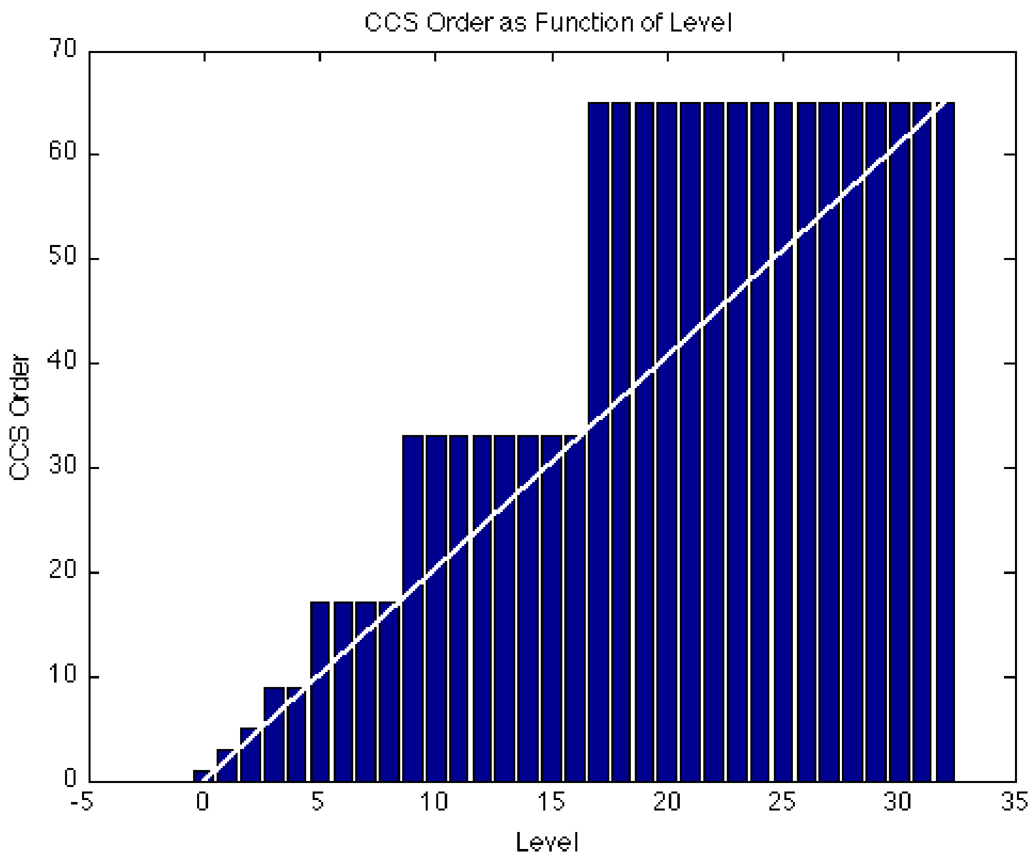

Figure 2: **"Slow" Nested Clenshaw Curtis Family**
The slow nested CC_SE family has an approximately linear growth in the order.
At level 10, the order is 32 and at levels 20 and 30, it is 64.
The white line indicates the function $\mathbf{o} = 2 * \mathbf{l} + 1$.

4 Sparse Grids

The real motivation for this discussion is the use of Clenshaw Curtis rules for the formation of sparse grids, which can be used to estimate integrals of functions of a multidimensional argument. In this context, the exponential order growth of the classical CC_E grid results in an anomalous result when we compute a formal 1D sparse grid. More seriously, when a sparse grid is constructed for a multidimensional problem, the order of that sparse grid is unneccessarily increased by the use of high order 1D rules as factors.

In order to discuss this issue, we review the procedure by which a sparse grid is formed. Note that the discussion begins at a general level, without specifying the particular family of 1D quadrature rules that are employed as factors. In fact, a different family could be used in each spatial dimension, but that will not concern us here. We will implicitly assume that the family is either the classical or slow Clenshaw Curtis family. Once we have seen how the family of 1D rules forms product rules that are combined into a sparse grid, we can address the question that concerns us, namely, the attainment of a specified degree of precision with a relatively moderate point count.

We suppose, then, that we are interested in computing sparse grids for the quadrature of functions $\mathbf{f(x)}$ where $\mathbf{x}$ is an element of an $\mathbf{M}$-dimensional space.

We assume that we have an indexed family of 1D quadrature rules $\mathcal{U}^i$. These quadrature rules are indexed by the variable i, which is called the *1D level*. The index i begins at 0, and the corresponding first 1D quadrature rule is often taken to be the midpoint rule. It is understood that as the index i increases, so do both the order (number of points) and the polynomial accuracy of the rule; the exact nature of these relationships is not necessary to specify yet; it is enough to understand that accuracy increases with i.

We can specify a product rule for an $\mathbf{M}$-dimensional space by creating a *level vector*, that is, an $\mathbf{M}$-vector $\mathbf{i}$, each of whose entries is the 1D level which indexes the rule to be used in that spatial coordinate. The resulting product rule may be written as $\mathcal{U}^{i_1} \otimes \cdots \otimes \mathcal{U}^{i_M}$. We say that this product rule has a *product level* of $|\mathbf{i}| = \sum_{k=1}^{M} i_k$. 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 $\mathbf{M}$ factors of the midpoint rule, each having 1D level of 0.

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

The formula for the sparse grid has the form:

$$\mathcal{A}(L,M) = \sum_{L-M+1 \le |\mathbf{i}| \le L} (-1)^{L-|\mathbf{i}|} \begin{pmatrix} M-1 \\ L-|\mathbf{i}| \end{pmatrix} (\mathcal{U}^{i_1} \otimes \cdots \otimes \mathcal{U}^{i_M})$$

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

Each sparse grid $\mathcal{A}(L,M)$ 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.

5 Sparse Grids Using the CC_E and CC_SE Families

Formally, the only thing we have to do to describe a sparse grid using the CC_E family is to substitute $\mathbf{CC^{i_k}}$ for each generic quadrature rule $\mathcal{U}^{i_k}$. It may be instructive, though, to write out the specific formulas for a few sparse grids in dimension 3, exhibiting how the sparse grid is formed by the sum of rules which are products of the 1D CC rules:

$$\mathcal{A}(0,3) = 1 \cdot CC^0 \otimes CC^0 \otimes CC^0$$

$$\begin{aligned}\mathcal{A}(1,3) =& 1\cdot CC^1\otimes CC^0\otimes CC^0\\ &+1\cdot CC^0\otimes CC^1\otimes CC^0\\ &+1\cdot CC^0\otimes CC^0\otimes CC^1\\ &-2\cdot CC^0\otimes CC^0\otimes CC^0\end{aligned}$$

$$\begin{aligned}\mathcal{A}(2,3) =& 1\cdot CC^2\otimes CC^0\otimes CC^0\\ &+1\cdot CC^1\otimes CC^1\otimes CC^0\\ &+1\cdot CC^1\otimes CC^0\otimes CC^1\\ &+1\cdot CC^0\otimes CC^2\otimes CC^0\\ &+1\cdot CC^0\otimes CC^1\otimes CC^1\\ &+1\cdot CC^0\otimes CC^0\otimes CC^2\\ &-2\cdot CC^1\otimes CC^0\otimes CC^0\\ &-2\cdot CC^0\otimes CC^1\otimes CC^0\\ &-2\cdot CC^0\otimes CC^0\otimes CC^1\\ &+1\cdot CC^0\otimes CC^0\otimes CC^0\end{aligned}$$

Now we know already the precision of the 1D CC rules. Novak and Ritter [2] have shown that a sparse grid of level **l**, constructed from the usual CC_E family in this way, will have total precision $\mathbf{2\,l+1}$. By this expression we mean that such a rule can correctly compute the integral of any monomial $\prod_{i=1}^{M} x_i^{e_i}$ for which the exponents are all nonnegative, and the sum of the exponents satisfies:

$$\sum_{i=1}^{M} e_i \leq 2\,l+1$$

Except in the case $M = 1$, this precision result is *sharp*, that is, there will be monomials of total degree $2\,l+2$ which cannot be exactly integrated by the rule.

The fact that the precision of the sparse grid grows linearly with the level, whereas the order and precision of the 1D CC factors grows exponentially, is the anomaly that first suggested that there might be a more economical way of constructing sparse grids from 1D CC rules.

Formally, the construction of a sparse grid is the same whether the CC_E or CC_SE family is used. Thus, the formulas for the CC_E versions of $\mathcal{A}(0,3), \mathcal{A}(0,3)$, and $\mathcal{A}(0,3)$ become formulas for the CC_SE versions simply by replacing each 1D CC_E rule by the corresponding 1D CC_SE rule.

Since the two 1D families are identical through the third level, it is only when a sparse grid reaches level 4 that we begin so see any differences in total order, corresponding to the use of the more economical CC_SE family.

It should be clear from the definition of the CC_SE family that we are guaranteed to use the minimal number of points in the 1D rules necessary to achieve the 1D precision requirement. It follows from Novak and Ritter [2] that a sparse grid constructed from the CC_SE family will also achieve the total precision $\mathbf{2\,l+1}$ of the CC_E family. In other words, the higher order rules used by the underlying CC_E family are not necessary to achieve the total precision requirement associated with the sparse grid.

Note that the increased order of a CC_E grid is not entirely wasted. It does mean that certain higher order monomials will be integrated exactly; but because some monomials of the same degree will not be integrated exactly, this extra accuracy does not affect the total precision or the asymptotic accuracy of the grids. So a CC_E grid may have a lower error than a CC_SE grid of the same level, but the error reduction is small, and too costly in terms of extra function evaluations.

6 Comparison of Accuracy

We reserve the word *precision* for measuring the polynomial precision of a 1d rule or multidimensional sparse grid; it is used when claiming that a rule can integrate exactly all monomials of total degree less than or equal to that limit.

By *accuracy* we may simply mean the collection of all monomials that a rule or sparse grid can integrate exactly.

We know that a CC_E sparse grid of level **l** will have precision **2 l + 1**. For a 2D sparse grid we can make an "accuracy plot", which fills in a box at each point (e_1, e_2) for which the sparse grid can exactly integrate $x^{e_1} y^{e_2}$. If the sparse grid is of level **l**, then every box on or below the "precision line" $x + y = 2\,l + 1$ must be filled in. If the sparse grid has "extra accuracy", then some boxes above this line may also be filled in. Such extra accuracy will slightly improve the results of approximate integration, but presumably, the points of extra accuracy come at the same cost as the necessary accuracy that lies on or below the precision line. Thus, if a sparse grid's accuracy plot shows a lot of extra accuracy above the precision line, this suggests that the grid may be inefficient.

The following images depict the accuracy plots of 2D sparse grids of levels 0 through 3, formed from a 1D Clenshaw Curtis rule of either standard or slow exponential growth.

The following images depict the expected accuracy of 2D sparse grids of levels 4, 5, and 6, formed from the CC_E family (on the left) and the CC_SE family (on the right). Note that the level 6 CC_E plot goes off the scale, reaching values of magnitude 65 on the first 2 columns and last two rows.

As the level increases, the plots of the CC_E sparse grids show symptoms of the "hyperbolic cross" phenomenon. On the other hand, the CC_SE sparse grids show a strong tendency not to stray too far above the precision line.

7 Comparison of Point Growth

A primary motivation for sparse grids is to achieve a desired level of polynomial precision for approximate integration while avoiding the exponential cost associated with a product grid rule. Since the approximate integral is computed as a weighted sum of function values, the "cost" of a quadrature procedure is usually taken as the number of function evaluations, and hence, equivalently, as the number of abscissas or points.

This is one reason that it is important to be able to determine the point count or order of a sparse grid, given its level, spatial dimension, and the underlying 1D family of quadrature rules from which it is constructed. For the CC_E and CC_SE families, it is actually possible to derive simple procedures to quickly evaluate the order. We will use such a procedure in order to produce order tables for both families.

The formula for the point count takes advantage of the nestedness of the CC_E and CC_SE families. We begin by noting that for the CC_E family, the order of the selected CC rule as a function of level has the form:

$$\text{order\_1d} = \{1, 3, 5, 9, 17, 33, \ldots, 2^l + 1, \ldots\}$$

Because of nestedness, we may then define and easily compute the vector **new_1d** to count the number of "new" points that occur for the first time at a given level:

$$\text{new\_1d} = \{1, 2, 2, 4, 8, 16, \ldots, 2^{l-1}, \ldots\}$$

Now to construct a sparse grid in dimension **M** and level **L**, Smolyak's formula tells us to combine product rules whose product levels lie in the limited range between $\max(0, L - M + 1)$ and L. When the underlying family is nested, and we are only interested in what points are included in the sparse grid, we can think of the sparse grid as being formed from the *entire range* of product rule levels, 0 to L. The advantage to thinking this way is that we can now logically assign each point in the final grid a time of "first appearance" in the grid, by starting with product rule level 0. For each point in the grid, there is a minimum product rule level at which it appears, and for that level it appears in exactly one product grid. This means that one way

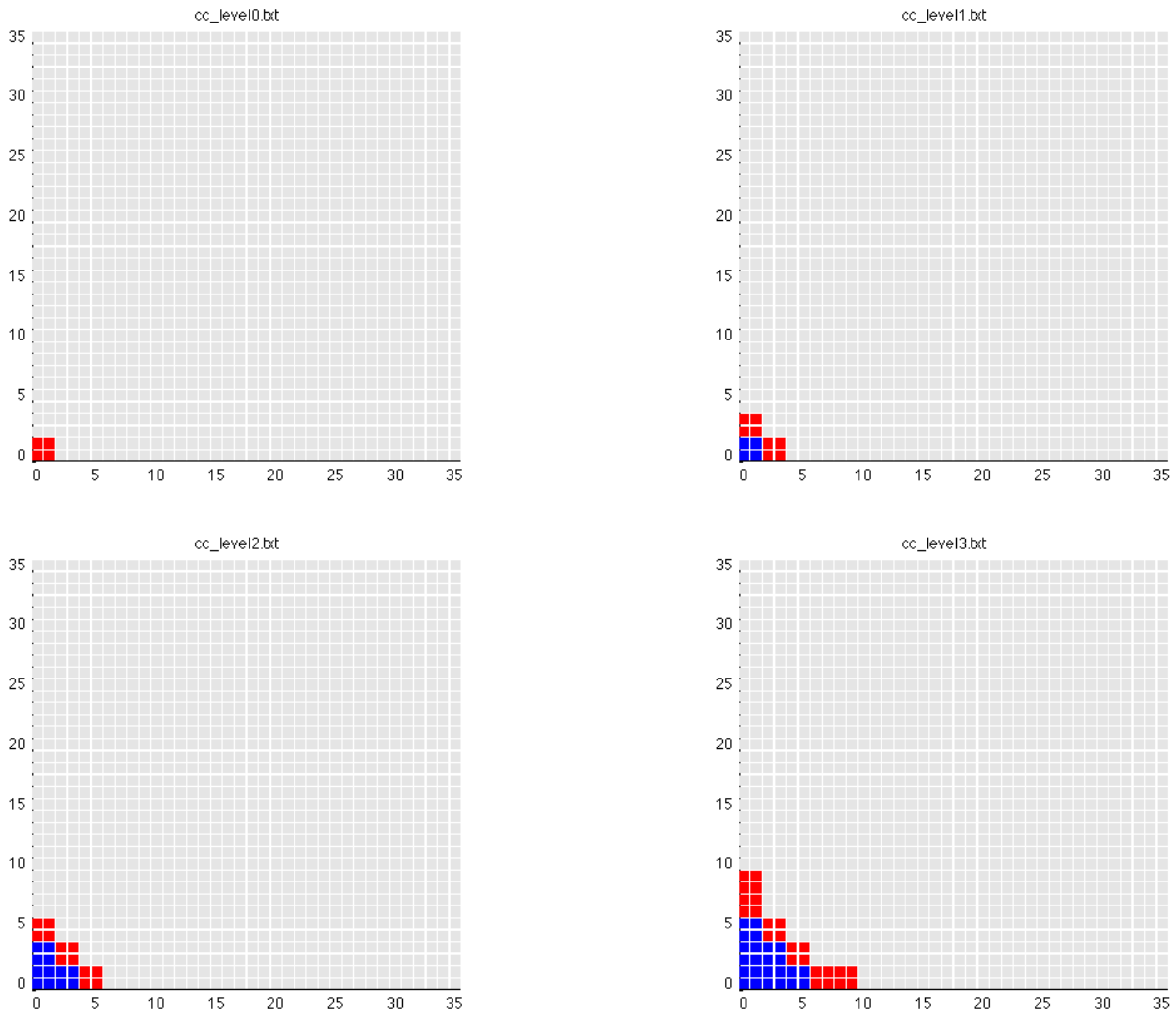

Figure 3: Accuracy tables for 2D CC_E sparse grids, levels 0 through 3.
For these levels, the CC_SE sparse grids have identical behavior.

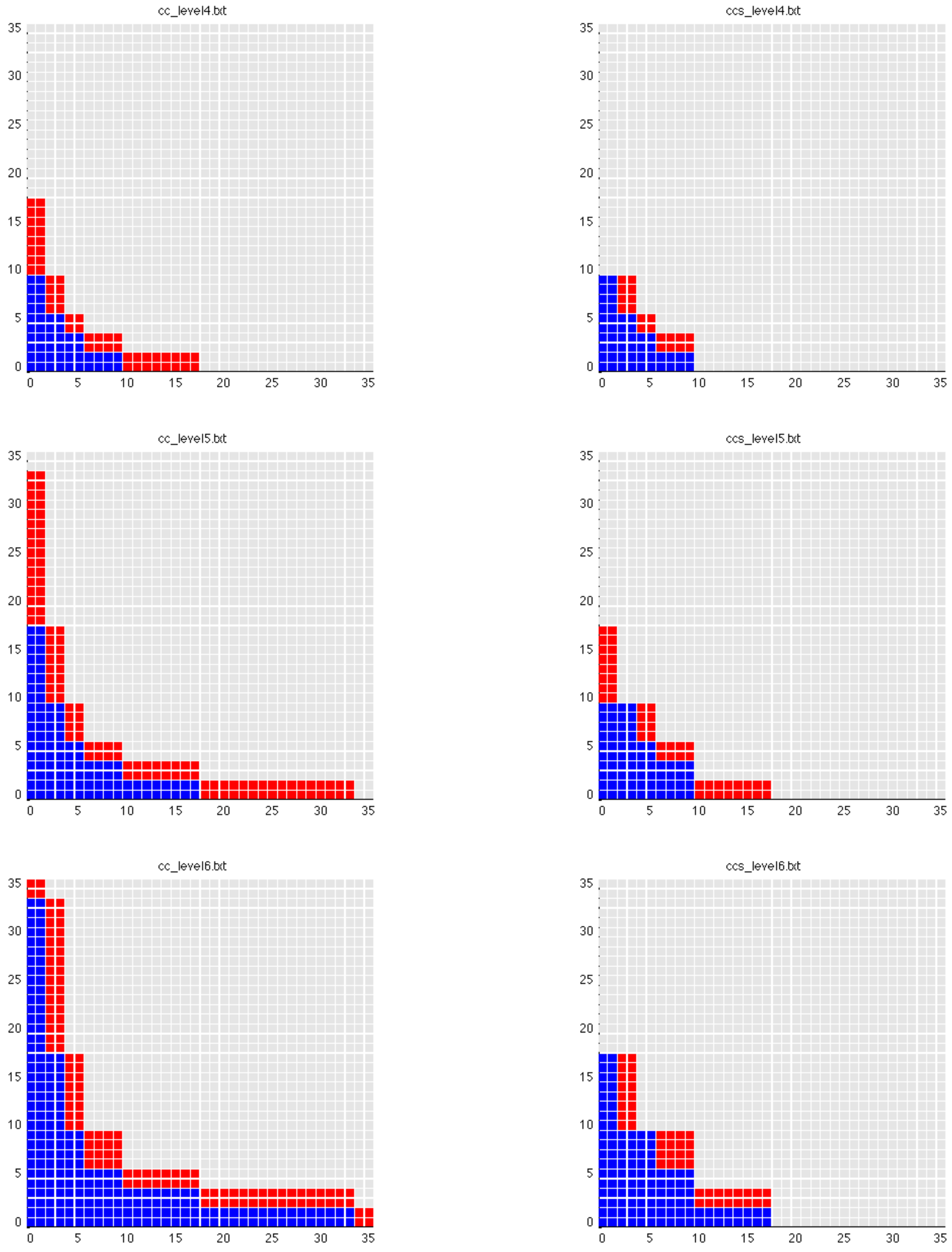

Figure 4: Accuracy tables for 2D CC_E and CC_SE sparse grids, levels 4, 5 and 6. Some of the CC_E data for level 6 has exceeded the limits of the plot.

L	New						
5	16	16					
4	8	8	16				
3	4	4	8	8			
2	2	2	4	4	8		
1	2	2	4	4	8	16	
0	1	1	2	2	4	8	16
	New	1	2	2	4	8	16
	L	0	1	2	3	4	5

Table 1: Counting the points in a 2D CC_E sparse grid.

to count all the points in the sparse grid is to count only the first appearances of points in the component product grids.

But this turns out to be easy. Any component product rule is defined by its level vector, **level_1d**, which in turn gives us a decomposition of that product grid into its 1D factors. The number of points that make their first appearance in this grid is simply

$$\text{new}(\text{level\_1d}) = \prod_{i=1}^{M} \text{new\_1d}(\text{level\_1d}(i))$$

and therefore the number of points in the sparse grid is

$$\text{order}(L, M) = \sum_{l=1}^{L} \sum_{|level\_1d|=l} \prod_{i=1}^{M} \text{new\_1d}(\text{level\_1d}(i))$$

which is the sum, over all product rule levels **l** from 0 to **L**, of the sum over all product grids of level exactly **l**, of the number of points which first appear in that product grid.

For a 2D grid, the result can easily be worked out by performing what amounts to a convolution on the **new_1d** vector. For instance, the order of the sparse grid of level 3, dimension 2 is

$$\begin{aligned} \text{order}(3,2) = & \, 1 \\ & + (2*1+1*2) \\ & + (2*1+2*2+1*2) \\ & + (4*1+2*2+2*2+1*4) \\ & = 29 \end{aligned}$$

For 2D, the procedure can also be visualized in a tabular array, as in Table 7

This table shows the number of points that have their "first appearance" in each of the product rules that is used to build a CC_E sparse grid in dimension 2 of level 5. Totalling the values inside the box produces 145, the number of points in the sparse grid.

For the general case, the count is easy to program, as long as we have some procedure for generating all possible level vectors that have a given level. Here is the body of a C++ function that computes **order**, the order of a sparse grid. It is to be understood that the function **comp_next** produces, one at a time, the level vectors **level_1d** that sum to **l**.

```
new_1d = new int[level+1];
new_1d[0] = 1;
new_1d[1] = 2;
j = 1;
```

```
for ( l = 2; l <= level; l++ )
{
  j = j * 2;
  new_1d[l] = j;
}
level_1d = new int[m];
order = 0;
for ( l = 0; l <= level; l++ )
{
  more = false;
  for ( ; ;)
  {
    comp_next ( l, m, level_1d, &more );
    v = 1;
    for ( dim = 0; dim < m; dim++ )
    {
      v = v * new_1d[level_1d[dim]];
    }
    order = order + v;
    if ( !more )
    {
      break;
    }
  }
}
```

Novak and Ritter [3] computed and displayed the point counts for sparse grids based on the standard Clenshaw Curtis family, for levels 1 through 8, and dimensions 5, 10, 15, 20 and 25. Here, we present results for levels 0 through 10, and dimensions 1 through 10. Where they overlap, the tables agree with Novak and Ritter.

The same computation of order can be done for sparse grids based on the CC_SE family. Now, however, the 1D order as a function of level has the form:

$$\text{order\_1d} = \{1, 3, 5, 9, 9, 17, 17, 17, 17, 33, 33, \ldots\}$$

and the vector **new_1d** begins:

$$\text{new\_1d} = \{1, 2, 2, 4, 0, 8, 0, 0, 0, 16, 0, \ldots\}$$

Again, we can make a tabular array for a 2D case, as in Table 7.

Summing up the entries within the box, we compute the number of points in a CC_SE sparse grid for dimension 2, level 5, is 81.

The code to compute the order for a CC_SE sparse grid is quite similar to that for a CC_E, and in fact, only the definition of **new_1d** needs to be revised:

```
new_1d = new int[level_max+1];
new_1d[0] = 1;
new_1d[1] = 2;
p = 3;
o = 3;
for ( l = 2; l <= level; l++ )
{
```

DIM:	1	2	3	4	5
CC_E LEVEL					
0	1	1	1	1	1
1	3	5	7	9	11
2	5	13	25	41	61
3	9	29	69	137	241
4	17	65	177	401	801
5	33	145	441	1,105	2,433
6	65	321	1,073	2,929	6,993
7	129	705	2,561	7,537	19,313
8	257	1,537	6,017	18,945	51,713
9	513	3,329	13,953	46,721	135,073
10	1,025	7,169	32,001	113,409	345,665

DIM:	1	2	3	4	5
CC_SE LEVEL					
0	1	1	1	1	1
1	3	5	7	9	11
2	5	13	25	41	61
3	9	29	69	137	241
4	9	49	153	369	761
5	17	81	297	849	2,033
6	17	129	545	1,777	4,833
7	17	161	881	3,377	10,433
8	17	225	1,361	5,953	20,753
9	33	257	1,953	9,857	38,593
10	33	385	2,721	15,361	67,425

DIM:	1	2	3	4	5
LEVEL					
0	1.00	1.00	1.00	1.00	1.00
1	1.00	1.00	1.00	1.00	1.00
2	1.00	1.00	1.00	1.00	1.00
3	1.00	1.00	1.00	1.00	1.00
4	1.88	1.32	1.15	1.08	1.05
5	1.94	1.79	1.48	1.30	1.20
6	3.82	2.48	1.96	1.64	1.45
7	7.58	4.37	2.90	2.23	1.85
8	15.11	6.83	4.42	3.18	2.49
9	15.54	12.95	7.14	4.73	3.49
10	31.00	18.62	11.76	7.38	5.12

Table 2: CC_E Order, CC_SE Order, CC_E Order / CC_SE Order, dimensions 1 to 5. The improvement increases with level, decreases with dimension;

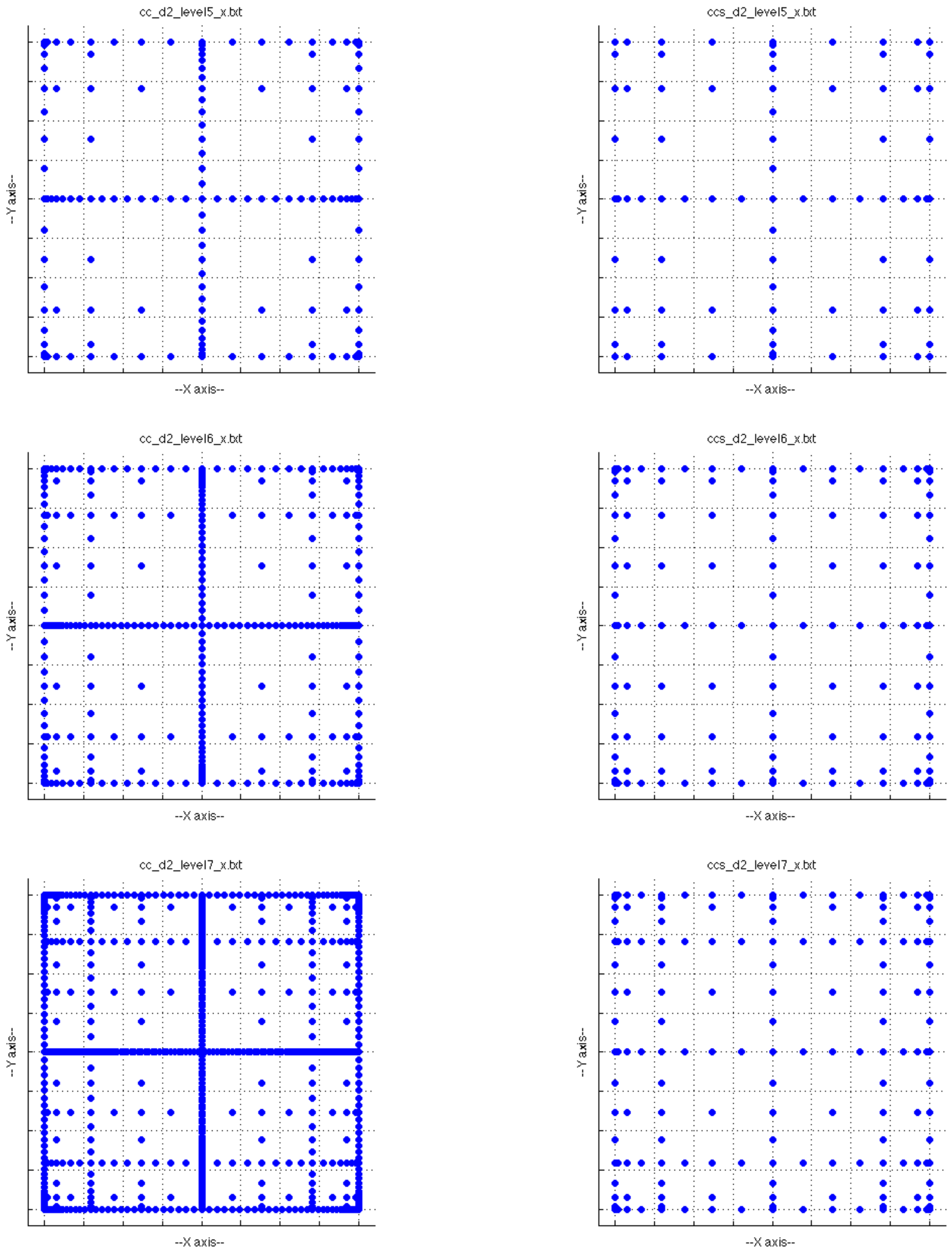

Figure 5: CC_E and CC_SE 2D sparse grids, levels 5, 6 and 7.

L	New						
5	8	8					
4	0	0	0				
3	4	4	8	8			
2	2	2	4	4	8		
1	2	2	4	4	8	0	
0	1	1	2	2	4	0	8
	New	1	2	2	4	0	8
	L	0	1	2	3	4	5

Table 3: Counting the points in a 2D CC_SE sparse grid.

```
    p = 2 * l + 1;
    if ( o < p )
    {
      new_1d[l] = o - 1;
      o = 2 * o - 1;
    }
    else
    {
      new_1d[l] = 0;
    }
  }
```

Again, we compute the number of points in sparse grids of levels 0 through 10, and dimensions 1 through 10. Because we are using the more economical CC_SE family, however, there are some dramatic differences when compared to the tables for the CC_E sparse grid. In particular, the values in column 1 decrease drastically.

In general, since the CCS_E and CC_E select the same 1D CC rules until level 4, we do not expect to see differences in the CC_SE table until that point. At level 4 and beyond, the difference is most noticeable for low dimensional grids. Starting at any fixed location in the table, the relative improvement from using the CC_SE family becomes stronger if we decrease the dimension or increase the level.

Notice that by the time we reach dimension 10, even the CC_SE sparse grid exceeds 1,000,000 points at level 8. If we think of a million points as roughly the maximum order we would care to consider, then for dimension 10 and beyond, the only advantage to using the CC_SE family will lie in the levels 4 through 8 at best. To emphasize this point, we now compare the CC_E and CC_SE point counts for dimensions 5, 10, 15, 20 and 25, for levels 0 through 8:

8 Conclusion

At first glance, it might seem that a comparison of Figure 1 and Figure 2 should have completely decided the question of whether the CC_SE family is superior to the CC_E family for sparse grids. The CC_SE family has essentially linear order growth while reliably achieving the 1D precision needed to guarantee the corresponding precision of the sparse grid.

However, the fact that the CC_SE family is exponentially better than the CC_E family at 1D levels of 10 or more matters much less when we realize that we will rarely be interested in such high order sparse grids, particularly for high dimensions, and that the improvement offered by the CC_SE family will really only be noticeable for sparse grids of low dimension (less than 10).

So the first conclusion of this investigation is that the use of the CC_SE family, which would seem to

DIM:	6	7	8	9	10
CC_E LEVEL					
0	1	1	1	1	1
1	13	15	17	19	21
2	85	113	145	181	221
3	389	589	849	1,177	1,581
4	1,457	2,465	3,937	6,001	8,801
5	4,865	9,017	15,713	26,017	41,265
6	15,121	30,241	56,737	100,897	171,425
7	44,689	95,441	190,881	361,249	652,065
8	127,105	287,745	609,025	1,218,049	2,320,385
9	350,657	836,769	1,863,937	3,918,273	7,836,545
10	943,553	2,362,881	5,515,265	12,133,761	25,370,753

DIM:	6	7	8	9	10
CC_SE LEVEL					
0	1	1	1	1	1
1	13	15	17	19	21
2	85	113	145	181	221
3	389	589	849	1,177	1,581
4	1,409	2,409	3,873	5,929	8,721
5	4,289	8,233	14,689	24,721	39,665
6	11,473	24,529	48,289	88,945	155,105
7	27,697	65,537	141,601	284,209	536,705
8	61,345	159,953	377,729	823,057	1,677,665
9	126,401	361,665	930,049	2,192,865	4,810,625
10	244,289	765,089	2,136,577	5,4363,21	12,803,073

Table 4: Order as a function of level and dimension, CC_E versuse CC_SE family, dimensions 6 to 10.

DIM=5:	CC_E	CC_SE	CC_E/CC_SE
LEVEL			
0	1	1	1.00
1	11	11	1.00
2	61	61	1.00
3	241	241	1.00
4	801	761	1.05
5	2,433	2,033	1.20
6	6,993	4,833	1.45
7	19,313	10,433	1.85
8	51,713	20,753	2.49

DIM=10:	CC_E	CC_SE	CC_E/CC_SE
LEVEL			
0	1	1	1.00
1	21	21	1.00
2	221	221	1.00
3	1,581	1,581	1.00
4	8,801	8,721	1.01
5	41,265	39,665	1.04
6	171,425	155,105	1.11
7	652,065	536,705	1.21
8	2,320,385	1,677,665	1.38

DIM=15:	CC_E	CC_SE	CC_E/CC_SE
LEVEL			
0	1	1	1.00
1	31	31	1.00
2	481	481	1.00
3	5,021	5,021	1.00
4	40,001	39,881	1.00
5	261,497	257,897	1.01
6	1,471,297	1,416,817	1.04
7	7,367,857	6,808,417	1.08
8	33,647,617	29,232,337	1.15

Table 5: Comparisons of order at dimensions 5, 10, 15.

DIM=20:	CC_E	CC_SE	CC_E/CC_SE
LEVEL			
0	1	1	1.00
1	41	41	1.00
2	841	841	1.00
3	11,561	11,561	1.00
4	120,401	120,241	1.00
5	1,018,129	1,011,729	1.01
6	7,314,609	7,185,969	1.02
7	46,106,289	44,365,169	1.04
8	261,163,009	243,234,369	1.07
DIM=25:	CC_E	CC_SE	CC_E/CC_SE
LEVEL			
0	1	1	1.00
1	51	51	1.00
2	1,301	1,301	1.00
3	22,201	22,201	1.00
4	286,001	285,801	1.00
5	2,976,161	2,966,161	1.00
6	26,139,361	25,888,561	1.01
7	199,876,961	195,656,561	1.02
8	1,361,884,161	1,308,121,361	1.04

Table 6: Comparisons of order at dimensions 20, 25.

offer the possibility of an enormous decrease in the order of sparse grids, only provides a moderate decrease in order, and this over a limited subset of the practical range of sparse grids.

This observation is not entirely negative, however. First of all, we see that the CC_SE family can be substituted for the CC_E with little effort, and hence the improved performance over low dimensional sparse grids can be obtained easily.

Of more importance, however, is the implication of the fact that substituting an essentially linear growth rule for the original exponential growth rule did not make much difference.

Since replacing the CC_E family by a family with linear growth did not appreciably decrease the point growth for higher dimensions ($10 < M$), this suggests that the exponential growth of the CC_E sparse grids does not actually impose a significant inflation of the order of sparse grids, particularly for sparse grids that lie within the practical range of dimension, level and order.

Hence, it is possible that the benefits of nesting may make CC_E sparse grids as efficient as those formed from more accurate but non-nested Gaussian rules.

This observation, in turn, suggest the usefulness of a further study, in which sparse grids based on the CC_E or CC_SE family are compared in efficiency with those based on other quadrature families, in particular the nested Gauss-Patterson family, and non-nested Gauss-Legendre. Such a comparison would be made by constructing sparse grids based on each of these families, with the property that each achieves the same precision, and then comparing the orders.

References

[1] John Burkardt, SGMGA: Sparse Grid Mixed Growth Anisotropic Rules, http://people.sc.fsu.edu/~burkardt/cpp_src/sgmga/sgmga.html.

[2] Erich Novak, Klaus Ritter, High dimensional integration of smooth functions over cubes, Numerische Mathematik, Volume 75, Number 1, November 1996, pages 79-97.

[3] Erich Novak, Klaus Ritter, Richard Schmitt, Achim Steinbauer, Simple cubature formulas with high polynomial exactness, Constructive Approximation, Volume 15, Number 4, December 1999, pages 499-522.

[4] Knut Petras, Smolyak cubature of given polynomial degree with few nodes for increasing dimension, Numerische Mathematik, Volume 93, Number 4, February 2003, pages 729-753.

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

[6] Lloyd Trefethen, Is Gauss quadrature better than Clenshaw-Curtis?, SIAM Review, Volume 50, Number 1, March 2008, pages 67-87.