Slow Exponential Growth for Gauss Patterson Sparse Grids

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

April 4, 2014

Abstract

When Gauss Patterson rules are used to form a sparse grid, the indexing of the underlying 1D family is crucial. It would seem natural to use the indexing that preserves nesting, but this leads to exponential growth in the order of the 1D rules. If the aim is to efficiently construct a family of sparse grids, indexed to achieve a linearly increasing level of precision, then it is possible to preserve the use of nested Gauss Patterson rules while sharply cutting the order growth. This is done by delaying the introduction of the next rule until the precision requirement of the sparse grid demands it.

1 Introduction

The sparse grid construction of Smolyak [8] 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, which forms a nested set. Nestedness is an advantage in the construction of sparse grids, and can help to control the growth in the number of abscissas.

There is a natural alternative to the Clenshaw Curtis rule, namely the rules formed from the Gauss Patterson procedure. These rules can also be generated as a nested family, and their precision is roughly double that of Clenshaw Curtis rules. However, the growth rate for the nested family of Gauss Patterson rules is roughly twice that of the Clenshaw Curtis rules. Therefore, as we shall see, if the typical procedure is used to produce sparse grids, the precision advantage of the Gauss Patterson rules must be balanced against its higher growth rate.

Moreover, whether the Clenshaw Curtis or Gauss Patterson rules are used in the standard procedure for generating a sparse grid, the nestedness of the families results in an exponential growth in the order of the 1D rules, and a corresponding cost in the resulting sparse grid.

The exponential growth of the 1D rules means that some of the higher order rules have a precision that is far in excess of what is needed for the requirements of the resulting sparse grid.

Thus, there is reason to search for alternative methods of selecting the underlying 1D factor rules, so that the precision requirement is met in a more efficient way. Such an alternative, here called *slow exponential growth*, has already been investigated for the case where Clenshaw Curtis rules are used [3], and it is the intent of this document to explore the case in which Gauss Patterson rules are used. It is expected that the alternative procedure can take advantage of the higher precision of the Gauss Patterson rules to produce sparse grids that attain a desired precision, at the cost of a point count that is well below that required for a standard Gauss Patterson sparse grid or for a slow exponential growth Clenshaw Curtis sparse grid.

2 Gauss Patterson Quadrature Rules

The quadrature problem in 1D considers the integral of a function $f(x)$ multiplied by a weight function $g(x)$ over an interval (a, b). One way to estimate this quantity is with a quadrature, that is, an n-fold weighted sum of the function evaluated at specific abscissas.

For a given interval and weight function, if the abscisass x and weights w may be freely chosen, (and if $g(x)$ satisfies some mild conditions of positivity and integrability) then there is a standard procedure to produce a Gauss quadrature formula of the form

$$\int_a^b f(x)\, g(x)\, dx \approx \sum_{i=1}^{n} w_i\, f(x_i)\,.$$

This formula obtains the highest possible polynomial precision of any rule using n points. This polynomial precision is $2n-1$, and hence the Gauss quadrature formula will produce the exact value of the integral of any function $f(x)$ which is a polynomial of degree $2n-1$ or less.

However, it is a common desire, when dealing with the quadrature problem, to desire the ability to produce a related *sequence* of estimates, using an indexed family of formulas of increasing precision. In this way, the difference between successive estimates can be regarded as an approximation to the integration error, and hence can be a guide to whether the most recent estimate is satisfactory enough that the iterative process can be terminated.

In such a case, it is a great advantage if the collection of quadrature rules is *nested*, that is, that all the abscissas of one rule are included in the next rule. A typical nested family might involve a sequence of rules of orders 1, 3, 7, 15, and so on. Here, a rule of order n would be followed by a rule of order $2n+1$ which interleaves new and old points in a natural way.

However, if a nested family is desired, the successive rules cannot be produce by the standard Gaussian procedure. The Gaussian procedure requires a free choice for all the abscissas, whereas the nested family constrains the choice of some abscissas to agree with those of the previous rule.

In an effort to combine the advantages of nestedness with the high precision of Gaussian rules, Patterson [6] devised a modified Gaussian procedure, which produces a rule of the highest possible precision under the constraint of incorporating the abscissas of the previous rule. If our interval is $[-1, +1]$, and our weight function is $g(x) = 1$, and we start with the midpoint rule, then the procedure produces the classical Gauss Patterson family of rules. (Variations of this family can be produced by starting from a different initial rule, such as a 5 point Gauss rule.)

In this document, for convenience, we will refer to the classical Gauss Patterson rule simply as the Gauss Patterson rule, and we will occasionally also use the abbreviation "GP".

The Gauss Patterson family starts with the midpoint rule of precision 1, followed by the 3 point Gauss rule of precision 5. The next rule will be of order 7. A standard Gauss procedure would produce a rule of precision 13, but it would not be nested. The Gauss Patterson procedure imposes 3 nestedness constraints, which mean that the resulting rule will only have precision of 10. If we index the rules by i, assigning the midpoint rule the index of 0, then the order of the Gauss Patterson rules is

$$o(k) = 2^{k+1} - 1$$

The precision of a rule of a given index can be determined by considering the maximum possible precision (twice the order minus 1) and subtracting the number of constraints associated with the abscissas of the previous rule, and adding 1 because the rule, weight function, and region are symmetric and the order is odd:

$$\begin{aligned}
p(k) =& 2 * o(k) - 1 - o(k-1) + 1 \\
=& (2 * (2^{k+1} - 1) - 1) - 1 - (2^k - 1) + 1 \\
=& 2 * 2 * 2^k - 3 - 2^k + 1 + 1 \\
=& 3 * 2^k - 1.
\end{aligned}$$

This result holds for all k except the special case $k = 0$, when $p(k) = 1$.

Tables of the weights and abscissas for this Gauss Patterson family are available for indices k from 0 through 7, which correspond to orders 1 through 255 and precisions 1 through 383. Since, under the usual construction method, a sparse grid rule of level L requires the use of a 1D rule whose index is equal to L, this means that the GP family, indexed in this way, cannot be used to construct sparse grids beyond level 7.

3 Sparse Grids

The quadrature problem in M dimensions considers the integral of a function $f(x^M)$ multiplied by a weight function $g^M(x)$ over an M-dimensional domain $\mathcal{D}^{\mathcal{M}}$. If the domain $\mathcal{D}^{\mathcal{M}}$ is of a tractable type, one way to estimate the integral is by means of a quadrature rule, that is, an n-fold weighted sum of the function evaluated at specific abscissas.

One very tractable type of domain $\mathcal{D}^{\mathcal{M}}$ is a *product domain*, in which we have

$$\mathcal{D}^{\mathcal{M}} = D_1 \times D_2 \times \ldots \times D_M$$

with each D_i being a 1-dimensional domain. It may also be the case that our weight function $g^M(x^M)$ is *separable*, that is, that it satisfies the condition:

$$\}^{\mathcal{M}}(\S_\infty, \S_\in, ...\S_\Updownarrow) = g_1(x_1) \times g_2(x_2) \times \ldots \times g_M(x_M)$$

In that case, we can think of our quadrature problem in terms of iterated 1D integrals:

$$\int_{\mathcal{D}^{\mathcal{M}}} f(x^m)g(x^m)dx^m = \int_{D_M} \cdots \int_{D_2} \int_{D_1} f(x^M)g_1(x_1)dx_1 g_2(x_2)dx_2 \ldots g_m(x_m)dx_m$$

A sparse grid is a particular kind of quadrature rule, which is suitable for estimating integrals of the above form, in which the region is a product region and the weight function is separable. For our discussion of the Gauss Patterson rule, the region will simply be the unit hypercube $[-1, +1]^M$ and the weight function will be $g(x^M) = 1$, which is trivially separable.

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 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 $1 - M$ 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.

4 "GP_E": Gauss Patterson Exponential Growth Family

To construct a sparse grid using the Smolyak procedure, we need a family of 1D quadrature rules of increasing precision. We wish to consider the consequences of choosing the Gauss Patterson family for this purpose.

In order to simplify later discussion, we need to make a small distinction between the *index* of a Gauss Patterson rule, that is, the index k in the previous section, and the *1D level*, symbolized by j, which is going to keep track of how the sparse grid procedure selects elements of the family. In the present section, we will have k and j the same, but we need to prepare for the subsequent exploration in which a distinction must be made.

Let us suppose, then, that we construct a sparse grid in dimension M using the Gauss Patterson family in the standard way. Then, as the 1D level j is incremented, the order of the 1D rules will roughly double. This means that order of the 1D rules is an exponential function of the 1D level. The rapid growth in order means that for relatively low values of j we get an excessive number of points. The 1D rule will, of course, produce a correspondingly high *accuracy*, but this will not be reflected in a useful increase in the *precision* of the sparse grid itself.

Here we tabulate the number of points in a GP_E sparse grid, for dimensions M from 1 to 10 and sparse grid levels L from 0 to 7.

DIM:	1	2	3	4	5
GP_E LEVEL					
0	1	1	1	1	1
1	3	5	7	9	11
2	7	17	31	49	71
3	15	49	111	209	351
4	31	129	351	769	1,471
5	63	321	1,023	2,561	5,503
6	127	769	2,815	7,937	18,943
7	255	1,793	7,423	23,297	61,183

DIM:	6	7	8	9	10
GP_E LEVEL					
0	1	1	1	1	1
1	13	15	17	19	21
2	97	127	161	199	241
3	545	799	1,121	1,519	2,001
4	2,561	4,159	6,401	9,439	13,441
5	10,625	18,943	31,745	50,623	77,505
6	40,193	78,079	141,569	242,815	397,825
7	141,569	297,727	580,865	1,066,495	1,862,145

Notice, in particular, the behavior of the 1 dimensional rules. The rapid growth in the order is determined by the nesting requirement. If we are trying to estimate the error in an estimated integral, then the only way to produce such an estimate is to proceed from one level to the next, even though this requires a doubling of the number of points each time.

Suppose, however, that we have a weaker requirement, namely, that we wish to be able to ensure that our estimate has at least precision p. Then for a p value of 0 or 1, the 1 point rule will do. For p values of 2, 3, 4 or 5, the 3 point rule will do. Similarly, for p up to 13, the 7 point rule will do. In the next section, we will see how weakening our expectations on the rule will allow us to produce a family of sparse grids with slower growth in order, and with a predictable and moderated growth in precision.

5 "GP_SE": Gauss Patterson Slow Exponential Growth Family

When we used our family of 1D Gauss Patterson rules, we specified that for 1D level index j we should select the rule of index k, that is, for the next level, we simply supplied the next rule in the family. This guarantees an increase in the precision of the 1D rule, of course, so that each successive sparse grid in turn satisfies a higher precision requirement.

However, if we start with the precision requirement of the sparse grid, we can instead ask the question *what is the lowest index k which will satisfy the precision requirement of the current sparse grid?*. So let us assume we have a sparse grid of level L. It is common to desire the sparse grid to have precision $2L + 1$, and this can be guaranteed, in fact, if each 1D factor of index j has precision at least $2j + 1$.

The 1D GP family far exceeds this requirement. The sparse grids of level 0, 1, 2 and 3 achieve precisions of 1, 3, 5 and 7, for instance, but the underlying GP rules have 1D precisions of 1, 3, 7, and 15, and the disparity between the 1D and multidimensional precisions gets worse with increasing level.

We can expect significant savings in point count, therefore, if we define a new 1D family of rules, named "GP_SE". For this family, the indexing is determined by specifying that the rule of sparse grid 1D level j will be that Gauss Patterson rule of lowest index k which has precision $2j + 1$ or more. Now our first 6 precision requirements are { 1, 3, 5, 7, 9, 11 }. The GP rules of order 1, 3 and 7 have precisions 1, 5, and 11 respectively. Therefore, we can satisfy our initial precision requirements by repeating the 3 point rule once, and the 7 point rule twice, so that the initial sequence of orders in the GP_SE family will be { 1, 3, 3, 7, 7, 7 }.

The effect of this change will be most noticeable in the lower dimensional sparse grids, particularly the special case of a 1D sparse grid. If we go to higher dimensions, the improvement becomes obvious only as the level increases. Therefore, if we make tables of the point counts for the GP_E and GP_SE sparse grids, the relative improvement caused by using GP_SE increases reading down any column (fixed dimension, increasing level), and decreases reading across a row (increasing dimension, fixed level). However, the effect is still substantial for dimensions on the order of 10 and levels as low as 3 and 4.

Here we tabulate the number of points in a GP_SE sparse grid, for dimensions 1 to 10 and sparse grid levels 0 to 10.

DIM:	1	2	3	4	5	6	7	8	9	10
GP_SE LEVEL										
0	1	1	1	1	1	1	1	1	1	1
1	3	5	7	9	11	13	15	17	19	21
2	3	9	19	33	51	73	99	129	163	201
3	7	17	39	81	151	257	407	609	871	1,201
4	7	33	87	193	391	737	1,303	2,177	3,463	5,281
5	7	33	135	385	903	1,889	3,655	6,657	11,527	19,105
6	15	65	207	641	1,743	4,161	8,975	17,921	33,679	60,225
7	15	97	399	1,217	3,343	8,481	19,855	43,137	87,823	169,185
8	15	97	495	1,985	6,223	16,929	42,031	97,153	211,087	434,145
9	15	161	751	2,881	10,063	30,689	83,247	206,465	477,327	1,041,185
10	15	161	1,135	4,929	17,103	53,729	154,927	411,265	1,014,159	2,347,809

Note that we have evaluated the GP_SE family up to level $L = 10$, and could have gone much further, whereas the GP_E family stops at level $L = 7$ and if it were desired to extend the GP_E table further, every increase in level would require the computation of a new 1D Gauss Patterson rule of unusually high order.

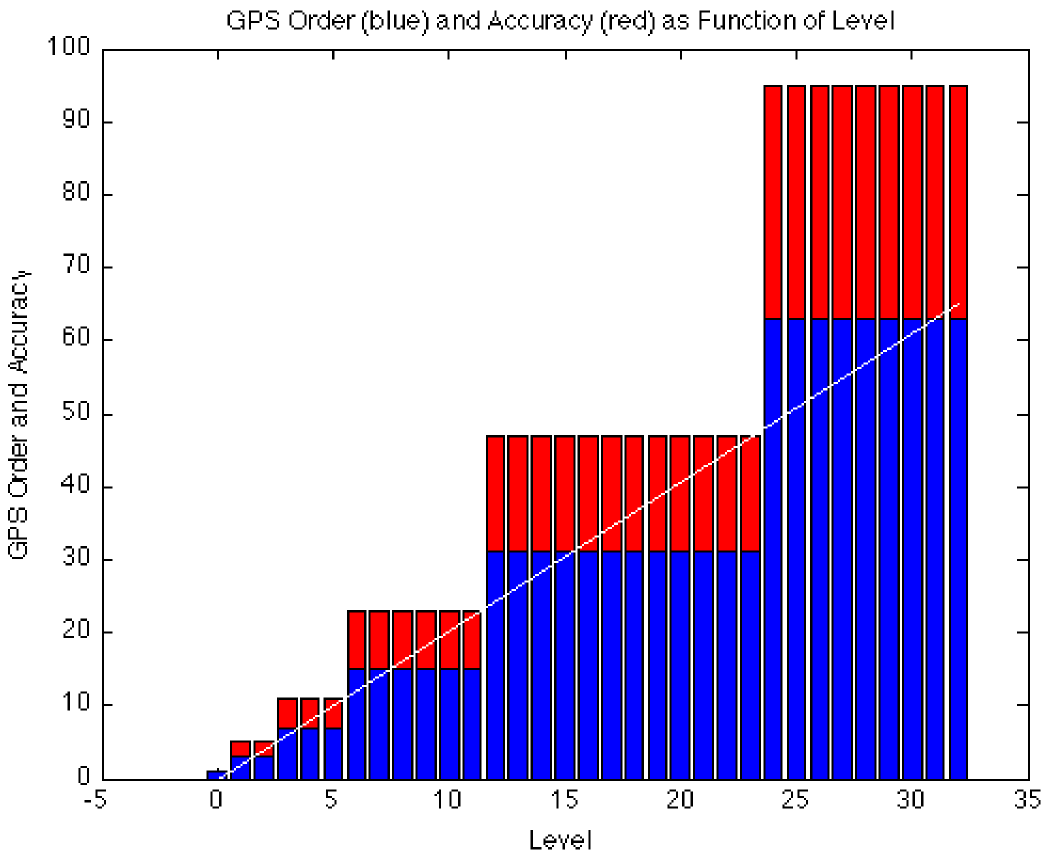


Figure 1: **"Slow" Nested GP_SE Family**
The slow growth GP_SE family has an approximately linear growth in the order and precision.
At level 10, the order is 15 and precision is 23.
The white line indicates the precision requirement $\mathbf{p} = 2 * \mathbf{L} + 1$.

6 Precision of Sparse Grids Using the GP_E and GP_SE Families

Novak and Ritter [4] have shown that a sparse grid of level L, constructed in the usual way, and using a family of 1D rules indexed so that rule j has precision $2j+1$, will have precision $2\,L+1$. 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 GP factors grows exponentially, is the anomaly that first suggested that there might be a more economical way of constructing sparse grids from a "slow growth" version of the GP_E family.

Since the GP_E and GP_SE families are identical for levels 0 and 1, it is only when a sparse grid reaches level 2 that we begin so see any differences in total order, corresponding to the use of the more economical GP_SE family.

It should be clear from the definition of the GP_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 [4] that a sparse grid constructed from the GP_SE family will also achieve the precision $2\,L+1$ that can be reached with the GP rule. In other words, the higher order rules used by the underlying GP family are not necessary to achieve the precision requirement associated with the sparse grid.

Note that the use of higher order GP rules in the GP_E family is not actually wasted - there is a benefit. 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 improve the precision of the rules. So a GP_E sparse grid may achieve a lower error than a GP_SE sparse grid of the same level when integrating certain functions, but the error reduction is small, and too costly in terms of extra function evaluations.

A more important point that favors the GP_E family over the GP_SE family arises when a sequence of calculations of increasing level are being performed. Each time the level of the GP_E rule is increased, it is guaranteed that the order will increase, and that new monomials of higher degree will be integrated exactly. However, for the GP_SE family, it is entirely possible, especially in low dimensions, that two successive sparse grids will actually be identical. In this case, a user comparing the resulting integral estimates might mistakenly conclude that convergence had been achieved.

7 Comparison of Accuracy

When we refer to the *accuracy* of a quadrature rule, we may simply mean the collection of all the monomials that the rule can integrate exactly.

We reserve the word *precision* for measuring the polynomial precision of a rule; it is used when claiming that a rule can integrate exactly all monomials of total degree less than or equal to that limit. We know that both GP_E and GP_SE sparse grids of level L will have precision $2\,L+1$.

In the 1D case, accuracy and precision tend to be used to mean the same thing. Even here, it is possible to assign different meanings to the two words. The midpoint rule on the interval [-1,+1], for instance, has precision 1, because it can integrate all polynomials of degree 1 or less. Strictly speaking, however, the midpoint rule is accurate for infinitely many polynomials, namely, any combination of the constant monomial and monomials of odd degree. We don't mean to further confuse the issue here, but we want to suggest that "precision" will have a stronger meaning than "accuracy" in this discussion.

We will now display some plots which highlight the difference between precision and accuracy.

For a 2D sparse grid rule we can make an "accuracy plot", which fills in a box at each point (e_1, e_2) for which the rule 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 rule 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 rule's accuracy plot shows a lot of extra accuracy above the precision line, this suggests that the rule may be less efficient than an optimal quadrature rule.

Figures 2 and 3 depict side by side the accuracy plots of GP_E and GP_SE sparse grids of levels 0 through 5. As the level increases, the plots of the GP_E grids show symptoms of the "hyperbolic cross" phenomenon. On the other hand, the GP_SE rules show a strong tendency not to stray too far above the precision line.

Figure 4 shows the accuracy plots for a GP_SE sparse grid of levels 6 and 7.

8 Comparison of Point Growth

A primary motivation for using sparse grids is to achieve a desired level of 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 GP_E and GP_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 GP_E and GP_SE families. We begin by noting that, in the 1D case, the order of a GP rule as a function of level j has the form:

$$\text{order\_1d} = \{1, 3, 7, 15, 31, 65, \ldots, 2^{j+1} - 1, \ldots\}$$

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:

$$\text{new\_1d} = \{1, 2, 4, 8, 16, 32, \ldots, 2^j, \ldots\}$$

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 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.

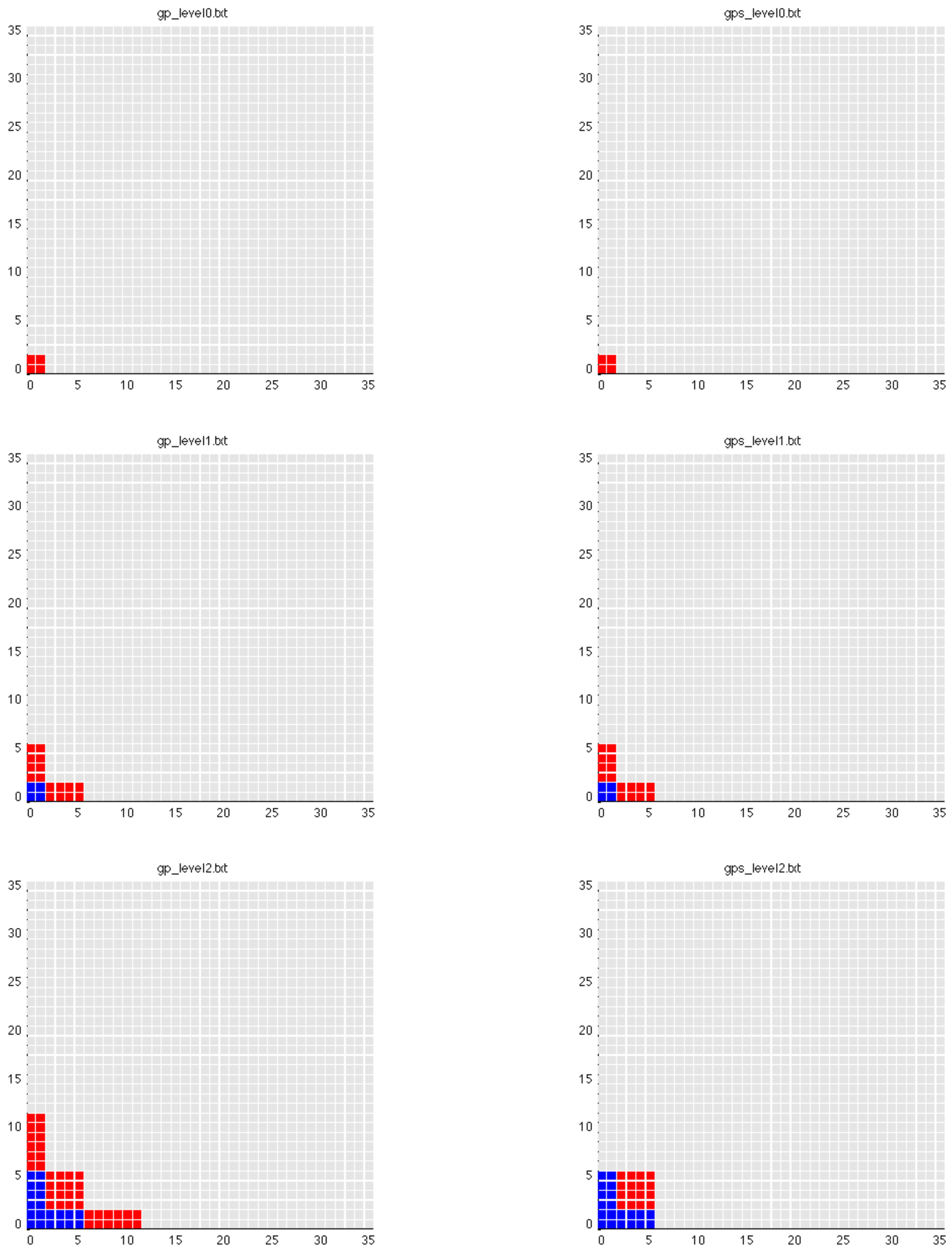


Figure 2: Accuracy tables for 2D GP_E and GP_SE sparse grids, levels 0, 1 and 2.

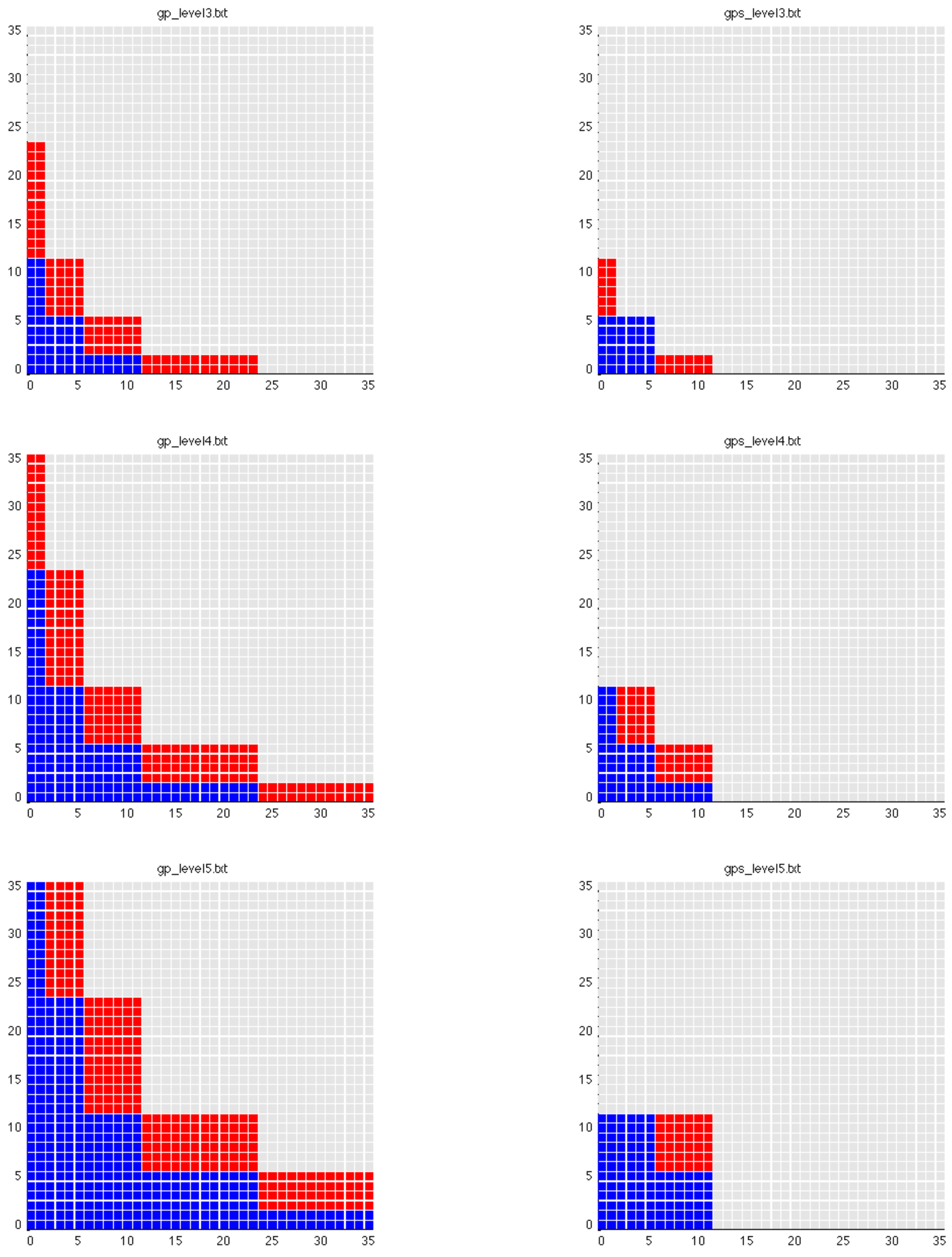
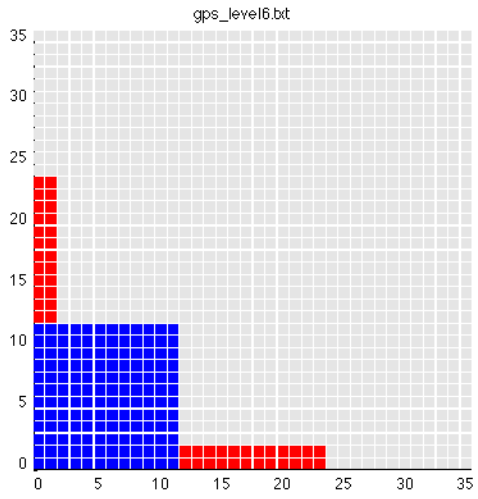


Figure 3: Accuracy tables for 2D GP_E and GP_SE sparse grids, levels 3, 4, and 5.



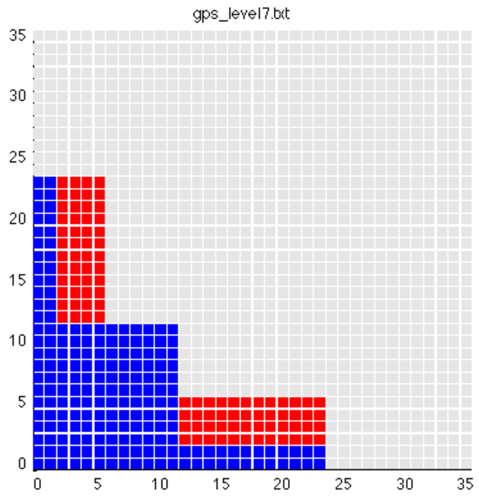


Figure 4: Accuracy tables for 2D GP_SE sparse grids, levels 6 and 7.

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 GP_E sparse grid of level 3, dimension 2 is

$$\begin{aligned}\text{order}(3,2) &= (1*1)\\ &+(2*1+1*2)\\ &+(4*1+2*2+1*4)\\ &+(8*1+4*2+2*4+1*8)\\ &=49\end{aligned}$$

The procedure can also be visualized in a tabular array, here shown for the GP_E sparse grid of level 5, dimension 2:

L	New						
5	32	32					
4	16	16	32				
3	8	8	16	32			
2	4	4	8	16	32		
1	2	2	4	8	16	32	
0	1	1	2	4	8	16	32
	New	1	2	4	8	16	32
	L	0	1	2	3	4	5

This table shows the number of points that have their "first appearance" in each of the product rules that is used to build a GP_E sparse grid in dimension 2 of level 5. The total is 1 * 1 + 2 * 2 + 3 * 4 + 4 * 8 + 5 * 16 + 6 * 32 = 321, 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;
  j = 1;
  for ( l = 1; 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;
      }
    }
  }
```

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

$$\text{order\_1d} = \{1, 3, 3, 7, 7, 7, 15, 15, 15, 15, 15, \ldots\}$$

and the vector **new_1d** begins:

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

Again, we can make a tabular array for a GP_SE grid of level 5 in dimension 2:

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

Summing up the entries within the box, we compute the number of points to be 33.

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

```
  order_1d[0] = 1
```

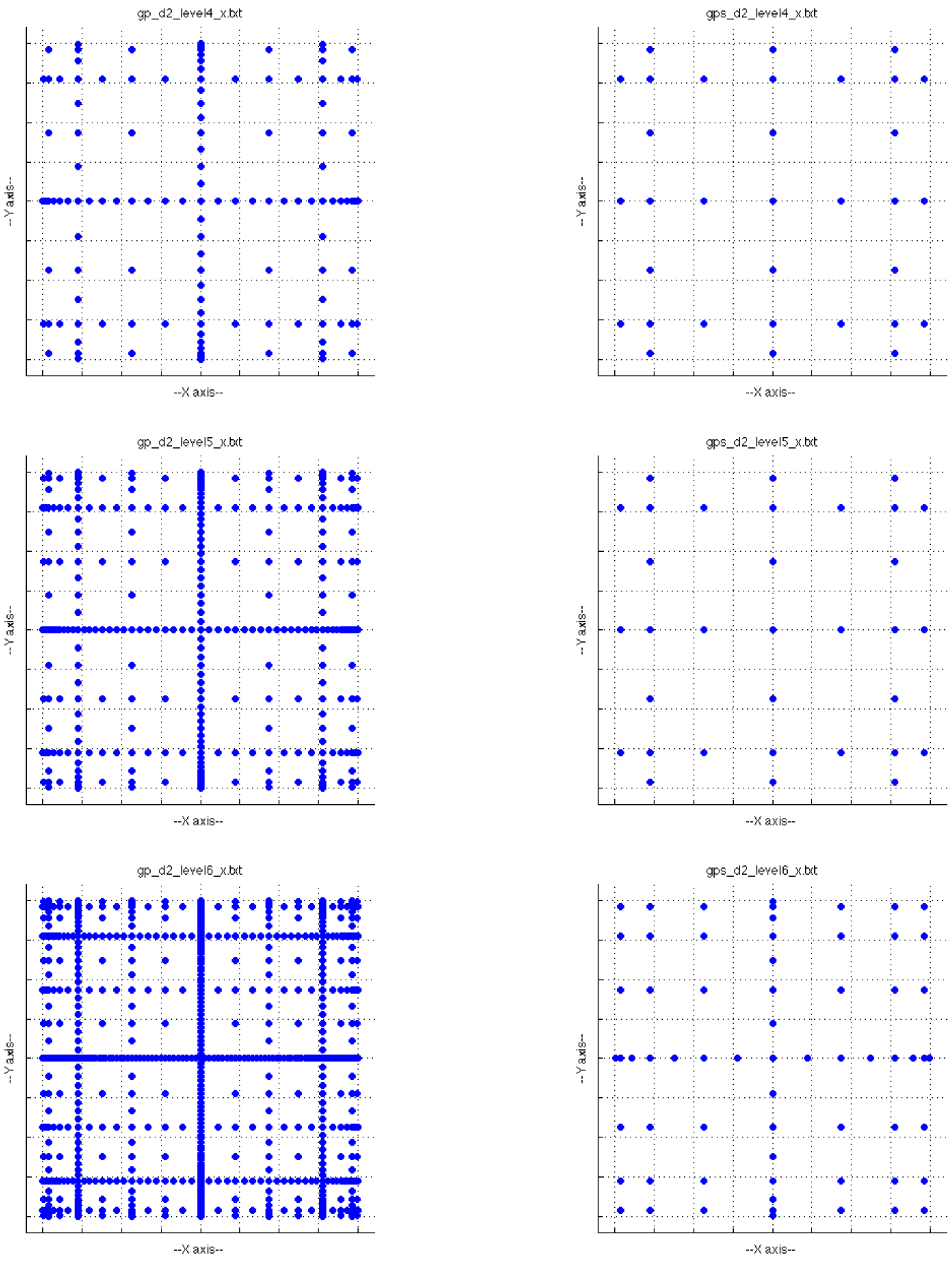


Figure 5: GP_E and GP_SE 2D sparse grids, levels 4, 5, and 6.

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	2.33	1.88	1.63	1.48	1.39
3	2.14	2.88	2.84	2.58	2.32
4	4.42	3.90	4.03	4.20	3.76
5	9.00	9.72	7.57	6.65	6.09
6	8.46	11.83	13.59	12.38	10.86
7	17.00	18.48	18.60	19.14	18.30

DIM:	6	7	8	9	10
LEVEL					
0	1.00	1.00	1.00	1.00	1.00
1	1.00	1.00	1.00	1.00	1.00
2	1.32	1.28	1.24	1.22	1.19
3	2.12	1.96	1.84	1.74	1.66
4	3.47	3.19	2.94	2.72	2.54
5	5.62	5.18	4.76	4.39	4.05
6	9.65	8.69	7.89	7.20	6.60
7	16.69	14.99	13.47	12.14	11.01

Table 1: GP_E Order / GP_SE Order, dimensions 1 to 10.
The improvement generally increases with level, decreases with dimension;

```
for ( level = 1; level <= level_max; level++ )
{
  p = 5;
  o = 3;
  while ( p < 2 * level + 1 )
  {
    p = 2 * p + 1
    o = 2 * o + 1
  }
  order_1d[level] = o
}
new_1d[0] = 1;
for { level = 1; level <= level_max; level++ )
{
  new_1d[level] = order_1d[level] - order_1d[level-1];
}
```

9 Conclusion

The GP_SE slow exponential growth family is an alternative to the standard GP_E indexing of the Gauss Patterson rules to form a sparse grid. The GP_SE formulation never uses more points than GP_E does. While the GP_E family will produce a sparse grid with somewhat more accuracy, both families produce a sparse grid of the same precision, that is, a sparse grid of level L, produced by either family, will be able to exactly integrate any polynomial of total degree $2L + 1$.

The advantage to using the GP_SE family is that it will require fewer function evaluations. The extent of this advantage varies. It is most obvious for low dimensions, or for high levels. However, at least in the range of dimensions up to 10, and for levels of at least 5 or 6, the GP_SE grid uses less than a fourth as many points as the GP_E grid does.

This document and the related document on the Clenshaw Curtis rule [3] consider ways of increasing the efficiency of a sparse grid by employing a nested quadrature rule, but restraining that rule's natural exponential growth. The procedure is done in such a way that the expected precision of the sparse grid of level L is retained. This means that, if precision is the goal, it is permissible to compare the efficiency of sparse grids by considering the point count at the same precision level and dimension. For instance, for precision 13, we must compare level 6 the exponential and slow exponential families, and level 3 for the moderate exponential families. At dimension 10, the corresponding point counts are:

CC_E = 652,065	CC_SE = 536,705
GP_E = 1,862,145	GP_SE = 169,185

so that if the standard construction with exponential growth were used, the CC_E family looks much superior to the GP_E. However, the dramatic improvement of the GP_SE family suggests that the original GP_E family was hampered by its slightly faster rate of growth in 1D compared to the CC_E family, and was not taking full advantage of the extra precision that the rule offers. By going to the slow exponential growth form, a 3-fold CC_E advantage over GP_E turns into more than a 3-fold advantage for the GP_SE family over CC_SE.

The **SANDIA_RULES** [1] includes some functions which can produce the CC_E, CC_SE, GP_E and GP_SE families of 1D quadrature rules. The **SGMGA** library [2] can compute sparse grids based on these rules.

References

[1] JOHN BURKARDT, SANDIA_RULES: Sparse Grid Mixed Growth Anisotropic Rules, http://people.sc.fsu.edu/∼jburkardt/cpp_src/sandia_rules/sandia_rules.html.

[2] JOHN BURKARDT, SGMGA: Sparse Grid Mixed Growth Anisotropic Rules, http://people.sc.fsu.edu/∼jburkardt/cpp_src/sgmga/sgmga.html.

[3] JOHN BURKARDT, Slow Exponential Growth for Clenshaw Curtis Sparse Grids, http://people.sc.fsu.edu/∼jburkardt/presentations/sgmga_ccs.pdf.

[4] ERICH NOVAK, KLAUS RITTER, High dimensional integration of smooth functions over cubes, Numerische Mathematik, Volume 75, Number 1, November 1996, pages 79-97.

[5] 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.

[6] THOMAS PATTERSON, The optimal addition of points to quadrature formulae, Mathematics of Computation, Volume 22, Number 104, October 1968, pages 847-856.

[7] 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.

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