FINAL REPORT: Advanced Numerical Methods for Computing Statistical Quantities of Interest from Solutions of Stochastic Partial Differential Equations

http://people.sc.fsu.edu/~jburkardt/presentations/sandia_2010.pdf

....

John Burkardt
Interdisciplinary Center for Applied Mathematics
& Information Technology Department
Virginia Tech

15 August 2010

This report represents the results of the collaboration between Dr. John Burkardt (the PI), at Virginia Tech, Dr. Clayton Webster (the original grant administrator), and Dr. Mike Eldred (who took over as grant administrator), at Sandia National Laboratories, with the supervision of Dr Terry Herdman at Virginia Tech. The grant began on January 1, 2008, with an initial termination of December 31, 2008, which was extended several times, most recently to 30 September 2010. However, as Dr Burkardt is leaving Virginia Tech as of 15 August 2010, he is relinquishing the grant at that time.

During this collaboration, the PI assisted in the investigation and resolution of a number of algorithmic, mathematical, and practical issues in the efficient, accurate, and robust numerical solution of *stochastic partial differential equations* ("SPDE"s) and similar issues that arise in the evaluation of quantities of interest that depend on such solutions.

The research carried out during this time was predominantly in the area of the construction and analysis of sparse grids, starting from a simple, commonly used procedure, which was to be extended in ways that correspond to the needs of computations involving SPDE's, including integration and interpolation, for collocation methods and polynomial chaos expansions.

Initial work focused on the generation of sparse grids which used the same fixed Clenshaw Curtis rule ("CC") in each dimension, used the same roughly exponential growth pattern for the families of one dimensional rules, and used a multidimensional linear constraint that resulted in isotropic grids. Figure 1 displays a simple example of the corresponding sparse grid of level 5 in dimension 2.

The Clenshaw Curtis sparse grid is an example of an isotropic sparse grid. That is, the resulting grid will have the same properties in each spatial dimension. The rule for constructing the grid of level L in dimension M has the form:

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

where the multidimensional linear constraint appears under the summation sign, and each $\mathcal{U}$ is a member of a given 1D family of quadrature rules.

The primary appeal of the sparse grid approach lies in the fact that it can achieve a desired degree of polynomial precision at a cost that is generally far lower than that of a corresponding product rule. Here, the

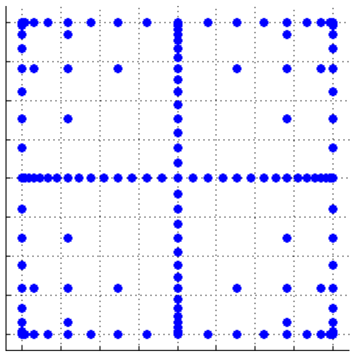

Figure 1: A sparse grid formed from Clenshaw-Curtis rules.

cost is generally quantified in terms of the number of abscissas involved in the rule. For product rules, the determination of the number of abscissas is straightforward. On the other hand, sparse grids are generated in a more elaborate way, and the counting is correspondingly more challenging, particular in cases where nesting occurs and duplicate points are to be identified.

Thus a particular concern in this project was the development of rules for predicting the number of distinct multidimensional abscissas required for a CC sparse grid, given the spatial dimension and the sparse grid level. Table 1 displays the results for such counts, ranging from dimensions 1 to 10, and from levels 0 to 10.

The counting procedure for the CC sparse grids was relatively easy to develop; however, the counting process depends a great deal on whether the given family of rules is nested. Since studies of non-nested or partially-nested families were to be considered, it was important to analyze the computation of the CC count in some detail, so that the necessary adjustments for other rules would be more readily determined. Table 2 breaks down the construction of a level 5 CC sparse grid in two dimensions, by considering it as the sum of product rules of levels 0 through 5. There is much overlapping as the higher order product rules are added, so the table is intended to count the number of points that first show up as a particular product rule is added. We begin, of course, with a single point representing the product CC(0)xCC(0). By tabulating the number of points that first appear with each added product rule, and analyzing the pattern, it was possible to come up with a computational procedure for the total number of unique points, and, moreover, to understand how the computation would need to be generalized for rules such as Gauss-Legendre or Gauss-Jacobi which have quite different nesting behaviors.

Having counted the *cost*, as the number of abscissas, it was necessary to verify the *benefit*, that is, the polynomial precision of a given sparse grid. For this purpose, a given sparse grid was used to integrate every monomial up to a user-specified degree, and the result compared to the known correct value. For the standard sparse grid construction, the typical precision result is that a sparse grid of level l will have polynomial precision $p = 2 * l + 1$. This was verified computationally.

Further understanding into the properties of sparse grids was obtained by considering the pattern of those monomials that exceeded the precision limit but which were nonetheless correctly integrated by the rule. For example, Figure 2 shows the required polynomial precision as a black line $x + y = 9$, the monomials exactly integrated by the level 3 rule marked by blue boxes, and the effect of moving to the level 4 rule indicated by the red boxes. It's clear that the level 4 rule is needed in order to fill in all the boxes below the line. What is surprising is how many boxes above the line get filled in. These extra monomials aren't required to meet the precision demand, and so the question naturally arises as to whether we can reduce the cost of the rule somewhat, while still making the required precision. These insights were used later, to develop more economical versions of the sparse grid rules, driven by the goal of meeting the precision requirement as cheaply as possible.

The CC rule is the primary starting point for sparse grid work because it is naturally nested. However,

DIM:	1	2	3	4	5	6	7	8	9	10
LEVEL										
0	1	1	1	1	1	1	1	1	1	1
1	3	5	7	9	11	13	15	17	19	21
2	5	13	25	41	61	85	113	145	181	221
3	9	29	69	137	241	389	589	849	1,177	1,581
4	17	65	177	401	801	1,457	2,465	3,937	6,001	8,801
5	33	145	441	1,105	2,433	4,865	9,017	15,713	26,017	41,265
6	65	321	1,073	2,929	6,993	15,121	30,241	56,737	100,897	171,425
7	129	705	2,561	7,537	19,313	44,689	95,441	190,881	361,249	652,065
8	257	1,537	6,017	18,945	51,713	127,105	287,745	609,025	1,218,049	2,320,385
9	513	3,329	13,953	46,721	135,073	350,657	836,769	1,863,937	3,918,273	7,836,545
10	1,025	7,169	32,001	113,409	345,665	943,553	2,362,881	5,515,265	12,133,761	25,370,753

Table 1: The number of distinct abscissas for a CC sparse grid.

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

Integrand	Error
1	0.0000000000000004
x	0.0000000000000002
y	0.0000000000000003
x^2	0.0000000000000002
xy	0.0000000000000002
y^2	0.0000000000000002
x^3	0.0000000000000002
x^2y	0.0000000000000002
xy^2	0.0000000000000002
y^3	0.0000000000000002
x^4	**0.0416666666666663**
x^3y	0.0000000000000003
x^2y^2	0.0000000000000001
xy^3	0.0000000000000003
y^4	**0.0416666666666663**
x^5	**0.1249999999999998**
x^4	**0.0416666666666663**
x^3y^2	0.0000000000000002
x^2y^3	0.0000000000000002
xy^4	**0.0416666666666663**
y^5	**0.1249999999999998**

Table 3: The error made when estimating the integrals of certain monomials using a CC sparse grid of level 1 in dimension 2. The data verifies the expectation of precision for monomials through total degree 3.

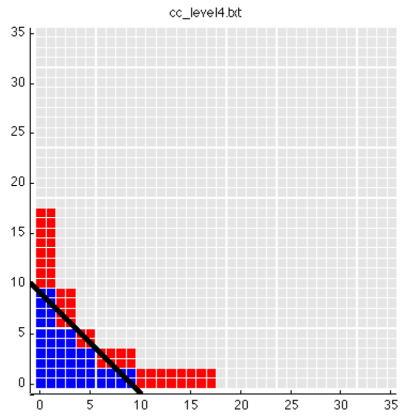

Figure 2: The precision requirement for level 4 as a black line; blue boxes are monomials that can be integrated by the level 3 rule, while red boxes show the monomials added when the level 4 rule is used.

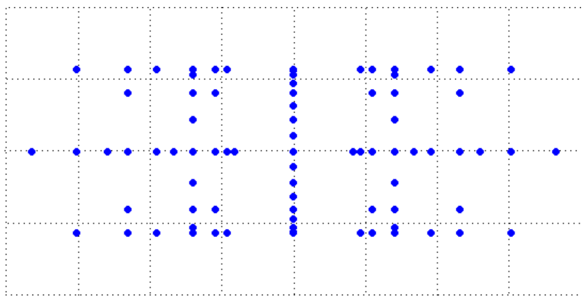

Figure 3: A sparse grid in 2 dimensions, and level 4, using a mixture of rules. The X dimension uses the Gauss-Hermite family, the Y dimension Clenshaw-Curtis.

this family of rules is not of the highest possible precision. For that reason, variations of the procedure were developed which were based on the Gauss-Patterson rule ("GP") and the Gauss-Legendre ("GL"). The GP rule retains the nestedness of the CC rule and improves the precision somewhat; the GL rule discards the nestedness almost entirely, and aims for maximal possible precision. Sparse grids based on these two rules turned out to have a number of interesting properties distinct from the CC family.

Newton-Cotes rules ("NC"), which are defined at equally spaced abscissas, were also considered for study. However, as rules for 1D integration, the NC family has the disadvantage that as the level increases, some of the weights become negative, and in an unbounded fashion. Even if the underlying family has only positive weights, applying the sparse grid procedure will generally result in some negative weights. If the underlying family already includes rules with negative weights, there was some concern that this could lead to a significant loss in accuracy for certain problems. Since the CC, GP and GL rules were already available, it was decided, after some initial investigations, that the NC family was not going to be further developed to participate in the sparse grid project.

Many of the SPDE computations of interest to researchers at Sandia involved probability density functions ("PDF"), in which case the polynomial precision of the CC, GL and GP rules would not apply. For this reason, techniques were selected for generating arbitrary members of the Gauss-Hermite and Gauss-Laguerre rules, both standard and generalized forms, and of the Gauss-Jacobi rule. Correspondingly, the sparse grid procedure could now produce sparse grids based on these 1D families as well.

So far, development had assumed that a given sparse grid used the same 1D family in all dimensions. However, it was a common feature of stochastic problems to model a set of variables, each of which was governed by a distinct PDF. In such a case, it was natural to hope that a sparse grid could be derived so that the spatial dimension corresponding to any variable was integrated with the appropriate rule. This resulted in an approach in which 10 different families of quadrature rules were available, and the user was free to construct a sparse grid in which each dimension used the rule appropriate for the corresponding variable. Figure 3 displays an example of such a sparse grid, in which Gauss-Hermite and Clenshaw-Curtis rules are combined.

Particular note is made of the 10th family of rules available for defining mixed grids. While the abscissas and weights of the other rules are assumed to be computed internally, by the procedure, a request for rule 10 is actually an indication that the family of rules is to be provided by the user, typically from a Golub-Welsch procedure. In this way, a user with a specialized problem involving unusual or empirical PDF's may still get the benefit of the sparse grid approach, at the cost of developing the abscissas and weights of the appropriate 1D quadrature rule. This feature has already been implemented and employed by researchers at Sandia.

Another issue that was examined involved the "growth rule" used to create a 1D family of rules. For the Clenshaw-Curtis family, this rule was an essentially exponential growth of the order o in terms of the level l; with the exception of the level 0 rule, the formula was

$$o(l) = 2^l + 1$$

L	P	CC(L)	CCS(L)
0	1	1	1
1	3	3	3
2	5	5	5
3	7	9	9
4	9	17	9
5	11	33	17
6	13	65	17
7	15	129	17
8	17	255	17
9	19	513	33
10	21	1025	33

Table 4: Comparison of the standard (CC) and slow (CCS) Clenshaw Curtis 1D families. For a given level L, and precision requirement P, we record the number of points used in the 1D rules.

Using this growth rule guaranteed the precision relationship $p(l) = 2l + 1$, as well as enforcing nesting. However, it seemed that, particularly when other families were involved, it might be valuable to allow the user to control the growth rule as well. To this end, it was determined that for each spatial dimension, a growth parameter might be defined, which allowed the user to control the growth. Particular examples of alternate growth rules included:

$$\begin{aligned} o(l) =& l + 1 \quad \text{(Slow linear growth)} \\ o(l) =& 2l + 1 \quad \text{(Moderate linear growth)} \\ o(l) =& 4l + 1 \quad \text{(Fast linear growth)} \end{aligned}$$

These linear rules were of particular interest for rules of Gaussian type; experience suggests that the low and moderate linear growth rules often produced satisfactory rules at low cost; the fast linear growth rule was less successful, involving excessive point growth without corresponding regular increases in polynomial precision.

Growth rules were also examined in the hopes of improving the performance of nested rules, without losing the nesting. Here, the particular examples of CC and GP rules were of interest. It was discovered that it was possible to slow down the growth of the 1D family, essentially by repeating the use of certain rules as long as they supplied a particular level of accuracy. Thus, in particular, if we start out by requiring that the sparse grid of level l must achieve a precision of $2l+1$, it turns out that our family of 1D rules must have at least this same precision growth. The interesting fact is that the standard CC family of 1D rules far exceeds this precision growth. As an economy measure, then, we can imagine producing a new "slow" CC family ("CCS") which contains the very same elements as the CC family, but which repeats a particular element as long as possible, until it can't match the 1D precision requirement.

The precision of a CC rule of order o is o, if o is odd. Thus, in Table 4, we tabulate the 1D level, the 1D precision requirement, and the orders of the corresponding elements of the CC and CCS families.

By being more economical in the 1D rules, the CCS family is able to produce multidimensional sparse grids of lower order. Since the 1D CC and CCS families only diverge at level 4, the improvement in the multidimensional case also only begins at that level. Table 5 shows the point counts for the CCS rule, which can be compared with the corresponding counts for the CC rule in Table 1. For a fixed dimension, the difference quickly grows. Moreover, the reduction in order comes at no loss of nesting. Slow growth versions of both the CC and GP families were therefore implemented; it was noted that the primary improvement comes for sparse grids in relatively low dimensions, say up to dimension 10. After that, at least for grids of moderate levels, most of the 1D factors will be of low level, and hence the differences between CC and CCS will not appear so strongly unless grids of quite high level are involved.

From the beginning of this project, a major goal was to develop the ability to produce anisotropic sparse

DIM:	1	2	3	4	5
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

Table 5: The number of distinct abscissas for a CCS sparse grid.

grids that correspond to anisotropies perceived in the underlying problem. For our purposes, anisotropy simply means the ability to express and implement preferences for particular spatial dimensions. For instance, if we needed to estimate the integral of a function $f(x, y) = x^2 sin(y)$, we see that there is little point in performing high accuracy integration in the x direction; therefore, given a budget of a fixed number of abscissas, we would be wise to allow our sparse grid to have a stronger resolving ability in the y direction.

In general, the user may only be able to provide some sort of weights for the spatial directions, with a relatively high weight indicating that the sparse grid should have more strength or precision in the corresponding spatial dimension.

Accordingly, a scheme was developed for taking a simple anisotropy weight vector from the user, and using that to develop an anisotropic sparse grid. deal with anisotropic problems by producing

The anisotropic sparse grid of 0-based index w, spatial dimension d and anisotropy vector α, is denoted by $\mathcal{A}(w, d, \alpha)$, and the defining equation has the form:

$$\mathcal{A}(w, d, \alpha) = \sum_{i \in Y_\alpha(w,d)} c_\alpha(\mathbf{i})(\mathcal{U}^{i_1} \otimes \cdots \otimes \mathcal{U}^{i_M}) \tag{2}$$

with the combining cofficient $c_\alpha(\mathbf{i})$, which takes the place of the combinatorial coefficient in Equation 1, defined by:

$$c_\alpha(\mathbf{i}) = \sum_{\substack{\mathbf{j} \in \{0,1\}^d \\ \mathbf{i}+\mathbf{j} \in X_\alpha(w,d)}} (-1)^{|\mathbf{j}|} \tag{3}$$

and the selection region of acceptable product rules defined by

$$Y_\alpha(w, d) = X_\alpha(w, d) \backslash X_\alpha(w - \frac{|\alpha|}{\underline{\alpha}}, d) \tag{4}$$

where

$$X_\alpha(w, d) = \{\mathbf{i} \in \mathbb{N}^d_+, \mathbf{i} \geq 1 : \sum_{n=1}^{d} (i_n - 1)\alpha_n \leq w\underline{\alpha}\} \tag{5}$$

with $\underline{\alpha} = \min_{1 \leq n \leq d} \alpha_n$ and $|\alpha| = \sum_{n=1}^{d} \alpha_n$.

It turns out that the expression for the combining coefficient $c_\alpha(\mathbf{i})$ has a fairly straightforward meaning. The combining coefficient is formed by a sum. The sum is over a set of perturbations of the fundamental level vector $\mathbf{i}$. The perturbed level vectors have the form $\mathbf{i}+\mathbf{j}$, where the entries of the $\mathbf{j}$ vector are 0 or 1. A perturbed level vector $\mathbf{i}+\mathbf{j}$ contributes to the sum only if it satisfies the same sparse grid selection constraint

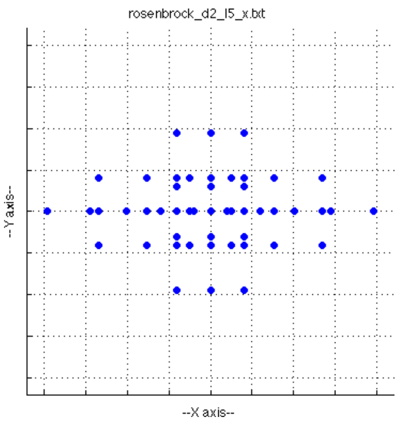

Figure 4: A sparse grid in 2 dimensions, and level 5, using the Gauss-Hermite rule in both dimensions, with an anisotropy designed to capture the Rosenbrock function.

that **i** does, that is, only if **i**+**j** is also a component of the sparse grid. In that case, a contribution of $(-1)^{|\mathbf{j}|}$ is added to $c_\alpha(\mathbf{i})$. So generating the combining coefficient is simply a matter of generating all the perturbed vectors, checking whether they satisfy the criterion, and if so incrementing the sum with +1 or -1.

The simplicity of Equation 3 for the combining coefficient is counterbalanced by the fact that the index on the summation indicates that we must consider all possible increments of the base level vector **level_1d** by a nonzero binary vector **j**. A naive algorithm would proceed to generate every such increment and evaluate the constraints. The cost of calculating a *single combining coefficient* of a sparse grid in dimension M would then be of the order of 2^M, which is exponential in the dimension. Such a cost would invalidate the entire approach, so an alternative method for evaluating these coefficients was needed if the anisotropic approach was to be carried out.

Now in most cases of interest, the values of the anisotropic weight vector and the level are such that the actual number of level vectors satisfying the level constraints is very small. Hence, if we ask what subset of those solutions can be reached by a binary perturbation of some particular solution, the count must be even smaller, and the upper limit of 2^M is an absurd overestimate. There is hope, then, that at least for modest values of the level, we can replace the naive approach by an intelligent, efficient search procedure, particularly if, as we search, we can rapidly reject large portions of the search region.

After much consideration and trial, a procedure was found which avoids the exponential explosion seemingly inherent in the simple definition of the combinatorial coefficient. The procedure orders the set of all possible perturbations and then proceeds to consider the perturbations in a particular order. If it determines that a particular perturbation cannot satisfy the constraint, then not only does it rule this perturbation out, but it is also able to rule out a (generally large) family of related perturbations, namely those which have 1's in the same place as the given perturbation. This allows the procedure to skip quickly through the list and avoid the exponential search.

Thus, an anisotropic version of the sparse grid algorithm was developed, which includes the ability to mix families of rules, to choose the growth rule for each dimension, and now to specify an anisotropic weight vector that predisposes the sparse grid to grow more vigorously in certain directions. This procedure now embodied in a single interface all the advances that had been made so far.

Figure 4 displays an example of such a grid, associated with the Rosenbrock function. The grid has been weighted to show a preference for the x direction.

The anisotropic algorithm has only been in place for a relatively short time. Efforts have been made to verify the precision predictions, to experiment with new 1D families, and to modify the form of the constraint that selects the product rules used to assemble a particular anisotropic sparse grid. A number of Sandia researchers have employed the algorithms for their work, and in particular, Mike Eldred has been using it extensively.

Other issues remain open, such as adaptivity, in which a sparse grid could be used to estimate the errors incurred by ignoring certain monomials, and then selectively add the product rules necessary to capture monomials associated with high errors. Another issue is the use of hierarchical functions, or composite functions, to handle cases in which the function may have local singularities. Some of these issues were included in the “wish list” of topics proposed as suggestions for study at the beginning of this project, and remain worth investigating in the future.

Over the course of the development of the procedures described here, Mike Eldred has repeatedly updated Sandia’s Design Analysis Kit for Optimization and Terascale Applications (DAKOTA) to include implementations of the algorithms. Now, with the advent of Sandia’s Predictive Engineering and Computational Sciences package (PECOS), many of these same functions are migrating there as well. DAKOTA and PECOS thus make the functionality of the sparse grid approaches available to a wide range of researchers, and do so within the context of a wider software environment that enables the rapid design of software applications that analyze stochastic phenomena.