Sparse Grids for Stochastic Integrals

John Burkardt,
Information Technology Department,
Virginia Tech.

.....

Department of Scientific Computing,
Florida State University,
08 October 2009.

.....

http://people.sc.fsu.edu/~jburkardt/presentations/...
sparse_2009_fsu.pdf

Sparse Grids for Stochastic Integrals

1. **Integrals Arising from Stochastic Problems**
2. Quadrature Rules
3. Product Rules for Higher Dimensions
4. Smolyak Quadrature
5. Covering Pascal's Triangle
6. How Grids Combine
7. Sparse Grids in Action
8. Smoothness is Necessary
9. A Stochastic Diffusion Equation
10. Software
11. Conclusion

Stochastic Integrals: Background

Stochastic problems are mathematical models that attempt to include an aspect that is nondeterministic or uncertain.

In a mathematical model, even uncertainty has to be described precisely.

So when an uncertain process produces a new state or result, we imagine this as coming about because that state was selected from an ensemble of possible states, with each state having a particular probability of being selected.

If our stochastic problem involved only a single such choice, then it should be clear that we can "understand" our problem by considering every possible outcome, and that we can compute an expected value by multiplying every possible outcome by its probability.

Stochastic Integrals: Background

Stochastic problems typically involve far more than a single discrete uncertain choice.

If a very sensitive weathervane points north at sunrise, then we can imagine that at each moment its current direction is affected by the wind, which has a stochastic component. The final direction of the weathervane can be determined only by summing up the continuous influence of the wind.

Stochastic Integrals: Background

An initially uniform pipeline endures corrosion both inside and outside. If we wish to consider the likely state of the pipeline after twenty years, and we only know the general features of the soil and weather conditions over that time, then the state of the pipeline can be modeled as a spatially stochastic function.

Stochastic Integrals: Background

Actually, the final location of the weathervane, under stochastic influence, is likely to be the same as the deterministic location, if the noise has an equal tendency to turn the wind clockwise or counterclockwise.

A more interesting question is to measure the absolute angular distance that the weathervane travels, so we can estimate the wear on the bearing.

For the pipeline problem, in order to generate answers, we need to have some rough intuition in the variation of the cement mix that was used, the average and variation in corrosion under given rain and temperature conditions, and so on. Even uncertainty has to be partially parameterized by values we guess at.

Stochastic Integrals: Background

Partial differential equations allow us to describe the behavior of of a quantity $\mathbf{u}(\vec{x})$ and often to solve for or approximate its value.

A classic example is the diffusion equation:

$$-\nabla \cdot (a(\vec{x})\nabla u(\vec{x})) = f(\vec{x})$$

$\mathbf{a}(\vec{x})$ is a spatially varying diffusion coefficient;
$\mathbf{f}(\vec{x})$ represents a source term.

Stochastic Integrals: Background

A finite element approach would produce an approximate solution by integrating the equation against various test functions $\mathbf{v_i}(\vec{x})$:

$$\int_D a(\vec{x})\,\nabla u(\vec{x}) \cdot \nabla v_i(\vec{x})\,d\vec{x} = \int_D f(\vec{x}) v_i(\vec{x})\,d\vec{x}$$

By assuming $u(\vec{x}) = \sum_{j=1}^{N} c_j v_j(\vec{x})$, this becomes a set of **N** linear algebraic equations $A * c = f$.

Evaluating **A** and **f** requires approximate integration over **D**.

Stochastic Integrals: Background

A stochastic version of this equation might allow uncertainties in the diffusion coefficient:

$$-\nabla \cdot (a(\vec{x};\omega)\nabla u(\vec{x};\omega)) = f(\vec{x})$$

Here, ω represents the stochastic component, and we must even write **u** with an implicit dependence on ω, through **a**.

We could, if we liked, also consider uncertainties in the source term **f**.

Stochastic Integrals: Background

Now since ω is an unknown and undetermined quantity, it might seem that the solution process is hopeless.

Actually, if we could specify a particular set of values for the stochastic component ω, then presumably we could solve for $\mathbf{u}$, so our problem is really that our classical solution has now become a family of solutions with parameter ω.

Or, if we are willing to integrate over all possible values of ω, multiplied by the corresponding weight function $\rho(\omega)$, we can get the finite element coefficients of the expected value function $\bar{\mathbf{u}}(\vec{x})$.

Stochastic Integrals: Background

We want to know how stochasticity will affect the classical solution $\mathbf{u}(\vec{x})$; comparing $\mathbf{u}(\vec{x})$ to $\bar{\mathbf{u}}(\vec{x})$ will indicate the expected magnitude of the deviations caused by the stochastic component.

To compute $\bar{\mathbf{u}}(\vec{x})$ or other statistical moments, we add stochasticity, weight it, and then integrate it out. The integration processes collapses the infinite family of perturbed solutions into one object which represents a bulk average perturbed solution, and which can be compared to a classical solution.

You can plot $\bar{\mathbf{u}}(\vec{x})$ for instance, or evaluate it, or find its maximum.

Stochastic Integrals: Background

If we simply integrate over the whole probability space, we get:

$$\int_\Omega \int_D a(\vec{x};\omega)\,\nabla u(\vec{x};\omega)\cdot\nabla v_i(\vec{x})\,d\vec{x}\,\rho(\omega)\,d\omega =$$
$$\int_\Omega \int_D f(\vec{x})v_i(\vec{x};\omega)\,d\vec{x}\,\rho(\omega)\,d\omega$$

Computationally, we will work in some $\mathbf{M}$ dimensional space $\mathbf{\Omega^M}$; depending on our approach, we may have some basis functions for the noise, or other factors included in the integral.

We can still regard this as, essentially, an algebraic system $A * c = f$ for the finite element coefficients of $\bar{\mathbf{u}}(\vec{x})$, but now evaluating $\mathbf{A}$ and $\mathbf{f}$ requires approximate integration over $\mathbf{D}$ AND over the very different space $\mathbf{\Omega^M}$.

Stochastic Integrals: Background

In summary, we want to add a stochastic component to a standard PDE, and then use integration to average over all possible perturbed states, applying a weight associated with the probability of each perturbation.

We seek the expected value function $\bar{\mathbf{u}}(\vec{x}) = \mathbf{E}(\mathbf{u}(\vec{x}; \omega))$ for our stochastically perturbed system.

Our approximation procedure will involve integration over the fixed and low-dimensional spatial domain $\mathbf{D}$, (which we know how to do) and over a probability space $\mathbf{\Omega^M}$ whose shape is simple, but whose dimension $\mathbf{M}$ depends on how we approximate the stochastic component.

Stochastic Integrals: Background

We approximate integrals over a weighted **M**-dimensional space:

$$\int_{\Omega^M} f(\vec{x};\omega)\,\rho(\omega)\,d\omega$$

This integral is very different from a finite element spatial integral.

Here, ω is a vector whose dimension may be increased as we seek greater accuracy. This means we may easily find ourself integrating over a space of dimension $\mathbf{M} = 20$, 50 or 100.

While the dimension is a problem, the function $\rho(\omega)$ is usually smooth, and the "geometry" of $\mathbf{\Omega^M}$ is a simple product region.

Stochastic Integrals: Background

In order to complete the process of solving stochastic equations, we need a way to approximate the integral of smooth functions over simple regions in an unusually high dimensional space that is a product region.

There are a variety of 1D quadratures rules for integrating over the regions and weight functions associated with many probability distributions.

We can use a mixture of such rules to form a multidimensional **product rule**.

When the product rules become too expensive, we can turn to **sparse grids**.

Sparse Grids for Stochastic Integrals

1. Integrals Arising from Stochastic Problems
2. **Quadrature Rules**
3. Product Rules for Higher Dimensions
4. Smolyak Quadrature
5. Covering Pascal's Triangle
6. How Grids Combine
7. Sparse Grids in Action
8. Smoothness is Necessary
9. A Stochastic Diffusion Equation
10. Software
11. Conclusion

Quadrature Rules

A 1D quadrature rule is a set of **N** points **X** and weights **W** intended to approximate integrals over a particular interval **[a,b]**, perhaps including a weight function:

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

Note that $\rho(\mathbf{x})$ is a *weight function* and $\mathbf{w_i}$ is a *quadrature weight*. They are both kinds of weights, but only indirectly related.

For the integrals we are interested in, the interval is either [-1,+1], [0,$+\infty$) or ($-\infty$,$+\infty$) and $\rho(\mathbf{x})$ will be a smooth function.

Quadrature Rules

If a 1D quadrature rule can compute the exact integral when $\mathbf{f(x)}$ is a polynomial of degree **P** or less, the rule has **precision P**.

The precision of common quadrature families can be given in terms of the order (number of points) **N**:

- Interpolatory rules: $\mathbf{P = N\text{-}1}$ or **N**
- Gauss rules $\mathbf{P = 2 * N - 1}$;
- Monte Carlo and Quasi-Monte Carlo rules, $\mathbf{P = 0}$;
- "transform rules": tanh, tanh-sinh, erf rules $\mathbf{P = 1}$.

For a product rule, precision requires exact results for all polynomials whose *total degree* is **P** or less.

Quadrature Rules: Precision Can Mean Accuracy

Using a rule with **P = N** on a function with **N** derivatives, the low order terms get integrated exactly, leaving error $\sim O(\frac{1}{N}^{N+1})$.

(Taking the typical spacing between abscissas to be $h = \frac{1}{N}$.)

The integrands encountered in high dimensional problems are typically smooth, and suitable for high precision rules.

An interpolation rule with order **N** will typically have precision **N − 1**, but if the rule is symmetric and **N** is odd, then the precision bumps up to **N**. This is the case we will concentrate on, so we'll assume that the order is equal to the precision.

Quadrature Rules: Work, Results and Efficiency

In 1D, an interpolatory rule uses **N** quadrature points and picks up **N** monomials exactly. We will take this one-for-one ratio between work (function evaluations) and results (monomials exactly integrated) as close to maximally efficient.

It is common to carry out a sequence of approximations to an integral. If the sequence is constructed in such a way that the points are **nested**, then the function values can be saved and reused as part of the next approximation. This is a second possible efficiency.

Quadrature Rules: Families of Rules

Most quadrature rules are available in any order **N**.

Generally, increasing **N** produces a more accurate estimate, and the difference between estimates of order **N** and **N+K** can be used as a rough approximation to the error in the lower order estimate.

Thus, a procedure seeking an integral estimate with a certain accuracy will need to generate and apply elements of a family of quadrature rules of increasing order.

An efficient calculation seeks a family of rules and a growth pattern for **N** so that old abscissas are reused. This is called **nesting**.

Quadrature Rules: Order, Level, Growth Rule

The **order** of a rule, **N**, is the number of points or abscissas.

The **level** of a rule, **L**, is its index in a family. This is simply a way to select a subset of the rules for our iterative scheme.

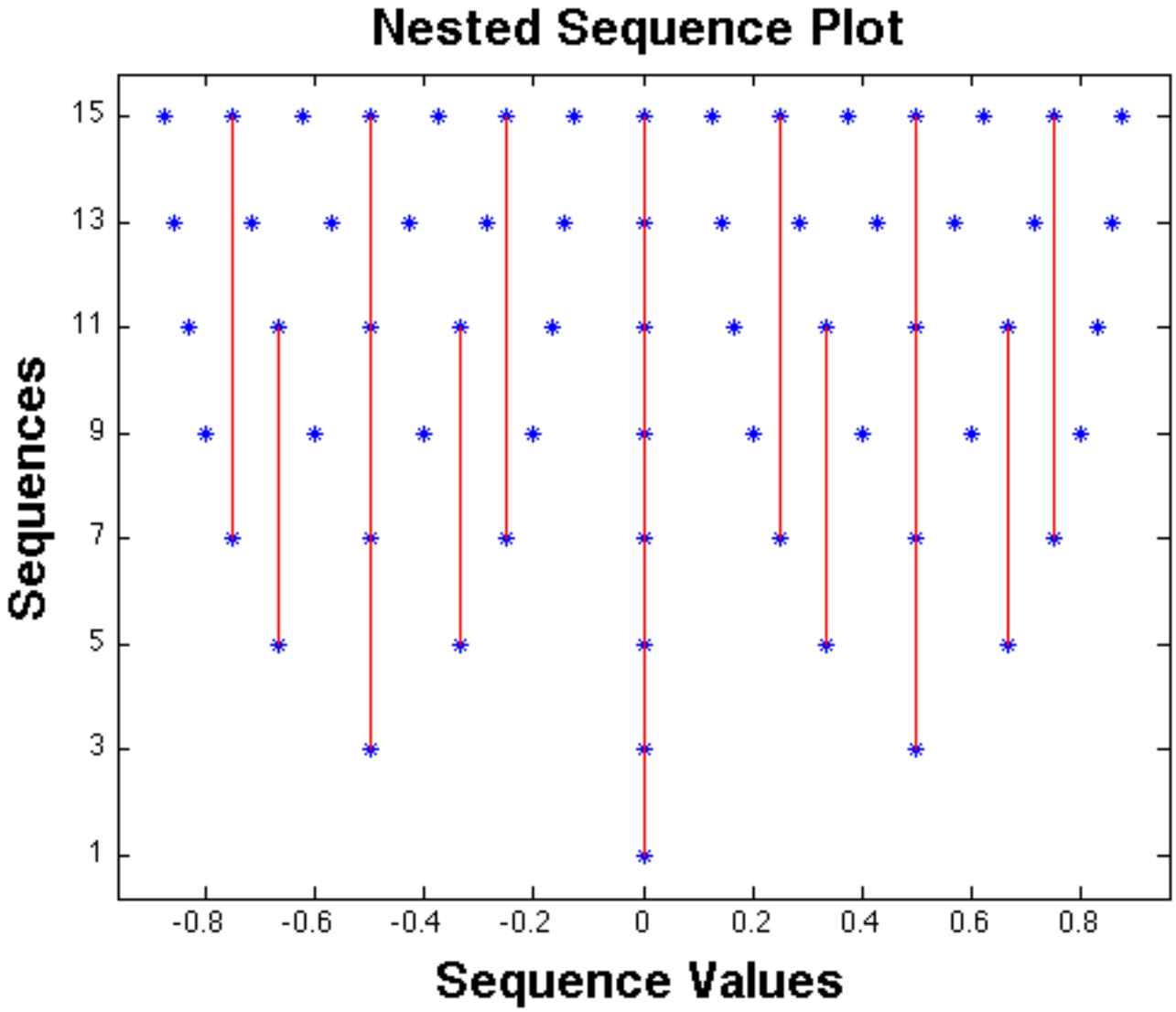
A family typically starts at level 0 with an order 1 rule.

The **growth rule** for a family relates level **L** and order **N**:

Linear growth is the simplest. It may allow a limited amount of nesting:

$$N = 2L + 1 : 1, 3, 5, 7, ...$$

Quadrature Rules: Newton Cotes Open,Linear Growth



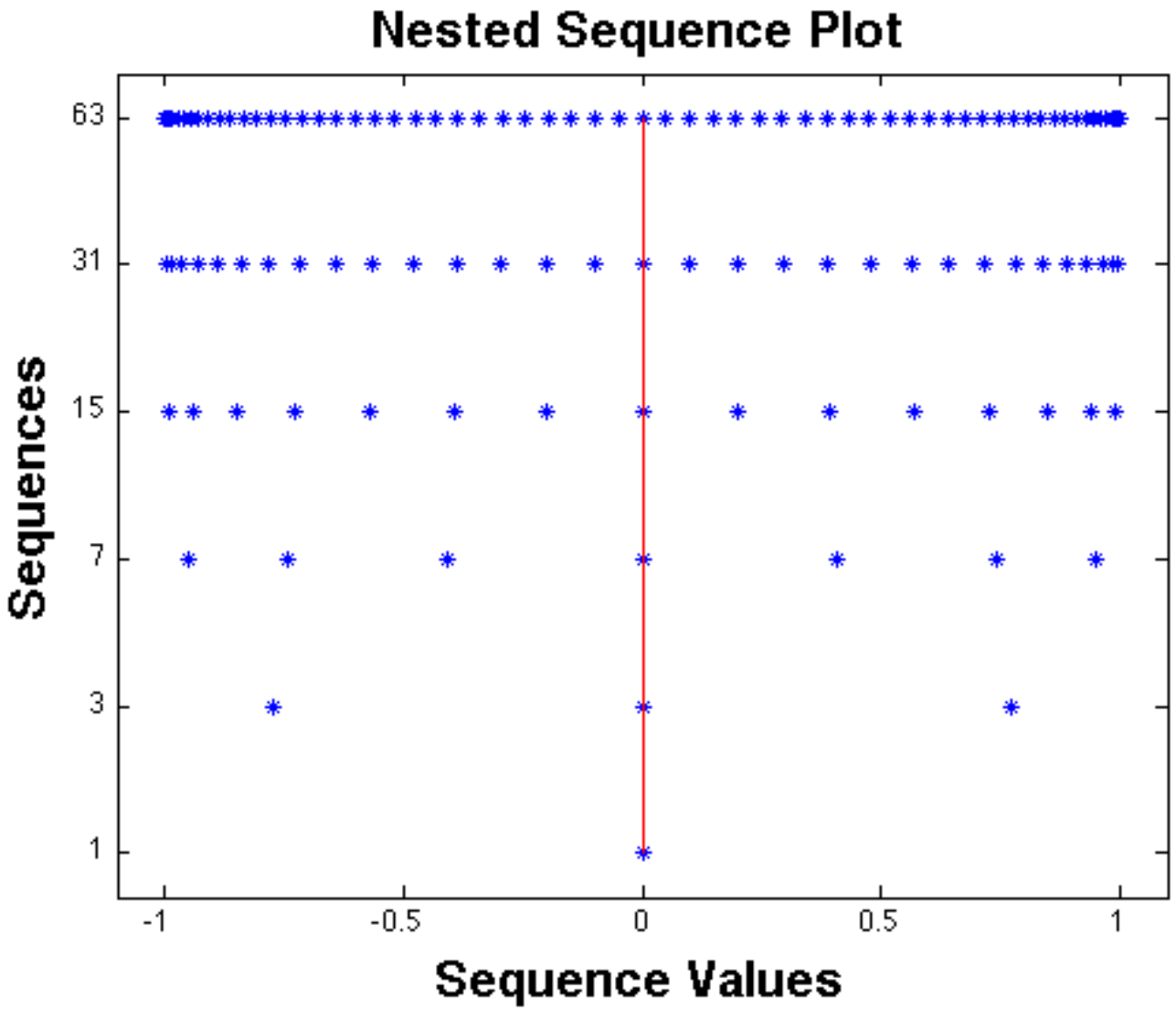
Quadrature Rules: Order, Level, Growth Rule

Exponential growth is rapid. Its exact form depends on whether the rule is closed (includes both endpoints of the interval) or open. Exponential growth is typical when nesting is exploited.

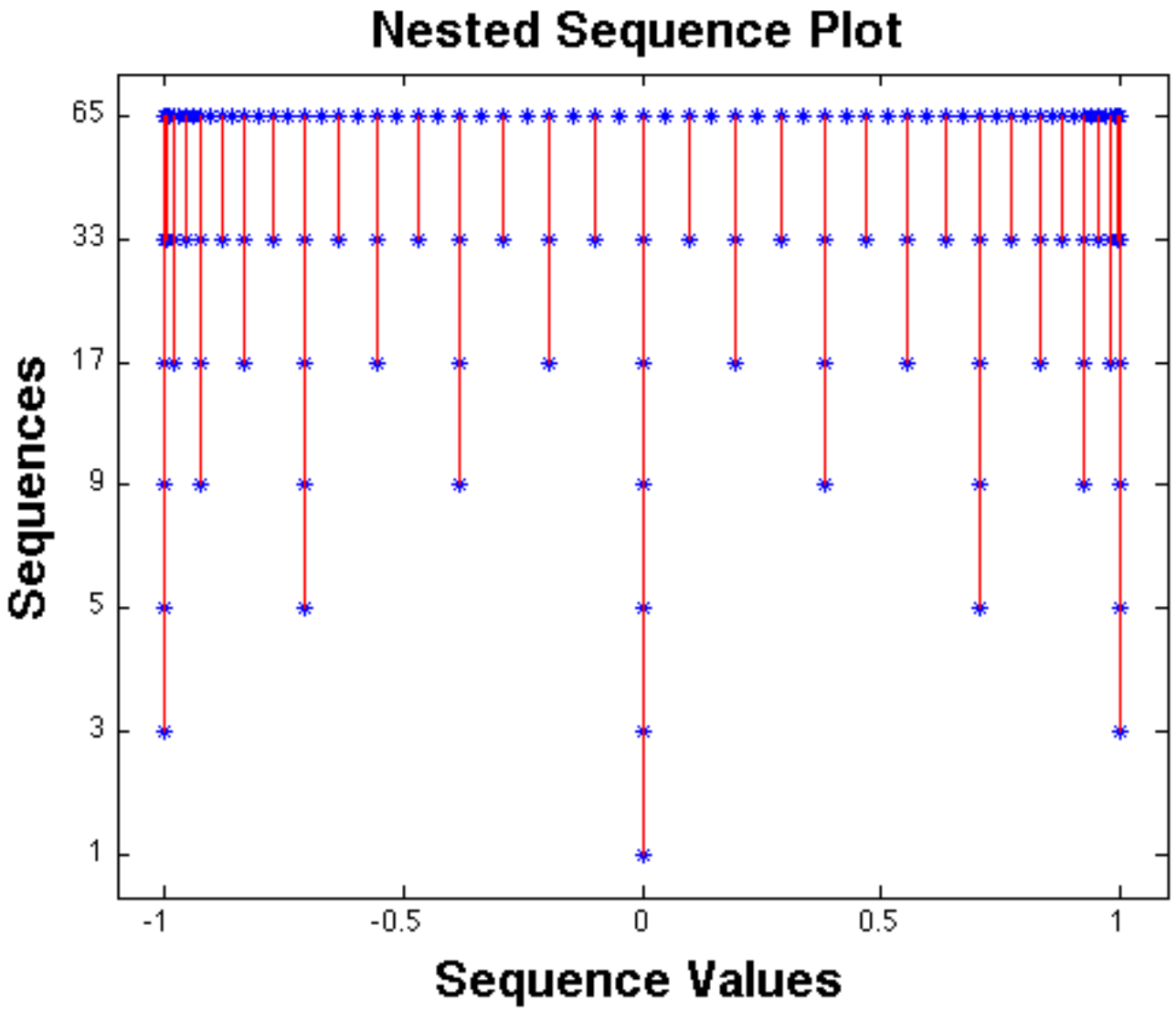
$N = 2^L + 1 : 1, 3, 5, 9, 17, 33 :$ closed rules, Clenshaw-Curtis

$N = 2^{L+1} - 1 : 1, 3, 7, 15, 31 :$ open rules, Gauss-Legendre

Quadrature Rules: Gauss Legendre, Exponential Growth



Quadrature Rules: Clenshaw Curtis, Exponential Growth



Sparse Grids for Stochastic Integrals

1. Integrals Arising from Stochastic Problems
2. Quadrature Rules
3. **Product Rules for Higher Dimensions**
4. Smolyak Quadrature
5. Covering Pascal's Triangle
6. How Grids Combine
7. Sparse Grids in Action
8. Smoothness is Necessary
9. A Stochastic Diffusion Equation
10. Software
11. Conclusion

PRODUCT RULES: Formed from 1D Rules

Let $\mathcal{Q}_L$ be the L-th member of a family of 1D quadrature rules, with order N_L and precision P_L.

We can construct a corresponding family of 2D product rules as $\mathcal{Q}_L \otimes \mathcal{Q}_L$, with order N_L^2 and precision P_L.

This rule is based on interpolating data on the product grid; the analysis of precision and accuracy is similar to the 1D case.

Similarly, we can construct an M-dimensional product rule. Notice that the order of such a rule is N_L^M, but the precision stays at P_L. Thus, even for a relatively low precision requirement, the order of a product rule will grow very rapidly if **M** increases.

How many points would you use in a product rule for **M**$=100$*?*

PRODUCT RULES: Formed from 1D Rules

For example, we can "square" the order 3 Legendre rule on [-1,1]:

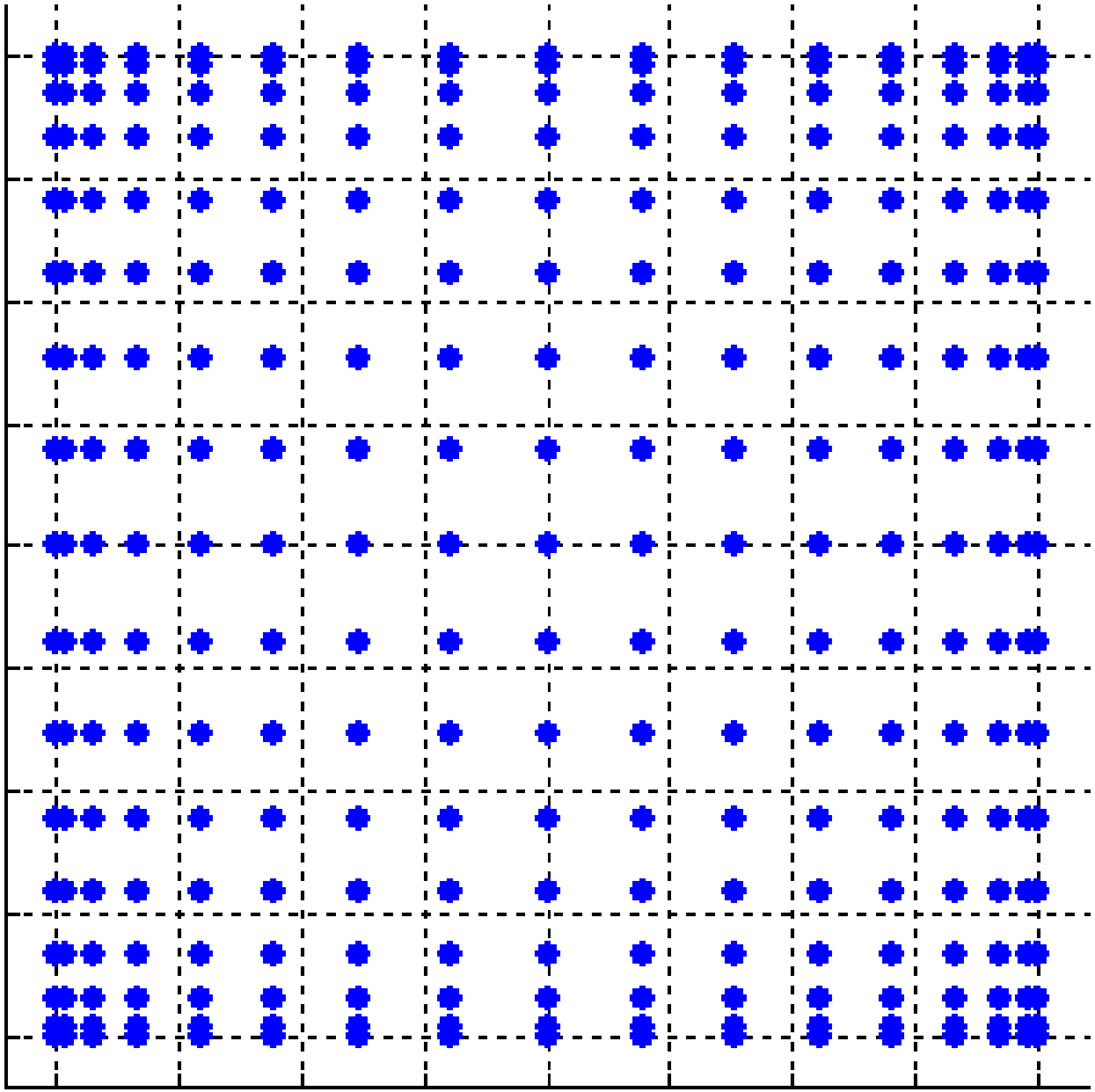
$$\int_{-1}^{+1} f(x)\,dx \approx 0.55 * f(-0.77) + 0.88 * f(0.0) + 0.55 * f(0.77)$$

to get a 2D product rule:

```
 0.30 f(-0.77,+0.77) +0.48 f(0,+0.77) +0.30 f(0.77,+0.77)
+0.48 f(-0.77, 0.00) +0.77 f(0, 0.00) +0.48 f(0.77, 0.00)
+0.30 f(-0.77,-0.77) +0.48 f(0,-0.77) +0.30 f(0.77,-0.77)
```

where point $(\mathbf{x_i}, \mathbf{y_j})$ has weight $\mathbf{w_i} * \mathbf{w_j}$.

PRODUCT RULES: 17x17 Clenshaw-Curtis



PRODUCT RULES: Do We Get Our Money's Worth?

Suppose we form a 2D quadrature rule by "squaring" a 1D rule which is precise for monomials 1 through x^4.

Our 2D product rule will be precise for any monomial in x and y with individual degrees no greater than 4.

The number of monomials we will be able to integrate exactly matches the number of abscissas the rule requires.

Our expense, function evaluations at the abscissa, seems to buy us a corresponding great deal of monomial exactness.

But for interpolatory quadrature, many of the monomial results we "buy" are actually **nearly worthless**!.

PRODUCT RULES: Pascal's Precision Triangle

We can seek accuracy by requiring that our quadrature rule have a given precision. To say our quadrature rule has precision 5 is to say that it can correctly integrate every polynomial of degree 5 or less. This corresponds to integrating all the monomials below the 5-th diagonal in a sort of Pascal's triangle.

A given rule may integrate some monomials above its highest diagonal; but these extra monomials don't improve the overall asymptotic accuracy of the rule.

PRODUCT RULES: Pascal's Precision Triangle

Here are the monomials of total degree exactly 5. A rule has precision 5 if it can integrate these and all monomials below that diagonal.

7	!	y^7	xy^7	x^2y^7	x^3y^7	x^4y^7	x^5y^7	x^6y^7	x^7y^7
6	!	y^6	xy^6	x^2y^6	x^3y^6	x^4y^6	x^5y^6	x^6y^6	x^7y^6
5	!	y^5	xy^5	x^2y^5	x^3y^5	x^4y^5	x^5y^5	x^6y^5	x^7y^5
4	!	y^4	xy^4	x^2y^4	x^3y^4	x^4y^4	x^5y^4	x^6y^4	x^7y^4
3	!	y^3	xy^3	x^2y^3	x^3y^3	x^4y^3	x^5y^3	x^6y^3	x^7y^3
2	!	y^2	xy^2	x^2y^2	x^3y^2	x^4y^2	x^5y^2	x^6y^2	x^7y^2
1	!	y	xy	x^2y	x^3y	x^4	x^5y	x^6y	x^7y
0	!	1	x	x^2	x^3	x^4	x^5	x^6	x^7
P	!	0	1	2	3	4	5	6	7

PRODUCT RULES: Pascal's Precision Triangle

A product rule results in a rectangle of precision, not a triangle. The monomials above the diagonal of that rectangle represent a cost that does not correspond to increased overall asymptotic accuracy.

7	!	y^7	xy^7	x^2y^7	x^3y^7	x^4y^7	x^5y^7	x^6y^7	x^7y^7
6	!	y^6	xy^6	x^2y^6	x^3y^6	x^4y^6	x^5y^6	x^6y^6	x^7y^6
5	!	y^5	xy^5	x^2y^5	x^3y^5	x^4y^5	x^5y^5	x^6y^5	x^7y^5
4	!	y^4	xy^4	x^2y^4	x^3y^4	x^4y^4	x^5y^4	x^6y^4	x^7y^4
3	!	y^3	xy^3	x^2y^3	x^3y^3	x^4y^3	x^5y^3	x^6y^3	x^7y^3
2	!	y^2	xy^2	x^2y^2	x^3y^2	x^4y^2	x^5y^2	x^6y^2	x^7y^2
1	!	y	xy	x^2y	x^3y	x^4	x^5y	x^6y	x^7y
0	!	1	x	x^2	x^3	x^4	x^5	x^6	x^7
P	!	0	1	2	3	4	5	6	7

PRODUCT RULES: It Gets Worse in Higher Dimensions

Consider products of a 10 point rule with precision up to x^9.

We only need to get to diagonal 9 of Pascal's precision triangle. The monomials up to that row can be computed as a multinomial coefficient. Compare the number of abscissas to monomials!

Dim	Abscissas	Monomials	Wasted	Percentage
1D	10	10	0	0%
2D	100	55	45	45%
3D	1,000	220	780	78%
4D	10,000	715	9,285	92%
5D	100,000	2,002	97,998	97%
6D	1,000,000	5,005	994,995	99%

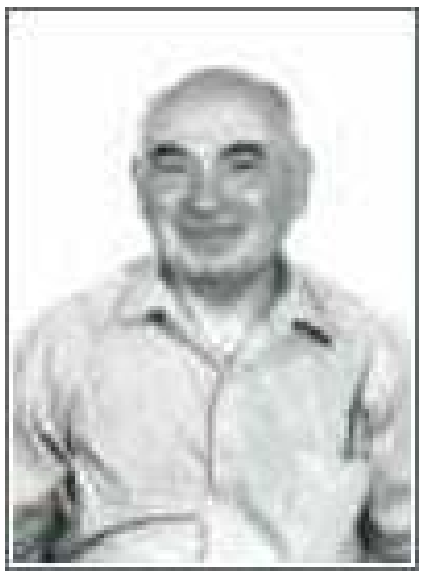
In 5D, there are only 2,002 items to search for.
Can't we find a quadrature rule of roughly that order?

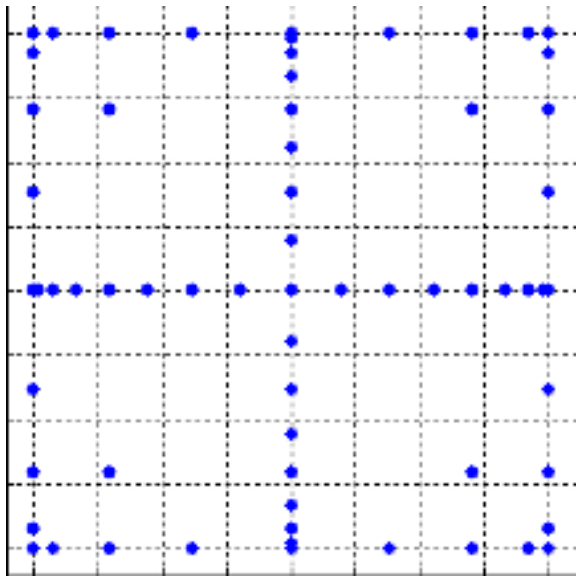
Sparse Grids for Stochastic Integrals

1. Integrals Arising from Stochastic Problems
2. Quadrature Rules
3. Product Rules for Higher Dimensions
4. **Smolyak Quadrature**
5. Covering Pascal's Triangle
6. How Grids Combine
7. Sparse Grids in Action
8. Smoothness is Necessary
9. A Stochastic Diffusion Equation
10. Software
11. Conclusion

SMOLYAK QUADRATURE

Sergey Smolyak (1963) suggested **sparse grids**:





- an algebraic combination of low order product grids;
- Pascal's precision diagonals achieved with far fewer points;

Smooth $f(x)$ + precision $\Rightarrow$ accuracy + efficiency.

SMOLYAK QUADRATURE

Very simply, the idea of Smolyak quadrature is to try to keep the order low for a given precision request. To do this, you need to use product rules that only "catch" the monomials you're interested in.

Say we want to catch all monomials with total degree 5 or less. When we want to catch x^5, we don't want to pick up x^5y and x^5y^2 on up to x^5y^5. So we use a product rule of precision 5 in x, and 0 in y.

To catch all the monomials we want, and no more, we combine product rules with the "appropriate" coefficients.

SMOLYAK QUADRATURE: Construction

We have an indexed family of 1D quadrature rules $\mathcal{Q}^L$.
We form rules for dimension **M**, indexed by level **L**.
Here $\mathbf{i} = i_1 + \cdots + i_M$, where i_j is the "level" of the j-th 1D rule.

$$\mathcal{A}(L,M) = \sum_{L-M+1 \le |\mathbf{i}| \le L} (-1)^{L+M-|\mathbf{i}|} \left(\begin{array}{c} L+M \\ L+M-|\mathbf{i}| \end{array} \right) (\mathcal{Q}^{i_1} \otimes \cdots \otimes \mathcal{Q}^{i_M})$$

Thus, the rule $\mathcal{A}(L,M)$ is a weighted sum of product rules.

SMOLYAK QUADRATURE: A sum of rules/a rule of sums

The Smolyak construction rule can be interpreted to say:

Compute the integral estimate for each rule,
then compute the algebraic sum of these estimates.

but it can also be interpreted as:

Combine the component rules into a single quadrature rule,
the new abscissas are the set of the component abscissas;
the new weights are the component weights multiplied by the sparse grid coefficient.

SMOLYAK QUADRATURE: Efficiency from Nesting

Under the second interpretation, we can see that in cases where an abscissa is duplicated in the component rules, the combined rule can use a single copy of the abscissa, with the sum of the weights associated with the duplicates.

Duplication is a property inherited from the 1D rules.

Duplication is useful when computing a single sparse grid rule, but also when computing a sequence of sparse grids of increasing level. In some cases, all the values from the previous level can be reused.

SMOLYAK QUADRATURE: Using Clenshaw-Curtis

A common choice is 1D Clenshaw-Curtis rules.

We can make a nested family by choosing successive orders of 1, 3, 5, 9, 17, ...

We wrote $\mathcal{Q}^i$ to indicate the 1D quadrature rules indexed by a **level** running 0, 1, 2, 3, and so on.

We will use a plain Q_n to mean the 1D quadrature rules of **order** 1, 3, 5, 9 and so on.

We will find it helpful to count abscissas.

SMOLYAK QUADRATURE: Using Clenshaw-Curtis

Theorem

The Clenshaw-Curtis Smolyak formula of level **L** *is precise for all polynomials of degree* $2 * L + 1$ *or less.*

Thus, although our construction of sparse grids seems complicated, we still know the level of precision we can expect at each level.

SMOLYAK QUADRATURE: Precision

Level	1D abscissas	5D abscissas	10D abscissas	Precision
0	1	1	1	1
1	3	11	21	3
2	5	61	221	5
3	9	241	1581	7
4	17	801	8801	9
5	33	2433	41265	11
6	65	6993	171425	13

Recall 5D product rule required 100,000 abscissas to integrate 2,002 entries in Pascal's precision triangle (precision 9).

SMOLYAK QUADRATURE: Asymptotic Accuracy

Let N be the order (number of abscissas) in the rule $A(L, M)$.

let I be the integral of $f(x)$,
$f(x) : [-1, 1]^M \rightarrow R | D^\alpha$ continuous if $\alpha_i \leq r$ for all i;

The accuracy for a Smolyak rule based on a nested family satisfies:

$$||I - A(L, M)|| = O(N^{\frac{-r}{\log(2M)}})$$

This behavior is near optimal; no family of rules could do better than $O(N^{-r})$ for this general class of integrands.

SMOLYAK QUADRATURE: Efficiency

The space of **M**-dimensional polynomials of degree **P** or less has dimension $\left(\begin{array}{c} P+M \\ M \end{array} \right) \approx \frac{M^P}{P!}$.

For large **M**, a Clenshaw-Curtis Smolyak rule that achieves precision **P** uses $N \approx \frac{(2M)^P}{P!}$ points.

Thus, if we are seeking exact integration of polynomials, the Clenshaw-Curtis Smolyak rule uses an optimal number of points (to within a factor 2^P that is independent of **M**).

Notice there is no exponent of **M** in the point growth.

1. Integrals Arising from Stochastic Problems
2. Quadrature Rules
3. Product Rules for Higher Dimensions
4. Smolyak Quadrature
5. **Covering Pascal's Triangle**
6. How Grids Combine
7. Sparse Grids in Action
8. Smoothness is Necessary
9. A Stochastic Diffusion Equation
10. Software
11. Conclusion

COVERING PASCAL'S TRIANGLE

A family of precise interpolatory rules must cover successive rows of Pascal's precision triangle in a regular way.

In higher dimensions, the triangle is a tetrahedron or a simplex.

The product rule does this by "overkill".

Smolyak's construction covers the rows, but does so much more economically, using lower order product rules.

COVERING PASCAL'S TRIANGLE

Let's watch how this works for a family of 2D rules.

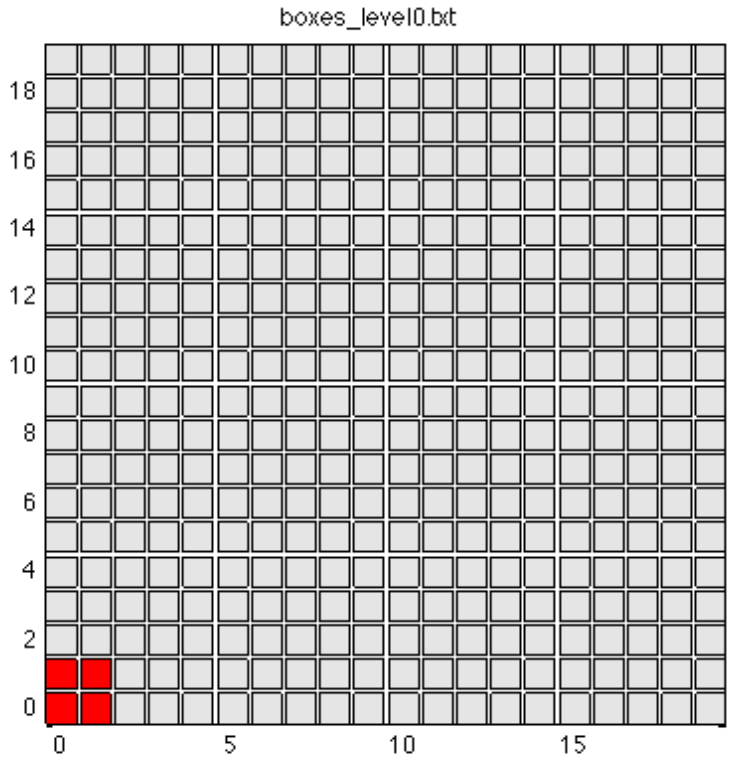
I've had to turn Pascal's triangle sideways, to an XY grid. If we count from 0, then box (I,J) represents $x^i y^j$.

Thus a row of Pascal's triangle is now a **diagonal** of this plot.

The important thing to notice is the maximum diagonal that is completely covered. This indicates the precision of the rule.

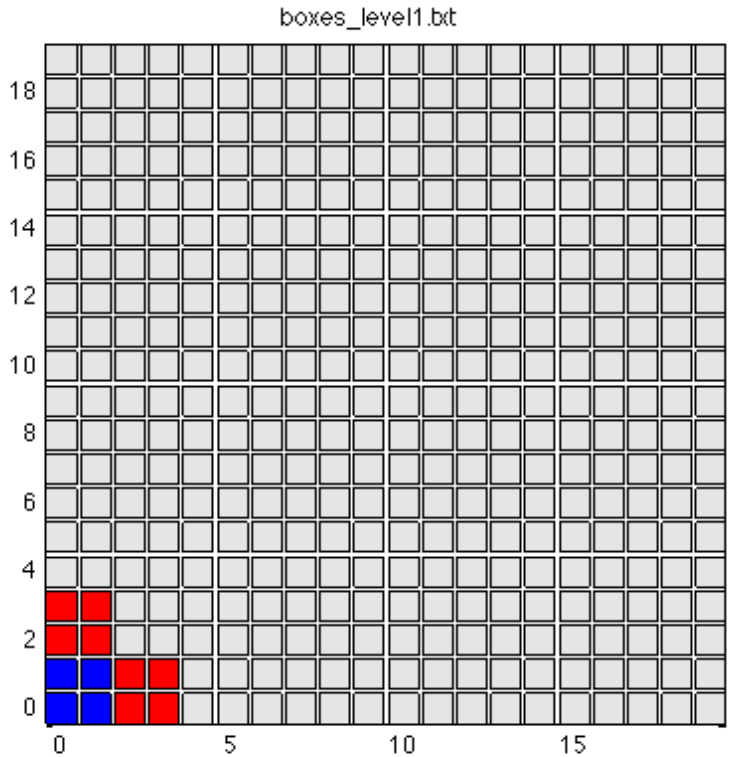
We will see levels 0 through 4 and precisions 1, 3, 5, 7 and 9.

COVERING PASCAL'S TRIANGLE: 2D Level 0



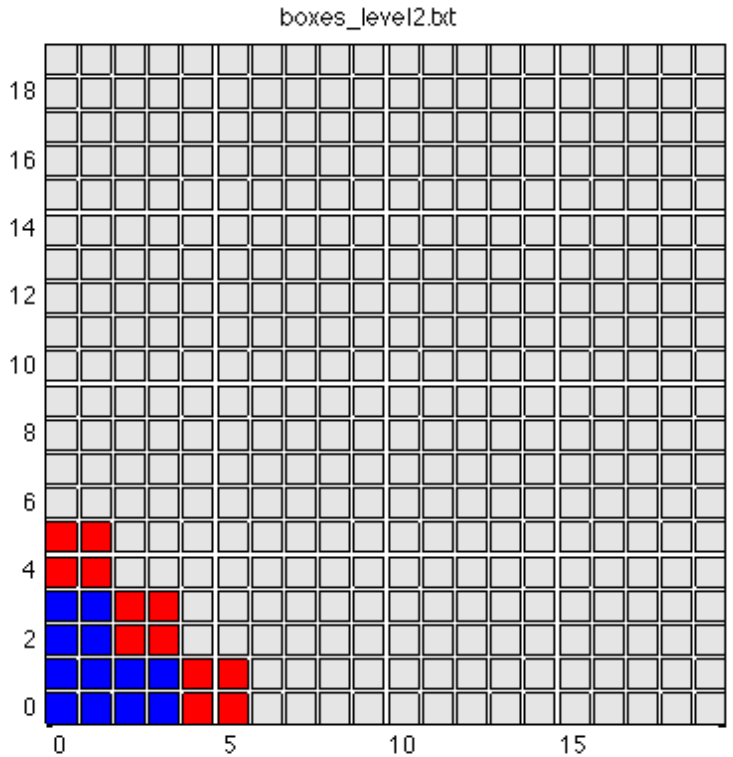
$Q_1 \otimes Q_1$

COVERING PASCAL'S TRIANGLE: 2D Level 1



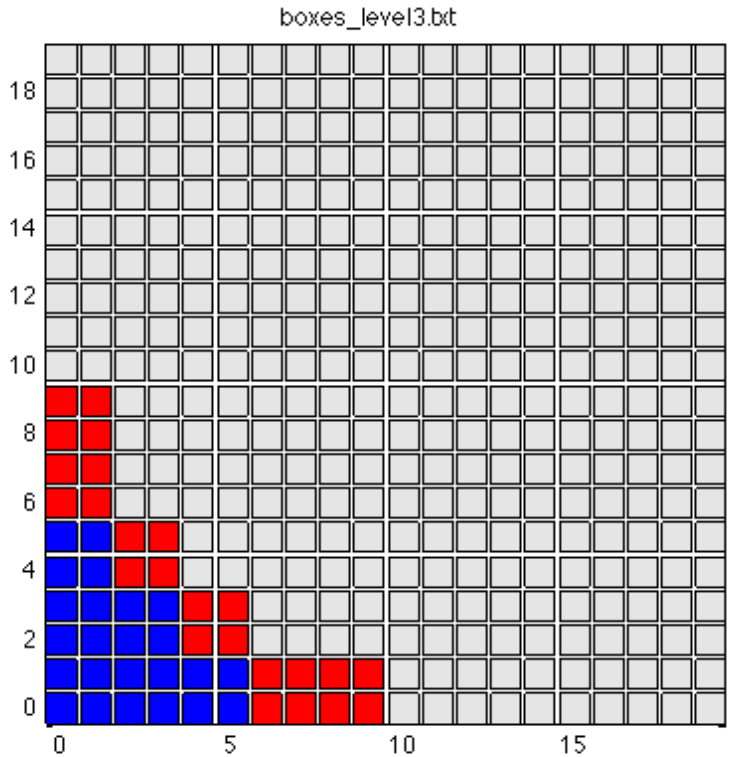
$Q_3 \otimes Q_1 + Q_1 \otimes Q_3$ - old

COVERING PASCAL'S TRIANGLE: 2D Level 2



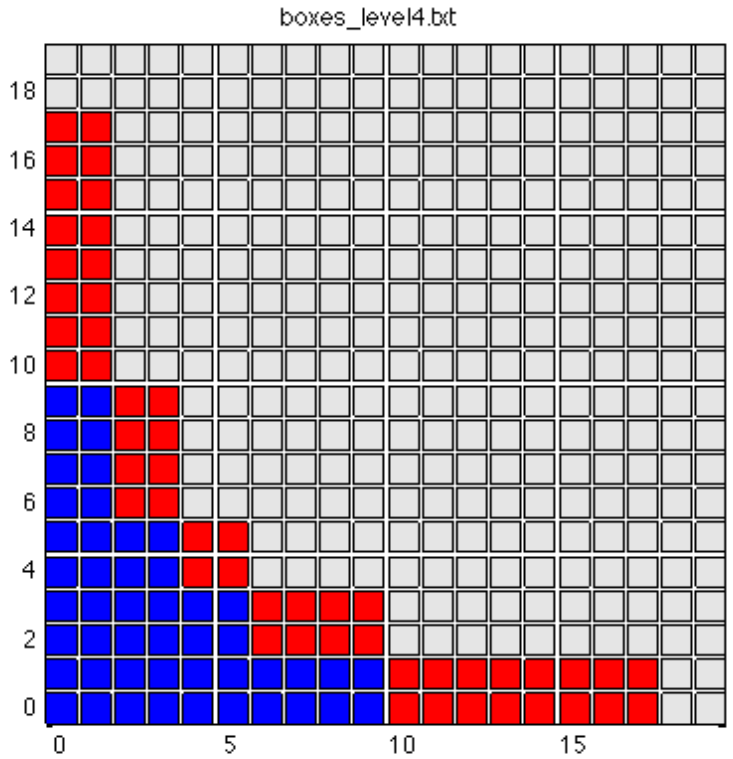
$Q_5 \otimes Q_1 + Q_3 \otimes Q_3 + Q_1 \otimes Q_5$ - old.

COVERING PASCAL'S TRIANGLE: 2D Level 3



$Q_9 \otimes Q_1 + Q_5 \otimes Q_3 + Q_3 \otimes Q_5 + Q_1 \otimes Q_9$ - old;

COVERING PASCAL'S TRIANGLE: 2D Level 4



$Q_{17} \otimes Q_1 + Q_9 \otimes Q_3 + Q_5 \otimes Q_5 + Q_3 \otimes Q_9 + Q_1 \otimes Q_{17}$ - old;

COVERING PASCAL'S TRIANGLE: Comments

When based on an exponential growth rule like $N = 2^L + 1$, each new level of a Smolyak family:

- covers 2 more diagonals, increasing precision by 2;
- avoids filling in the heavy "half" of the hypercube that the product rule fills;
- adds long but thin regions of excess levels along the axes;

The excess levels come about because we are trying to exploit nesting as much as possible. As the spatial dimension increases, the relative cost of the excess levels decreases.

COVERING PASCAL'S TRIANGLE: Comments

When based on a linear growth rule like $N = 2 * L + 1$, each new level of a Smolyak family:

- covers 2 more diagonals, increasing precision by 2;
- avoids filling in the heavy "half" of the hypercube that the product rule fills;
- has no wasted level coverage at all;

So for linear growth, every level is "useful" (no excess levels at all) but because linear growth interferes with nesting, we probably increase the number of quadrature points needed. (So we buy only what we need, but each item costs a bit more.) As the spatial dimension increases, the relative cost of not using nesting increases.

Sparse Grids for Stochastic Integrals

1. Integrals Arising from Stochastic Problems
2. Quadrature Rules
3. Product Rules for Higher Dimensions
4. Smolyak Quadrature
5. Covering Pascal's Triangle
6. **How Grids Combine**
7. Sparse Grids in Action
8. Smoothness is Necessary
9. A Stochastic Diffusion Equation
10. Software
11. Conclusion

HOW GRIDS COMBINE

We said that the Smolyak construction combines low order product rules, and that the result can be regarded as a single rule.

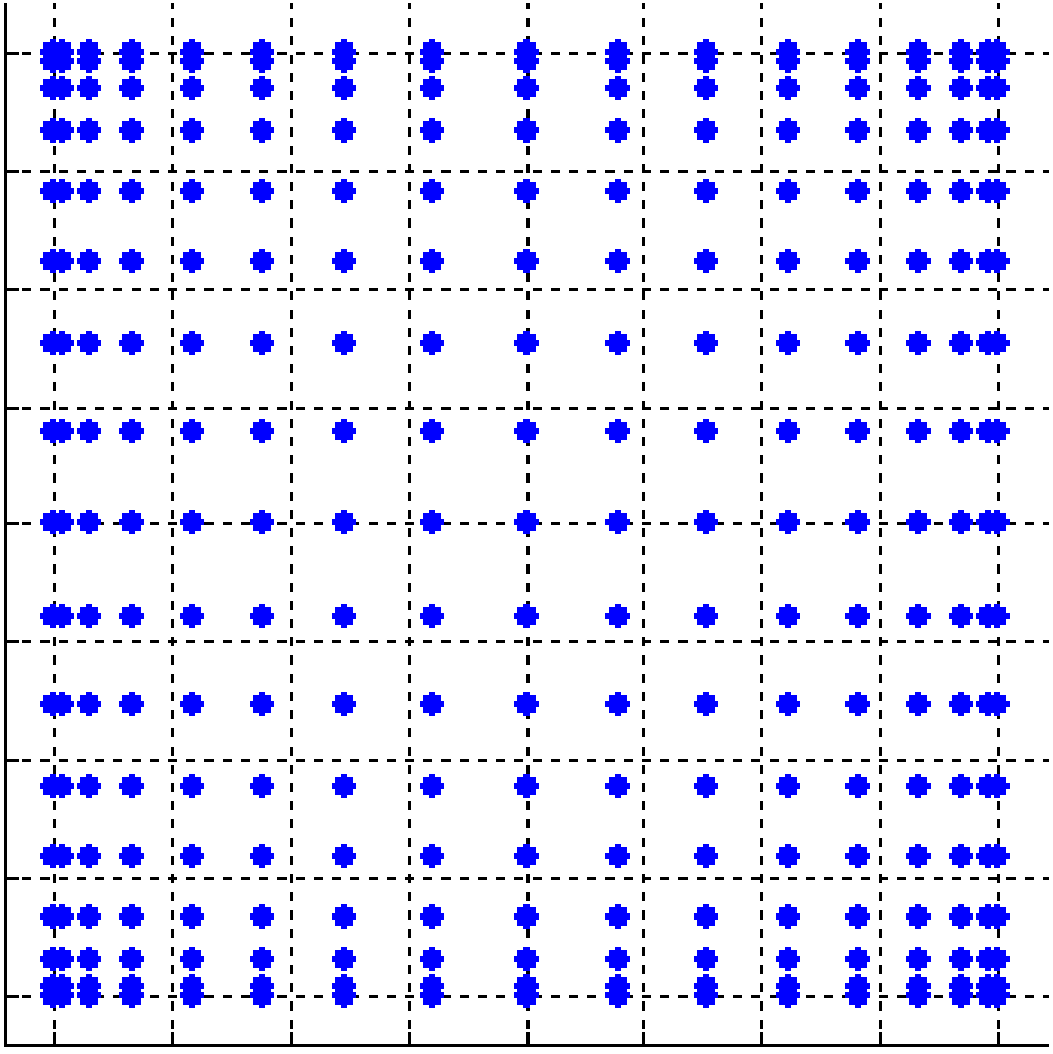
Let's look at the construction of the Smolyak grid of level **L**=4 and hence precision **P**=9 in 2D.

Our construction will involve 1D rules of orders 1, 3, 5, 9 and 17, and product rules formed of these factors.

Because of nesting, every product rule we form will be a subset of the 17x17 full product grid.

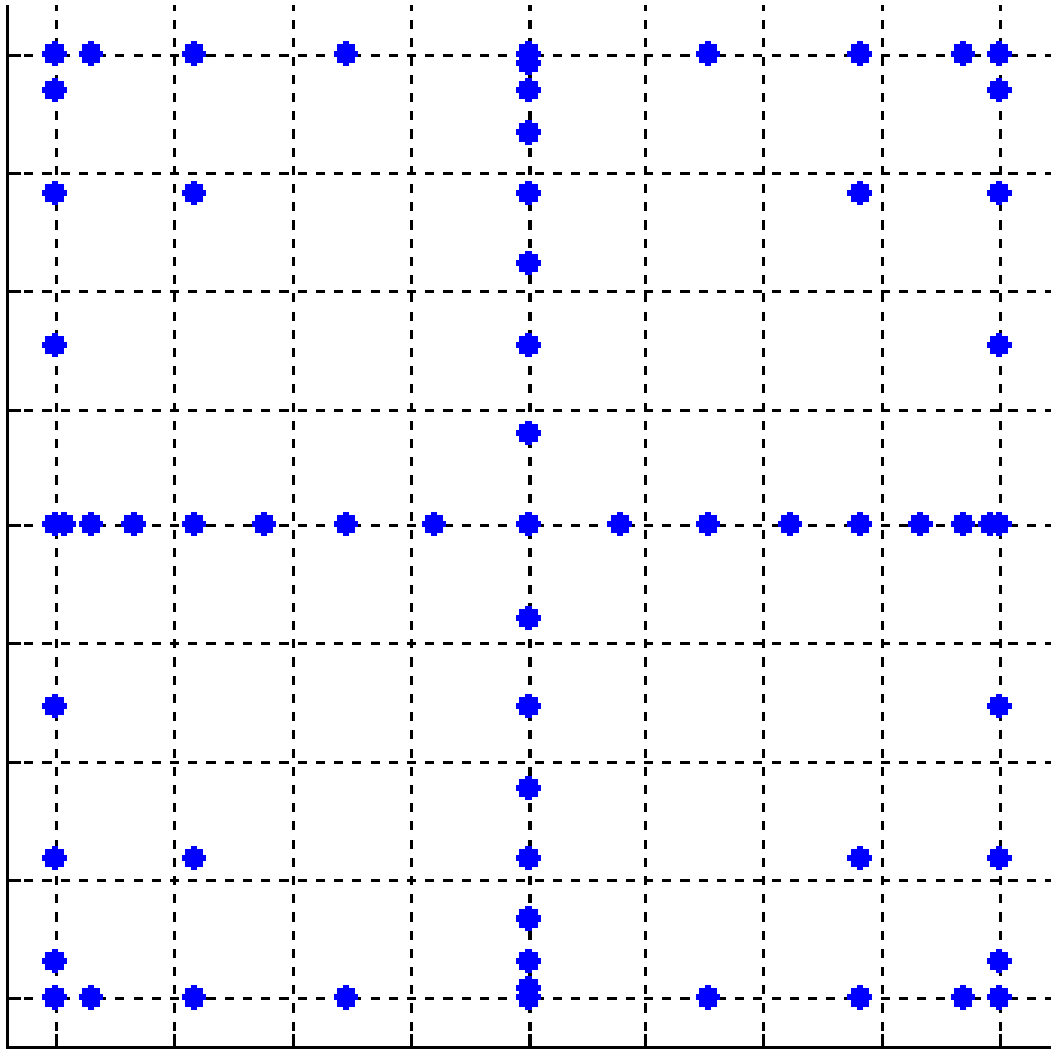
HOW GRIDS COMBINE: 2D Order 17 Product Rule

A 17x17 Clenshaw-Curtis product grid (289 points).



HOW GRIDS COMBINE: 2D Level4 Smolyak Grid

The sparse grid is a subset of the 17x17 product grid (65 points).



HOW GRIDS COMBINE: 2D Level4 Smolyak Grid

It's easy to get the impression that the 65-point sparse grid can replace the 289 point product grid - that is, that it has the same precision. That's not true!

A 17x17 product grid has precision 16 (actually 17, by symmetry). The Clenshaw Curtis sparse grid of level 4, with 65 points, has precision 9.

We should compare it to the product grid of 9x9=81 points which also has precision 9. We see that there's not much difference in order!

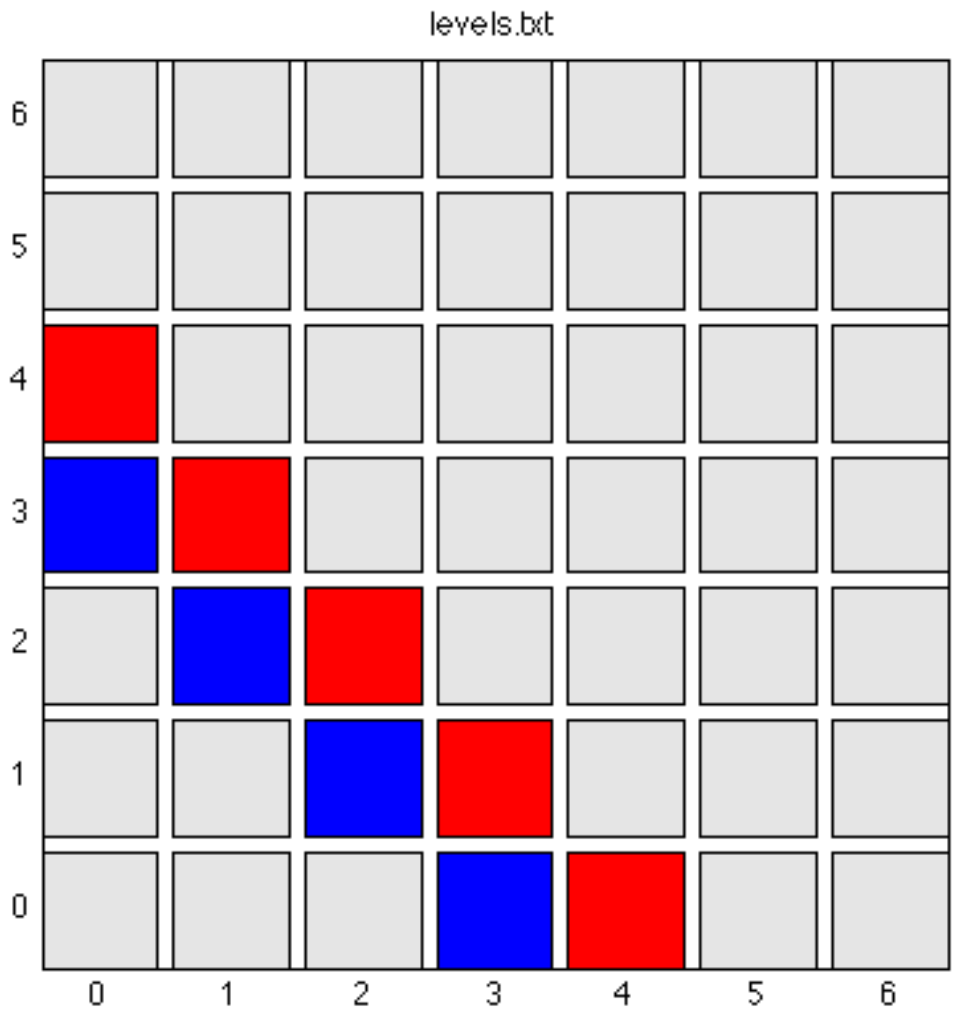
Sparse grids in low dimensions (2 or 3) are not competitive with product grids. We use them as examples because they are possible to plot!

HOW GRIDS COMBINE

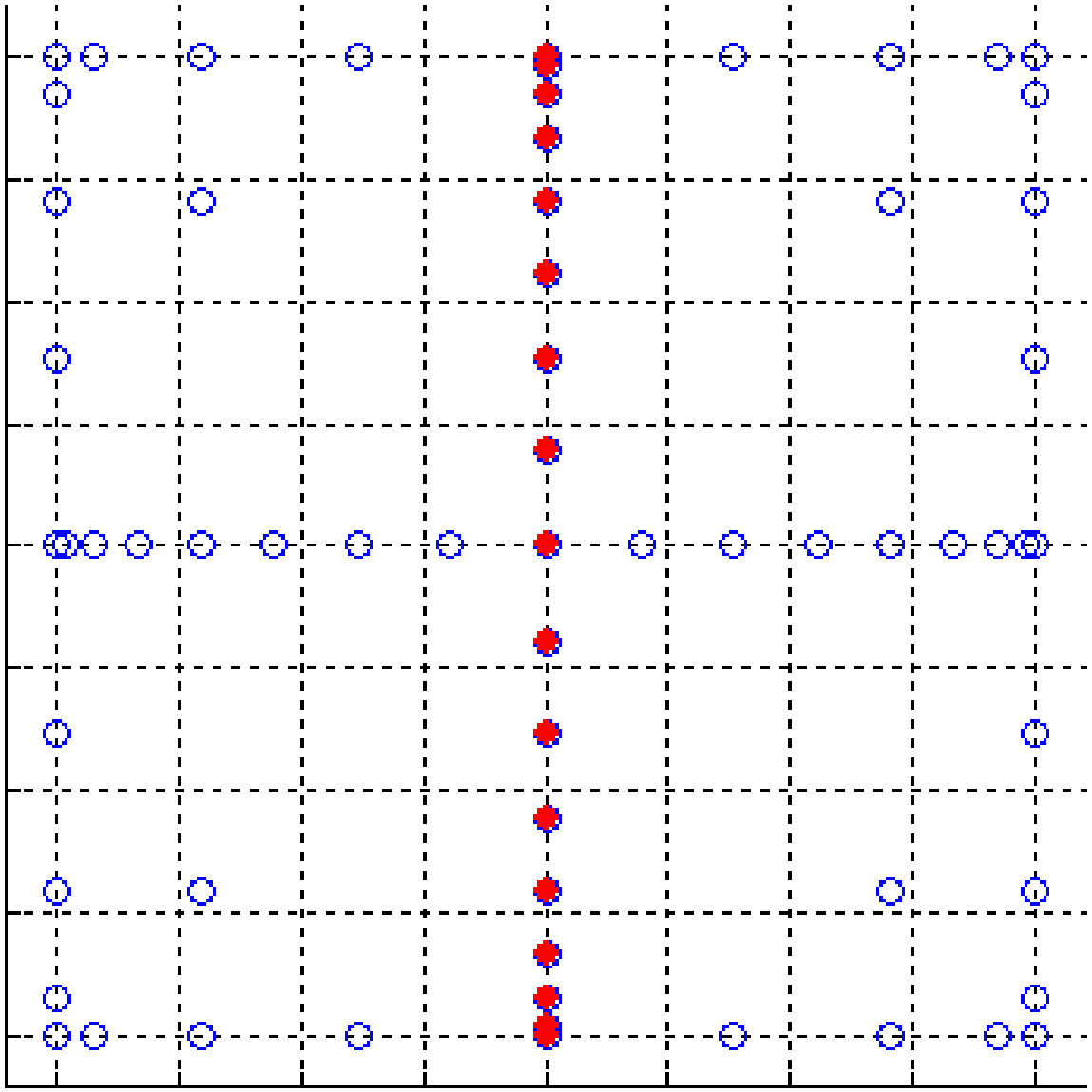
$$\mathcal{A}(4,2) = \sum_{3 \le |\mathbf{i}| \le 4} (-1)^{4-|\mathbf{i}|} \begin{pmatrix} -1 \\ 4-|\mathbf{i}| \end{pmatrix} (\mathcal{Q}^{i_1} \otimes \mathcal{Q}^{i_2})$$

$$\begin{aligned}
= &+ \mathcal{Q}^0 \otimes \mathcal{Q}^4 \quad (Q_1 \otimes Q_{17}) \\
&+ \mathcal{Q}^1 \otimes \mathcal{Q}^3 \quad (Q_3 \otimes Q_9) \\
&+ \mathcal{Q}^2 \otimes \mathcal{Q}^2 \quad (Q_5 \otimes Q_5) \\
&+ \mathcal{Q}^3 \otimes \mathcal{Q}^1 \quad (Q_9 \otimes Q_3) \\
&+ \mathcal{Q}^4 \otimes \mathcal{Q}^0 \quad (Q_{17} \otimes Q_1) \\
&- \mathcal{Q}^0 \otimes \mathcal{Q}^3 \quad (Q_1 \otimes Q_9) \\
&- \mathcal{Q}^1 \otimes \mathcal{Q}^2 \quad (Q_3 \otimes Q_5) \\
&- \mathcal{Q}^2 \otimes \mathcal{Q}^1 \quad (Q_5 \otimes Q_3) \\
&- \mathcal{Q}^3 \otimes \mathcal{Q}^0 \quad (Q_9 \otimes Q_1)
\end{aligned}$$

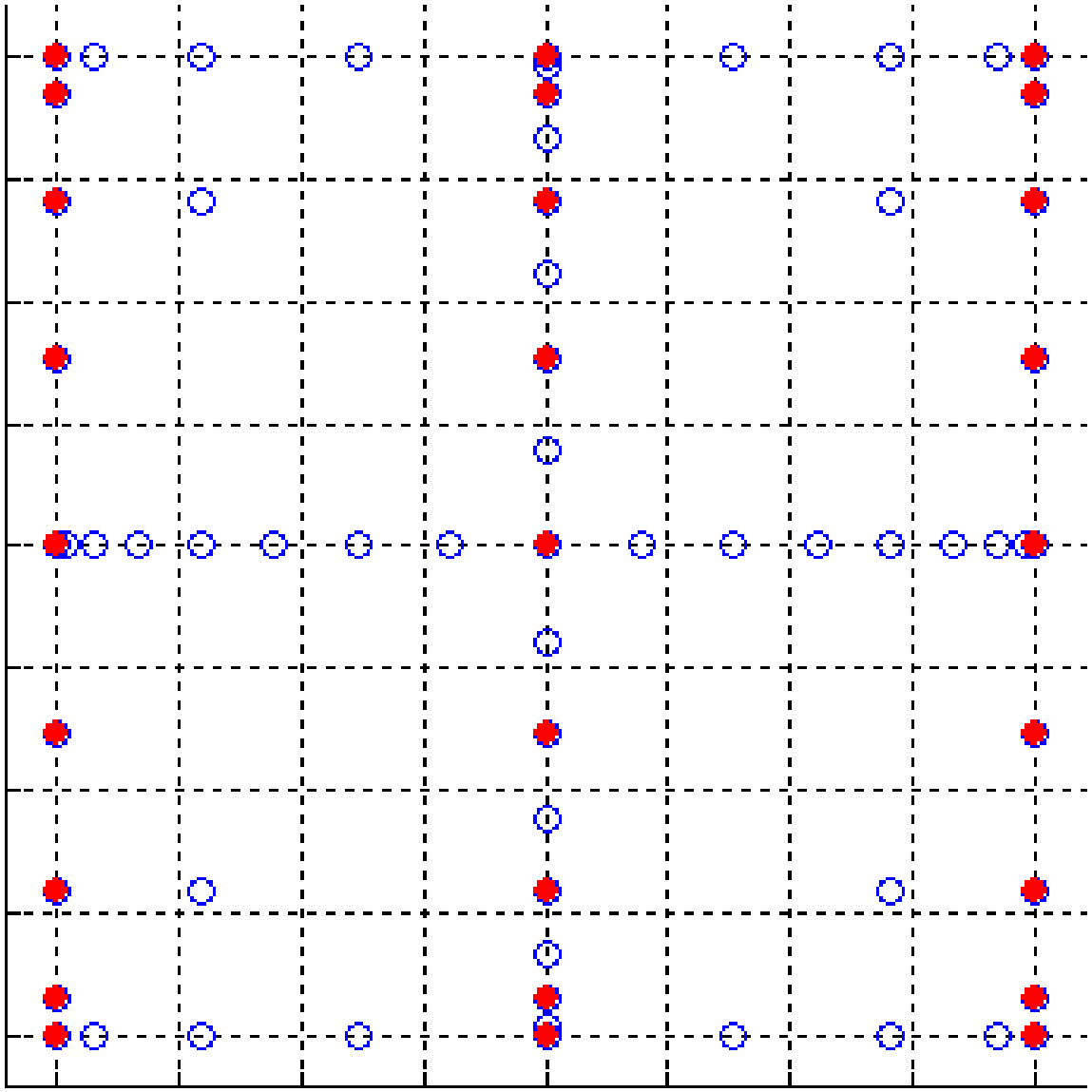
HOW GRIDS COMBINE: Red Rules - Blue Rules



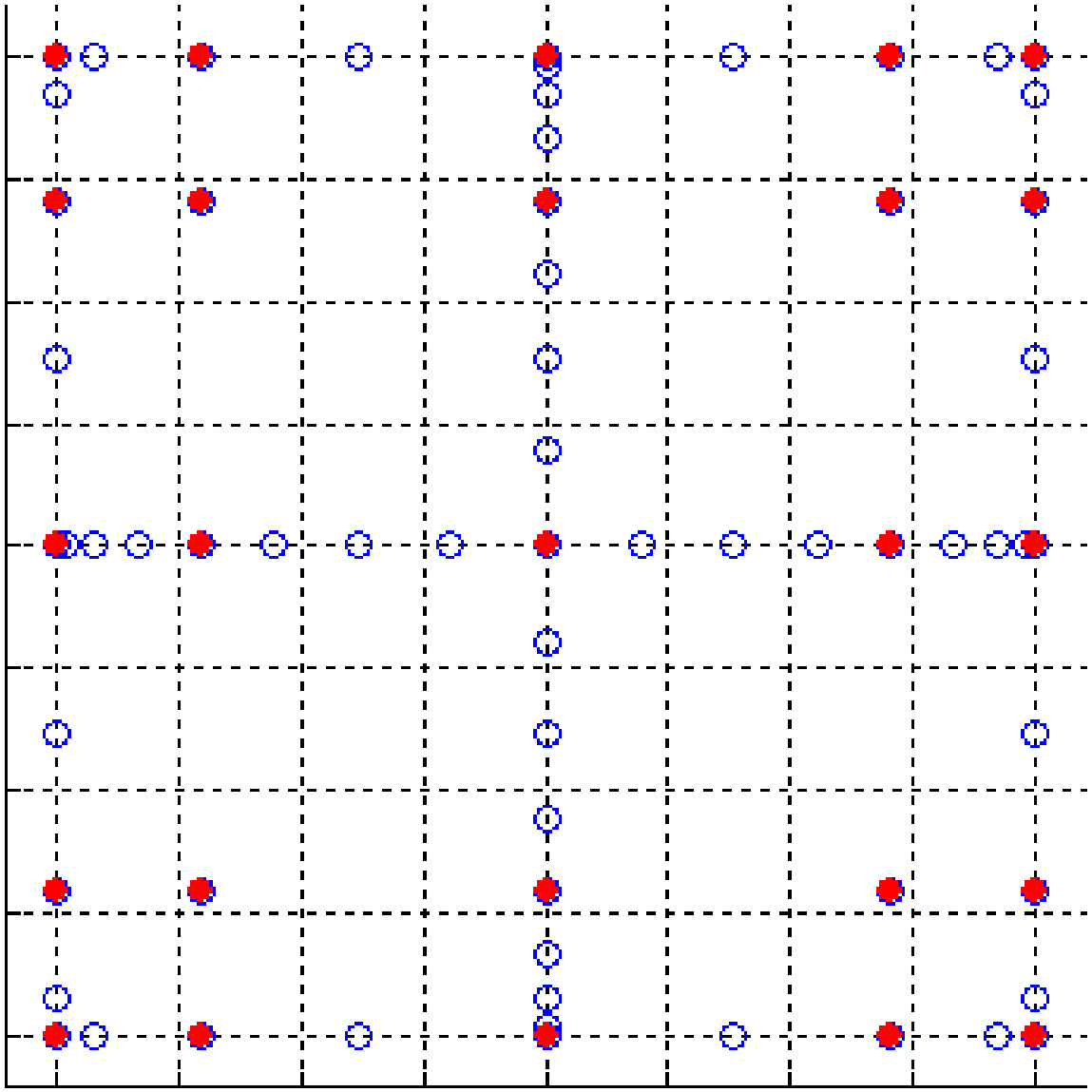
HOW GRIDS COMBINE: 2D Level4 1x17 component



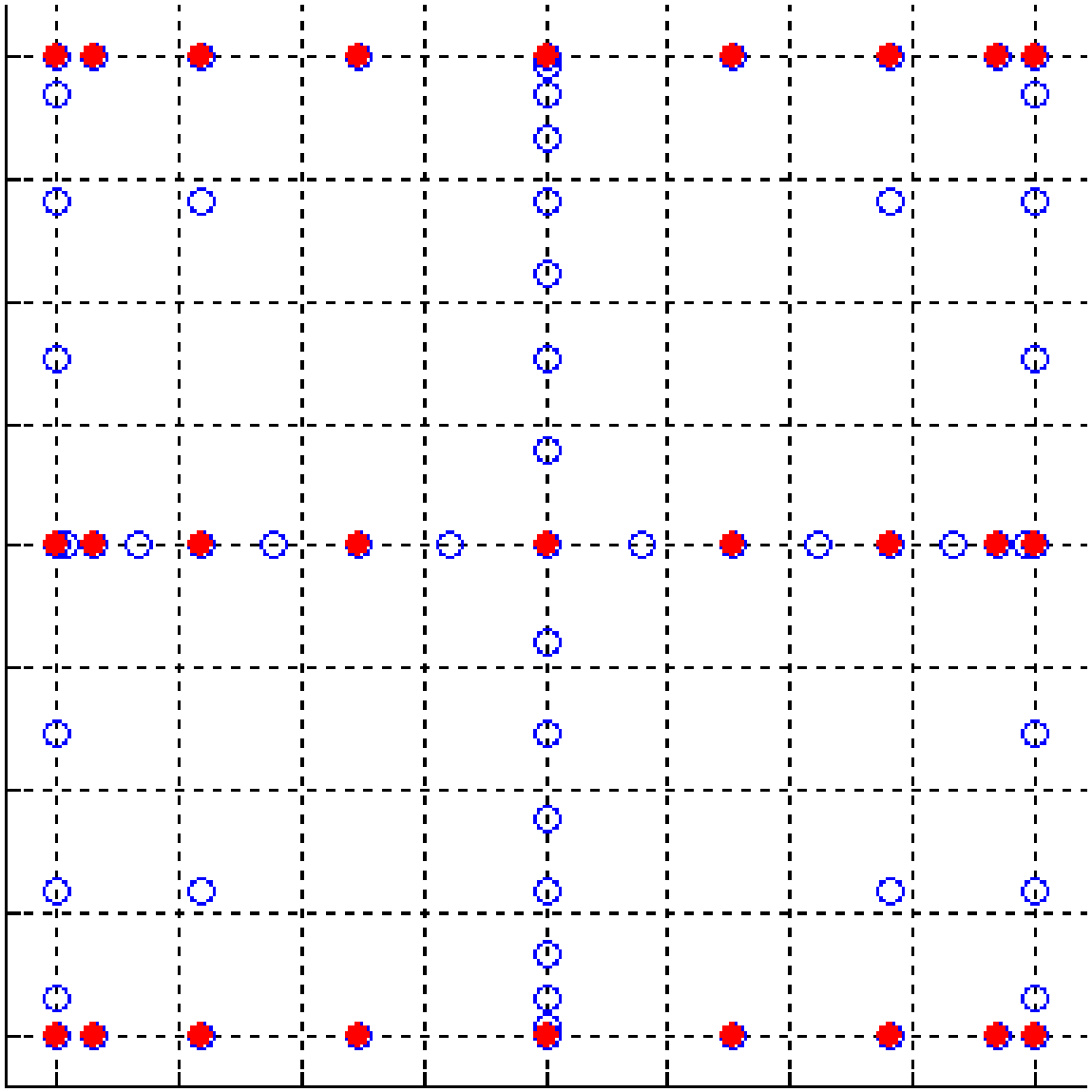
HOW GRIDS COMBINE: 2D Level4 3x9 component



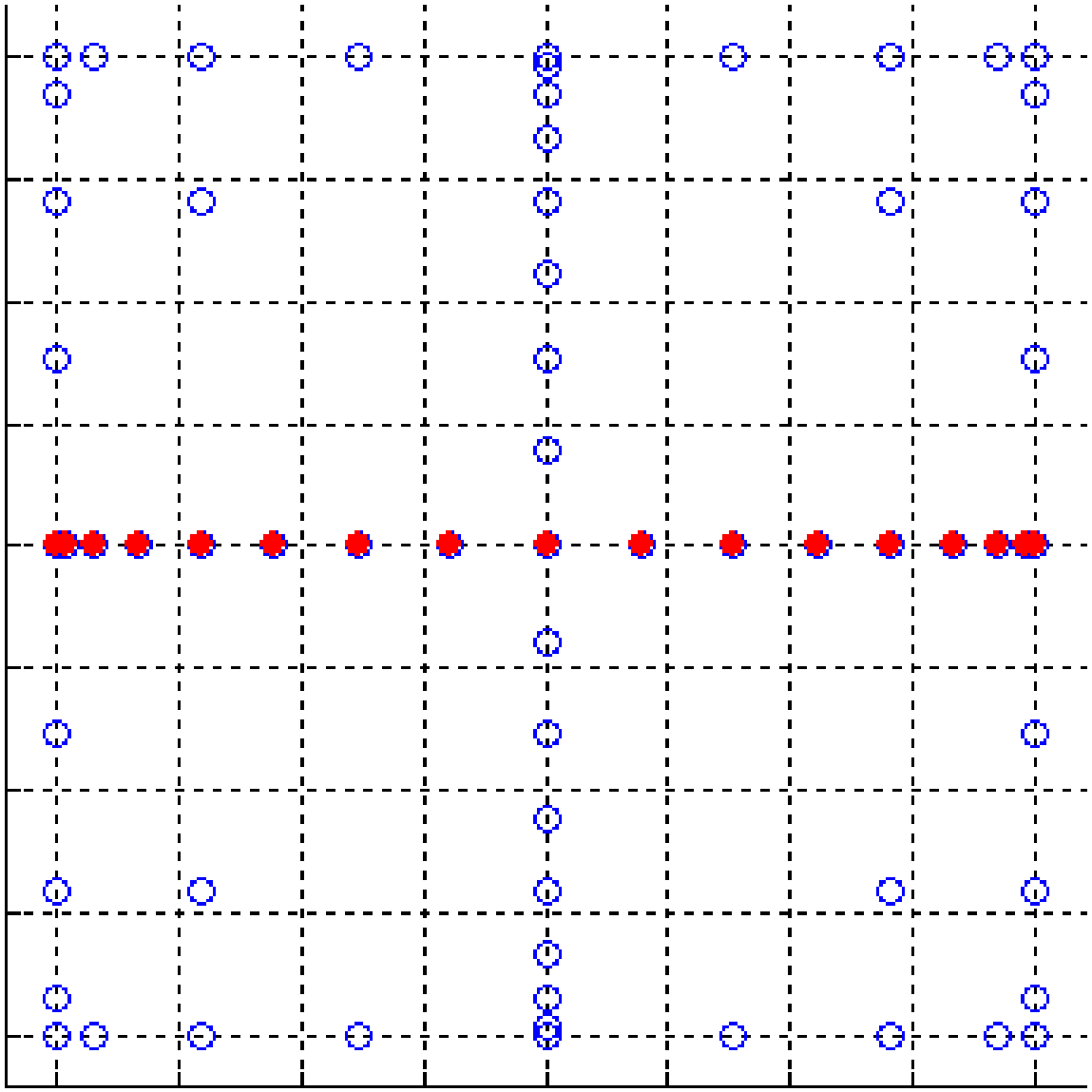
HOW GRIDS COMBINE: 2D Level4 5x5 component



HOW GRIDS COMBINE: 2D Level4 9x3 component



HOW GRIDS COMBINE: 2D Level4 17x1 component



HOW GRIDS COMBINE: Red Rules - Blue Rules

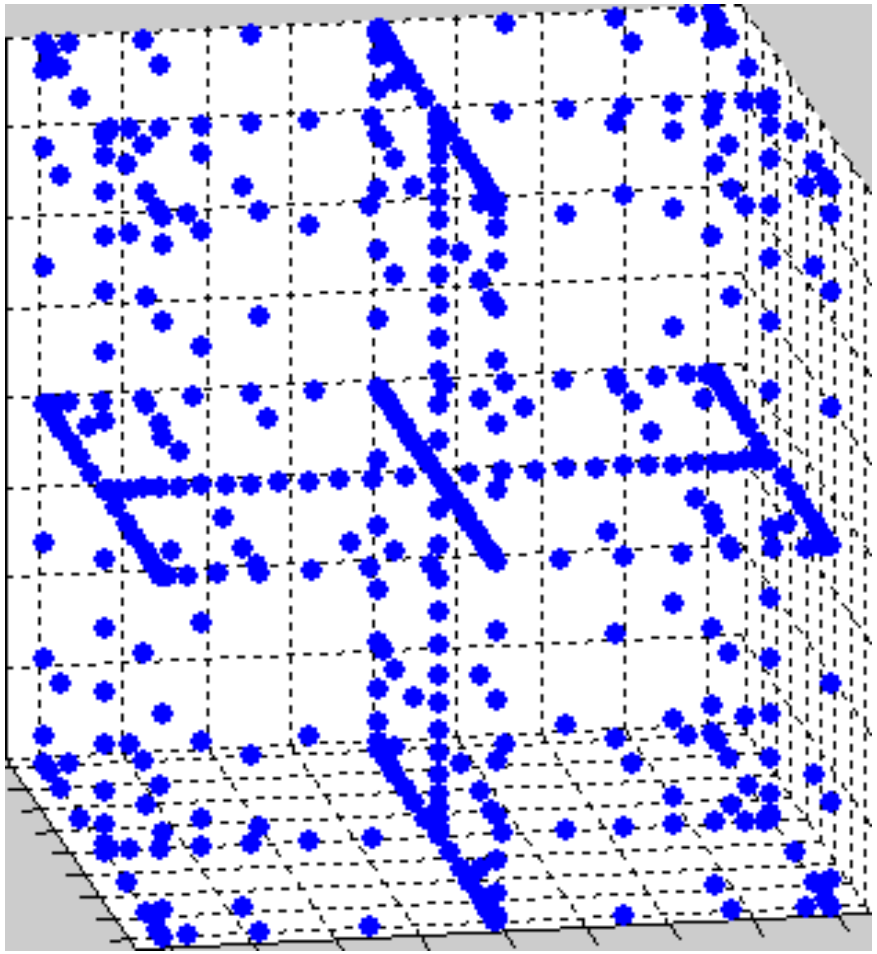
We've shown the component "red" rules, which show up in the sum with a positive sign.

The 'blue" rules are similar, though at a lower level:

- The first of the 5 red rules has order 1x17;
- The first of the 4 blue rules has order 1x9.

Notice that this rule is "symmetric" in all dimensions. If we have a 65x3 rule, we are also guaranteed a 3x65 rule. The Smolyak formula is **isotropic**.

HOW GRIDS COMBINE: 3D Level5 Smolyak Grid



3D sparse grid, level 5, precision 11 uses 441 abscissas;

3D product grid of precision 11 uses 1,331 abscissas.

Sparse Grids for Stochastic Integrals

1. Integrals Arising from Stochastic Problems
2. Quadrature Rules
3. Product Rules for Higher Dimensions
4. Smolyak Quadrature
5. Covering Pascal's Triangle
6. How Grids Combine
7. **Sparse Grids in Action**
8. Smoothness is Necessary
9. A Stochastic Diffusion Equation
10. Software
11. Conclusion

SPARSE GRIDS IN ACTION

Let's take a problem that's reasonable but not trivial.

We'll work in a space with dimension $\mathbf{M} = 6$.

We'll try to integrate the **Genz Product Peak**:

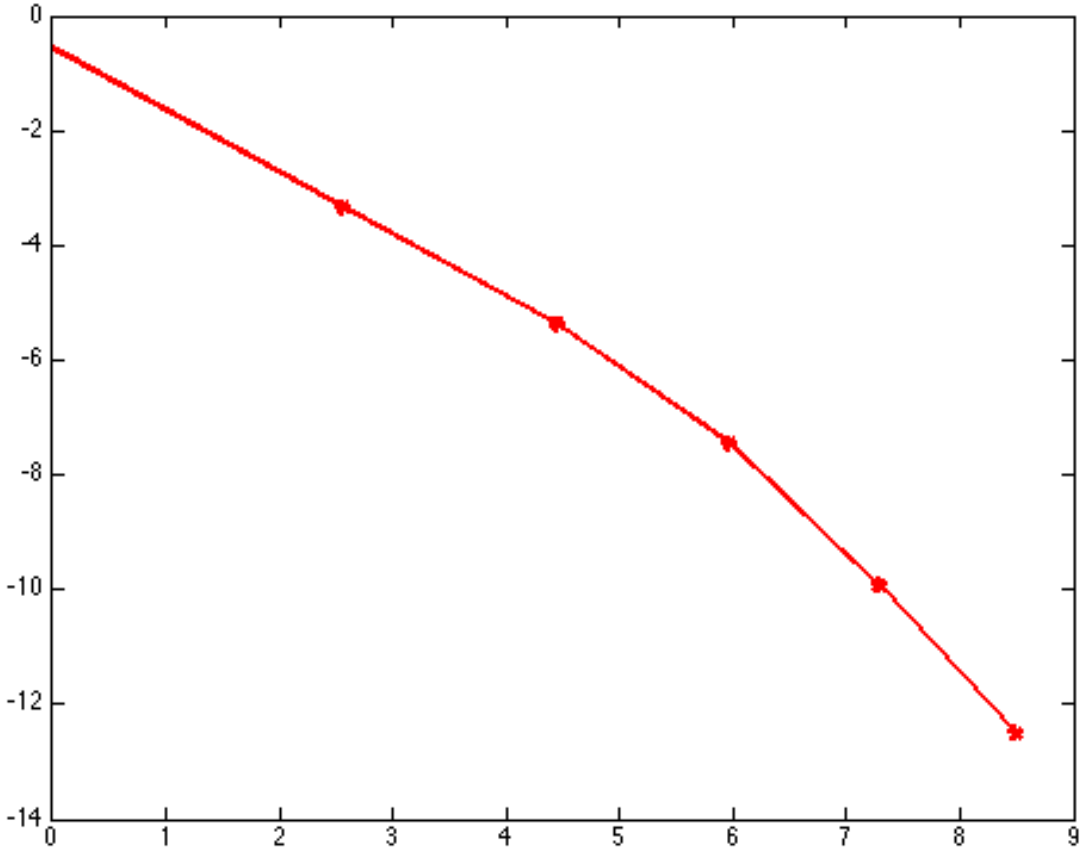
$$f(X) = \frac{1}{\prod_{i=1}^{M}(C_i^2 + (X_i - Z_i)^2)}$$

where C_i and Z_i are prescribed.

SPARSE GRIDS IN ACTION: 6D Smolyak

Level	Order	Estimate	Error
0	1	0.062500	0.573282
1	13	0.600000	0.0357818
2	85	0.631111	0.00467073
3	389	0.636364	0.000582152
4	1457	0.635831	0.0000492033
5	4865	0.635778	0.00000375410
∞	∞	0.635782	0.0000

SPARSE GRIDS IN ACTION:6D Smolyak



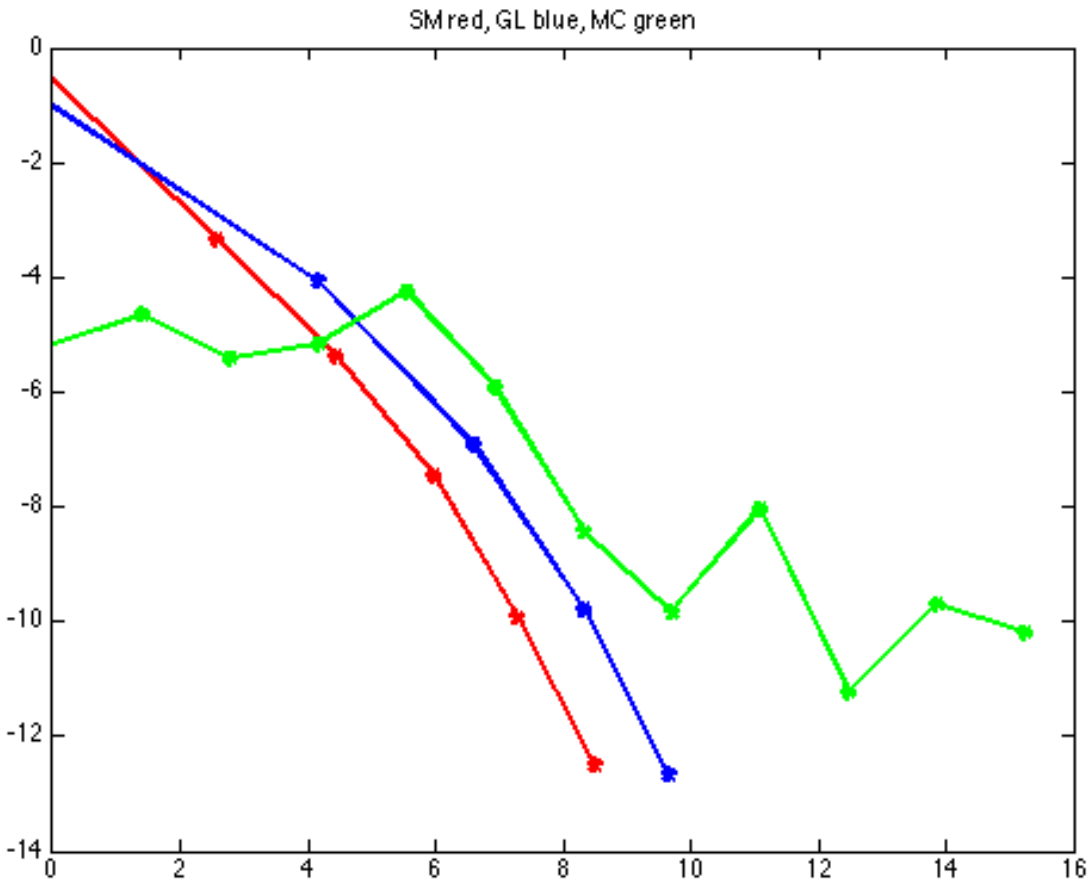
SPARSE GRIDS IN ACTION: 6D Gauss-Legendre

1D Order	6D Order	Estimate	Error
1	1	1.00000	0.364218
2	64	0.618625	0.0171570
3	729	0.636774	0.000992123
4	4096	0.635726	0.0000560162
5	15625	0.635785	0.00000314963
∞	∞	0.635782	0.0000

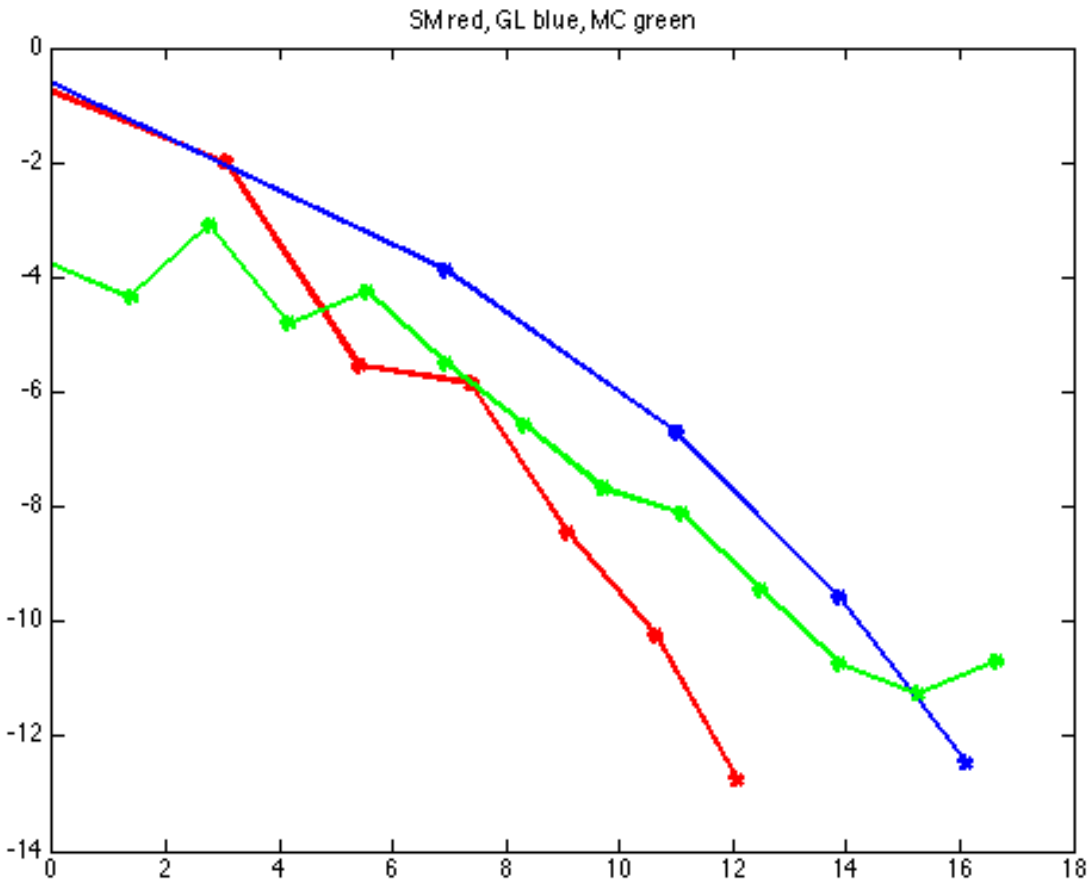
SPARSE GRIDS IN ACTION: 6D Monte Carlo

log2 (N)	N	Estimate	Error
0	1	0.641468	0.00568631
4	16	0.640218	0.00443594
8	256	0.650114	0.0143321
16	4096	0.636000	0.000218054
24	65536	0.636105	0.000323117
32	1048576	0.635843	0.0000612090
∞	∞	0.635782	0.0

SPARSE GRIDS IN ACTION: 6D Smolyak/GL/MC



SPARSE GRIDS IN ACTION: 10D Smolyak/GL/MC



SPARSE GRIDS IN ACTION: Thoughts

The graphs suggests that the accuracy behavior of the sparse grid rule is similar to the Gauss-Legendre rule, at least for this kind of integrand.

For 6 dimensions, the sparse grid rule is roughly 3 times as efficient as Gauss-Legendre, (4,865 abscissas versus 15,625 abscissas).

Moving from 6 to 10 dimensions, the efficiency advantage is 60: (170,000 abscissas versus 9,700,000 abscissas).

The Gauss-Legendre product rule is beginning the explosive growth in abscissa count.

Sparse Grids for Stochastic Integrals

1. Integrals Arising from Stochastic Problems
2. Quadrature Rules
3. Product Rules for Higher Dimensions
4. Smolyak Quadrature
5. Covering Pascal's Triangle
6. How Grids Combine
7. Sparse Grids in Action
8. **Smoothness is Necessary**
9. A Stochastic Diffusion Equation
10. Software
11. Conclusion

Smoothness: A Few Words of Wisdom

A sparse grid approach is the right choice when the function to be integrated is known to be smooth or to have bounded derivatives up to the order of the rule we are applying.

In those cases, the precision of a sparse grid extracts extra information from the function values, to provide accurate answers with efficiency.

But if the smoothness assumption is not true, the sparse grid approach will fail.

Smoothness: Characteristic Function of 6D Sphere

In the region $[-1,+1]^6$, define

$$f(x) = \begin{cases} 1, & \text{if } \|x\| \leq 1; \\ 0, & \text{if } \|x\| > 1. \end{cases}$$

This function is not even continuous, let alone differentiable. We will try to apply a series of Clenshaw Curtis sparse grids to this integrand.

The hypercube volume is 64;
the hypersphere volume is $\frac{\pi^3}{6} \approx 5.16771$.

Smoothness: Sparse Grid Quadrature

N	SG Estimate	SG Error	:	MC Estimate	MC Error
1	4.000	1.167	:	...	...
13	64.000	58.832	:	...	...
85	-42.667	-47.834	:	...	...
389	-118.519	-123.686	:	...	...
1457	148.250	143.082	:	...	...
4865	-24.682	-29.850	:	...	...

Can you see why negative estimates are possible even though the integrand is never negative?

Smoothness: MC Quadrature

N	SG Estimate	SG Error	:	MC Estimate	MC Error
1	4.000	1.167	:	0.00000	5.16771
13	64.000	58.832	:	0.00000	5.16771
85	-42.667	-47.834	:	3.01176	2.15595
389	-118.519	-123.686	:	4.77121	0.39650
1457	148.250	143.082	:	5.16771	0.01555
4865	-24.682	-29.850	:	5.41994	0.25226

Here, we make the Monte Carlo method look like a quadrature rule with equal weights.

Smoothness: MC Quadrature

So how far do we have to go to get 3 digits correct?

N	MC Estimate	MC Error
1	0.00000	5.16771
32	6.00000	0.83228
1,024	4.81250	0.35521
32,768	5.39063	0.22291
1,048,576	5.18042	0.01271
33,554,432	5.16849	0.00077
∞	5.16771	0.00000

The function values are only 0 or 1
the spatial dimension is "only" 6D...

...but 3 digit accuracy requires 33 million evaluations!

Smoothness: Adaptivity in Interpolation Order

A sparse grid **can** be used for this problem, if it uses rules that don't expect much more smoothness than the integrand has.

Here, we would want the underlying product grids to construct and integrate piecewise linear or piecewise constant interpolants.

This would be an example of **adaptivity**, that is, the ability of an algorithm to use its best method (high order polynomials) if it can, but to watch for danger signs and "gear down" to a slower but safer method when indicated.

Sparse Grids for Stochastic Integrals

1. Integrals Arising from Stochastic Problems
2. Quadrature Rules
3. Product Rules for Higher Dimensions
4. Smolyak Quadrature
5. Covering Pascal's Triangle
6. How Grids Combine
7. Sparse Grids in Action
8. Smoothness is Necessary
9. **A Stochastic Diffusion Equation**
10. Software
11. Conclusion

Stochastic Diffusion

$$-\nabla \cdot (a(\vec{x}) \nabla u(\vec{x})) = f(\vec{x})$$

$u(\vec{x})$ is an unknown quantity, like temperature;

$a(\vec{x})$ is a **known** physical property, the conductivity, which controls how quickly hot or cold spots average out.

- heat conduction;
- slow subsurface flow of water;
- particle diffusion;
- Black-Scholes equation (flow of money!).

Stochastic Diffusion: Uncertain Conductivity

Using a fixed value for $a(\vec{x})$ might be unrealistic.

Without variations in $a(\vec{x})$, we might never see the bumps and swirls typical of real physical problems.

We might think of $a(\vec{x})$ as a *random field* $a(\vec{x};\omega)$.

The ω represents the unknown variation from the average.

Stochastic Diffusion: Uncertain Solution

If $a(\vec{x};\omega)$ has an "unknown" component, then so does our solution, which we write $u(\vec{x};\omega)$.

$$-\nabla \cdot (a(\vec{x};\omega)\,\nabla u(\vec{x};\omega)) = f(\vec{x})$$

Now if we don't know what the equation is, we can't solve it!

Can we still extract information from the equation?

Stochastic Diffusion: Expected Values

Each variation ω determines a solution u.

If we added up every variation, we'd get an average or expected value for the solution.

The expected value is an important first piece of information about a problem with a random component.

$$E(u(\vec{x})) = \int_{\Omega} u(\vec{x};\omega)\,\rho(\omega)\,d\omega$$

It's like using weather records to estimate the *climate*.

Stochastic Diffusion: Approximate Integral

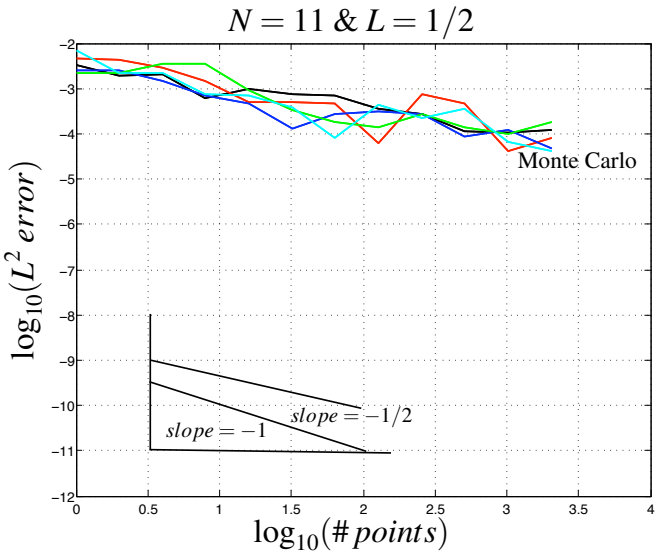
We approximate the function space Ω by an M-dimensional space Ω^M, of linear sums of perturbations ω.

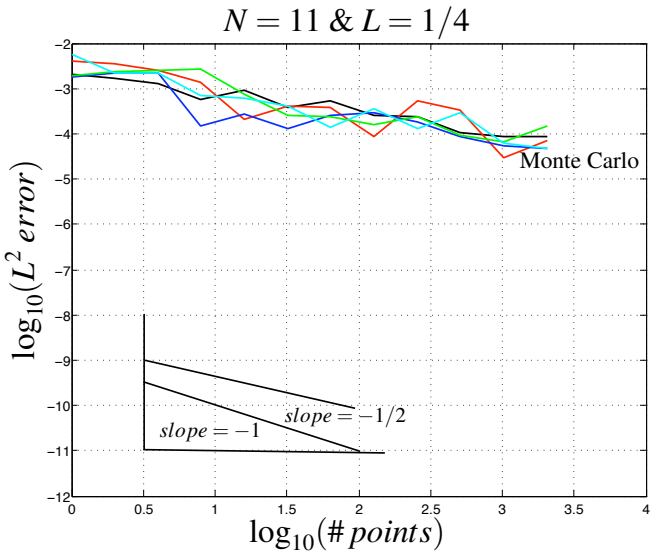
We now estimate the integral of $u(\vec{x};\omega)$ in Ω^M.

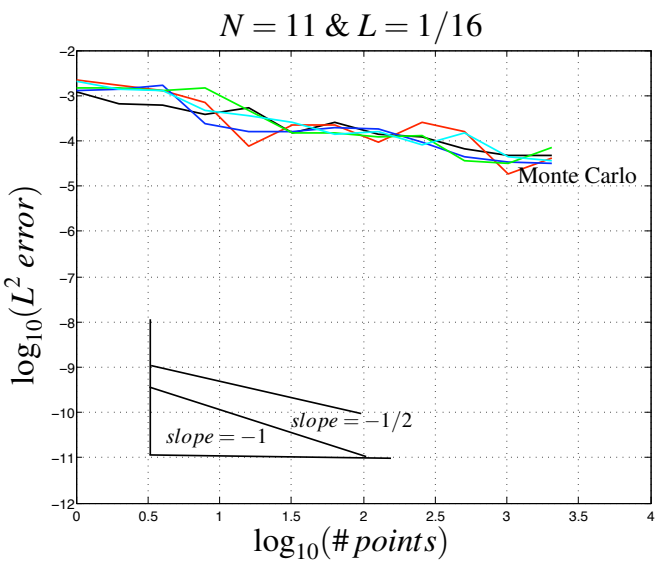
Monte Carlo: select random sets of parameters ω, (weighted by $\rho(\omega)$, solve for $u(\vec{x};\omega)$ and average.

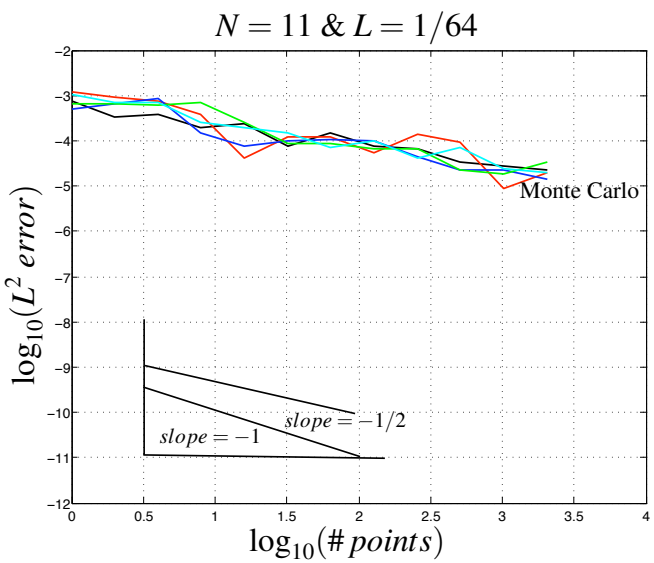
Sparse grid: choose a level, defining a grid of ω values in Ω^M. For each ω grid value, evaluate the spatial integrals to get a contribution to A and f; sum to get A and f, solve $A * x = f$ for the finite element coefficients for $E(u(\vec{x});\omega))$.

Stochastic Diffusion: Monte Carlo

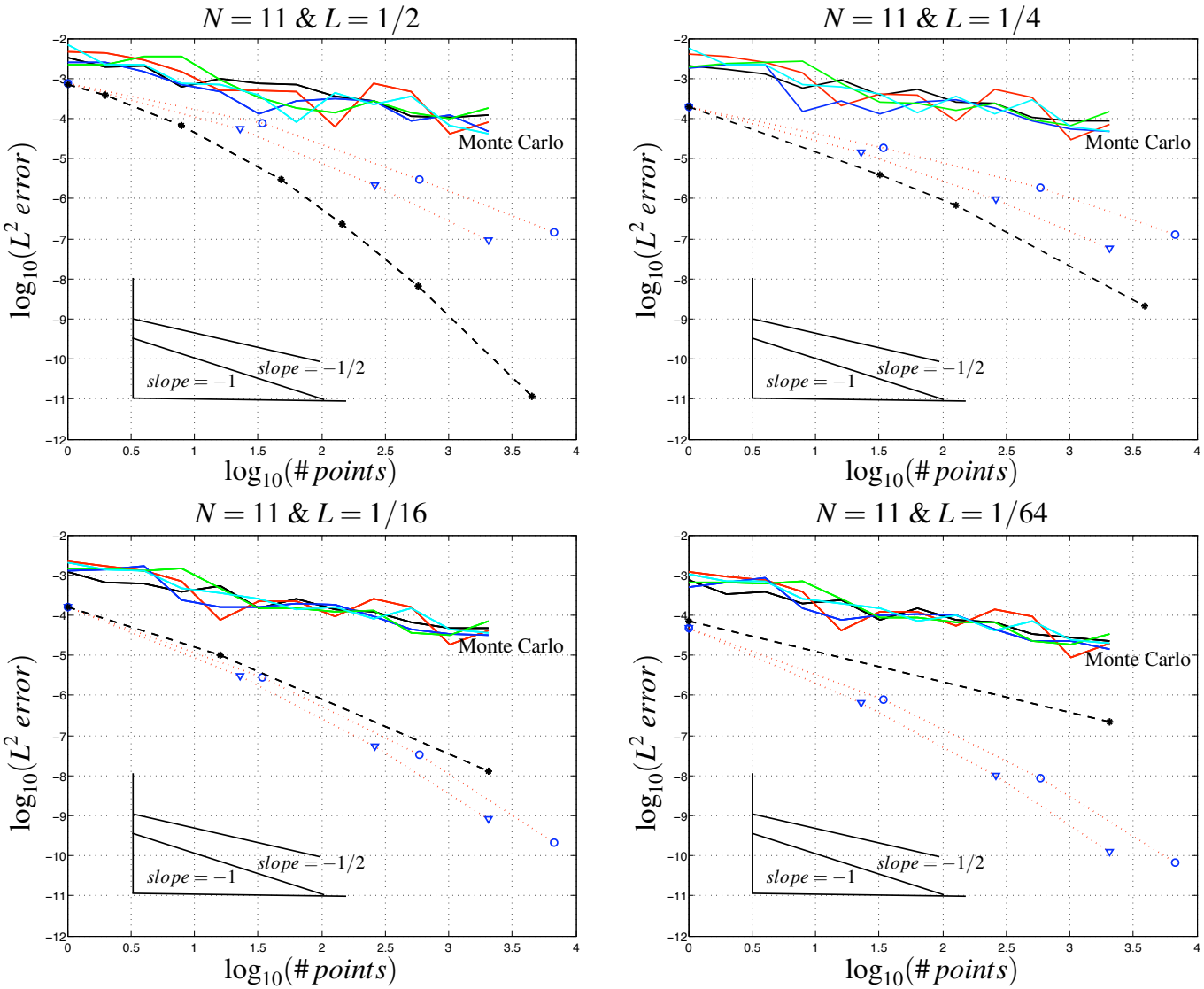








Stochastic Diffusion: Smolyak



Sparse Grids for Stochastic Integrals

1. Integrals Arising from Stochastic Problems
2. Quadrature Rules
3. Product Rules for Higher Dimensions
4. Smolyak Quadrature
5. Covering Pascal's Triangle
6. How Grids Combine
7. Sparse Grids in Action
8. Smoothness is Necessary
9. A Stochastic Diffusion Equation
10. **Software**
11. Conclusion

Software:

I have found a limited amount of software online for doing sparse grids.

- **SPINTERP** (in MATLAB, by Andreas Klimke) does interpolation based on data values on a sparse grid; ACM TOMS Algorithm 847.
- **SMOLPACK** (in C, by Knut Petras) estimates integrals using sparse grids.

Software: SPINTERP

Downloadable from NETLIB, or from
http://www.ians.uni-stuttgart.de/spinterp/

Constructs an interpolant function which is the sum of piecewise linear interpolants on each product grid that is part of the sparse grid. The interpolant is the sum of these interpolant functions.

The latest versions of the software can

- interpolate using higher order polynomials;
- evaluate the derivatives of the interpolant at any point;
- search for minimizers of the interpolant;
- use anisotropic grids, ignoring dimensions with little information.

Software: SPINTERP

The use of the program is very simple. Here we define a function $\mathbf{f}(\mathbf{x}, \mathbf{y})$, indicate the spatial dimension is 2, and ask for a sparse grid interpolant over a specific rectangle. By default, the program will use enough points so that the interpolation error is judged to be reasonably small.

```
f = @(x,y) sin(x) + cos(y);
sp = spvals ( f, 2, [0 pi; 0 pi] )  <-- Set interpolant

x = pi*rand(1,5);
y = pi*rand(1,5);
spxy = spinterp ( sp, x, y ) <-- Evaluate interpolant
```

Software:

I have written C++, F90 and MATLAB versions of libraries and programs for sparse grid integration using specific 1D rules:

- **sparse_grid_cc** Clenshaw Curtis;
- **sparse_grid_gl** Gauss-Legendre;
- **sparse_grid_hermite** Hermite;
- **sparse_grid_laguerre** Laguerre;
- **sparse_grid_mixed** Mixture of 1D rules;
- **sparse_grid_mixed_growth** Mixture + growth;
- **sgmga** mixture + growth + anisotropic;

For example, look at
http://people.sc.fsu.edu/~jburkardt/m_src/sparse_grid_cc/sparse_grid_cc.html

Software: Sparse Grid Mixed Weight

Here is essentially the verbatim MATLAB code for computing the weights of a sparse grid rule that uses a mixed set of 1D factors.

Many operations are handled by function calls.

The important thing is to try to see that Smolyak's formula for $\mathcal{A}(q, d)$ is being implemented here.

An additional concern is that we are trying to take advantage of nesting. Thus, we have an array **sparse_unique_index** that helps us with the bookkeeping to deal with duplicate points.

Software: Sparse Grid Mixed Weight

```
function sparse_weight = sparse_grid_mixed_weight ( dim_num, level_max, ...
  rule, alpha, beta, point_num )

  point_total_num = sparse_grid_mixed_size_total ( dim_num, level_max, rule );

  sparse_unique_index = sparse_grid_mixed_unique_index ( dim_num, level_max, ...
    rule, alpha, beta, point_total_num );

  sparse_weight(1:point_num) = 0.0;

  point_total = 0;

  level_min = max ( 0, level_max + 1 - dim_num );

  for level = level_min : level_max

    level_1d = [];
    more_grids = 0;
    h = 0;
    t = 0;
```

Software: Sparse Grid Mixed Weight

```
    while ( 1 )

      [ level_1d, more_grids, h, t ] = comp_next ( level, dim_num, level_1d, ...
        more_grids, h, t );

      order_1d = level_to_order ( dim_num, level_1d, rule );

      order_nd = prod ( order_1d(1:dim_num) );

      grid_weight = product_mixed_weight ( dim_num, order_1d, order_nd, ...
        rule, alpha, beta );

      coeff = r8_mop ( level_max - level ) ...
        * r8_choose ( dim_num - 1, level_max - level );

      for order = 1 : order_nd

        point_total = point_total + 1;

        point_unique = sparse_unique_index(point_total);

        sparse_weight(point_unique) = sparse_weight(point_unique) ...
          + coeff * grid_weight(order);

      end
      if ( ~more_grids )
        break
      end
    end
  end
  return
end
```

Software: Rule Generation

The simplest family of sparse quadrature rules is based on a single 1D rule, and each spatial dimension is treated the same.

The family of rules is indexed by **L**, the level, which starts at 0 with the 1 point rule.

The number of points **N** depends on **L**, the spatial dimension **D**, and the nesting of the underlying rule.

For a given 1D rule (say **laguerre**), the routines available are:

- **sparse_grid_laguerre_size** returns the number of points
- **sparse_grid_laguerre_index** shows which 1D rule each abscissa component comes from
- **sparse_grid_laguerre** returns the weights and abscissas

Software: Rule Generation

```
N = sparse_grid_laguerre_size ( D, L );

W = new double[N];
X = new double[D*N];

sparse_grid_laguerre ( D, L, N, W, X );

sum = 0;
for ( p = 0; p < N; p++ )
{
  sum = sum + W[p] * f ( X+p*D );
}
```

Software: File Format

A file format for quadrature rules means that software programs can communicate;

Results can be precomputed.

File data can easily be checked, corrected, emailed, or otherwise exploited.

The basic format uses 3 files:

- **R file**, 2 lines, D columns, the "corners" of the region
- **W file**, N lines, 1 column, the weight for each abscissa
- **X file**, N lines, D columns, the abscissas

Software: File Format

The "columns" are simply numbers separated by blanks.

A single file could have been used, but it would have internal structure.

To determine D and N, a program reads the X file and counts the number of "words" on a line, and the number of lines.

No particular ordering for the abscissas is assumed, but each line of the W and X files must correspond.

I have used this format for a 3x3 Clenshaw Curtis product rule and a sparse grid rule for integration in 100D!

For directories of these kinds of files, look at
http://people.sc.fsu.edu/~jburkardt/datasets/sparse_grid_cc
and so on.

Software: File Format

R file

```
---------
-1.0 -1.0
+1.0 +1.0
```

W file	X file	
0.111	-1.0	-1.0
0.444	-1.0	0.0
0.111	-1.0	+1.0
0.444	0.0	-1.0
1.777	0.0	0.0
0.444	0.0	+1.0
0.111	+1.0	-1.0
0.444	+1.0	0.0
0.111	+1.0	+1.0

Software: File Format

Another advantage of exporting quadrature rules to a file is that it is possible to precompute a desired family of rules and store them.

These files can be read in by a program written in another computer language; they can be mailed to a researcher who does not want to deal with the initial rule generation step.

Software: Precision Testing

Once we have quadrature rules stored in files, we can easily run degree of precision tests.

An executable program asks the user for the quadrature file names, and M, the maximum polynomial degree to check.

The program determines the spatial dimension D implicitly from the files, as well as N, the number of points.

It then generates every appropriate monomial, applies the quadrature rule, and reports the error.

Software: Precision Checking

```
23 October 2008   8:04:55.816 AM

NINT_EXACTNESS
  C++ version

  Investigate the polynomial exactness of a quadrature
  rule by integrating all monomials of a given degree
  over the [0,1] hypercube.

NINT_EXACTNESS: User input:
  Quadrature rule X file = "ccgl_d2_o006_x.txt".
  Quadrature rule W file = "ccgl_d2_o006_w.txt".
  Quadrature rule R file = "ccgl_d2_o006_r.txt".
  Maximum total degree to check =        4
  Spatial dimension =            2
  Number of points  =            6
```

Sotware: Precision Checking

Error	Degree	Exponents
0.0000000000000001	0	0 0
0.0000000000000002	1	1 0
0.0000000000000002	1	0 1
0.0000000000000002	2	2 0
0.0000000000000002	2	1 1
0.0000000000000002	2	0 2
0.0000000000000002	3	3 0
0.0000000000000002	3	2 1
0.0000000000000000	3	1 2
0.0000000000000001	3	0 3
0.041666666666665	4	4 0
0.0000000000000001	4	3 1
0.0000000000000000	4	2 2
0.0000000000000001	4	1 3
0.027777777777779	4	0 4

Sparse Grids for Stochastic Integrals

1. Integrals Arising from Stochastic Problems
2. Quadrature Rules
3. Product Rules for Higher Dimensions
4. Smolyak Quadrature
5. Covering Pascal's Triangle
6. How Grids Combine
7. Sparse Grids in Action
8. Smoothness is Necessary
9. A Stochastic Diffusion Equation
10. Software
11. **Conclusion**

CONCLUSION: A few observations

Sparse grids are based on combinations of product rules.

The combinations seek specific **precision** levels.

For integrands with bounded derivatives, precision produces **accuracy**.

By discarding some of the unneeded precision of product rules, sparse grids have a higher **efficiency**.

Abstract probability integrals, stochastic collocation and polynomial chaos expansions are examples of settings in which sparse grids may be useful.

CONCLUSION: A few observations

The underlying 1D quadrature rules could just as well be Jacobi, Laguerre, Hermite or their generalizations.

We can choose different quadrature rules for each dimension.

The rule family for a particular dimension could be a **piecewise** polynomial or some kind of **composite** rule. This makes it possible to handle functions with limited differentiability.

CONCLUSION: A few observations

The approach we have outline here is **isotropic**. It treats each spatial dimension with the same degree of importance.

But many very high dimensional problems that people work on are solvable in part because most of the dimensions have very little variation.

An **anisotropic** sparse grid rule can be developed which chooses the maximum order in each dimension based on weights.

These weights can be supplied in advance by the user, or determined adaptively.

CONCLUSION: References

Volker **Barthelmann**, Erich **Novak**, Klaus **Ritter**,
High Dimensional Polynomial Interpolation on Sparse Grids,
Advances in Computational Mathematics, 12(4) 2000, p273-288.

John **Burkardt**, Max **Gunzburger**, Clayton **Webster**,
Reduced Order Modeling of Some Nonlinear Stochastic PDE's,
IJNAM, 4(3-4) 2007, p368-391.

Thomas **Gerstner**, Michael **Griebel**,
Numerical Integration Using Sparse Grids,
Numerical Algorithms, 18(3-4), 1998, p209-232.

Andreas **Klimke**, Barbara **Wohlmuth**,
Algorithm 847: SPINTERP: Piecewise Multilinear Hierarchical Sparse Grid Interpolation in MATLAB,
ACM Transactions on Mathematical Software, 31,(4), 2005, p561-579.

CONCLUSION: References

Sergey **Smolyak**,
Quadrature and Interpolation Formulas for Tensor Products of Certain Classes of Functions,
Doklady Akademii Nauk SSSR, 4, 1963, p240-243.