Covering Pascal's Triangle on a Budget: Accuracy, Precision and Efficiency in Sparse Grids

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

.....

Computational Mathematics Seminar
Mathematics Department
Auburn University
06 February 2009

.....

https://people.sc.fsu.edu/~jburkardt/presentations/...
sparse_2009_auburn.pdf

Accuracy, Precision and Efficiency in Sparse Grids

INTRODUCTION - Modern Computations

Modern computations require high dimensional integration:

- Financial mathematics: 30D or 360D;
- Evolutionary biology (integration over evolutionary trees);
- Stochastic differential equations: 10D, 20D, 50D;
- Integrals derived from physics (Ising, quantum field theory).
- Atomic properties by integration over electron configurations weighted by probabilities

INTRODUCTION - High Dimensional Integration is Hard

These integrals are too complex to evaluate symbolically.

But even numerical approximation of high dimensional integrals can be **intractable**:

- The region may have very high dimension;
- The integrand may have endpoint singularities;
- The integrand may be very expensive to evaluate;
- There may be a need for *extreme* accuracy.

INTRODUCTION - Accuracy, Precision, Efficiency

In this talk, we consider the problem of constructing interpolatory quadrature rules for high dimensional regions.

For smooth integrands, rule **precision** implies **accuracy**.

But the natural method of creating precise rules, using products, incurs a cost that is exponential in the spatial dimension.

We show that this explosion is not a necessary feature of interpolation, and we investigate **efficient** methods of achieving precision, and hence accuracy, for smooth integrands in high dimensional spaces.

Accuracy, Precision and Efficiency in Sparse Grids

QUADRATURE: Approximation of Integrals

Integrals are numerically approximated by **quadrature rules**.

In 1D, this is a "mature" (dead?) area.

$$\int_{\Omega} f(x)\,dx \quad \approx \quad \sum_{i=1}^{N} w_i\, f(x_i)$$

- Interpolatory rules: Newton-Cotes, Chebyshev points;
- Semi-interpolatory rules: Gauss rules;
- Sampling rules: Monte Carlo and Quasi-Monte Carlo;
- Transform rules: tanh, tanh-sinh, erf rules.

QUADRATURE: Families of Rules

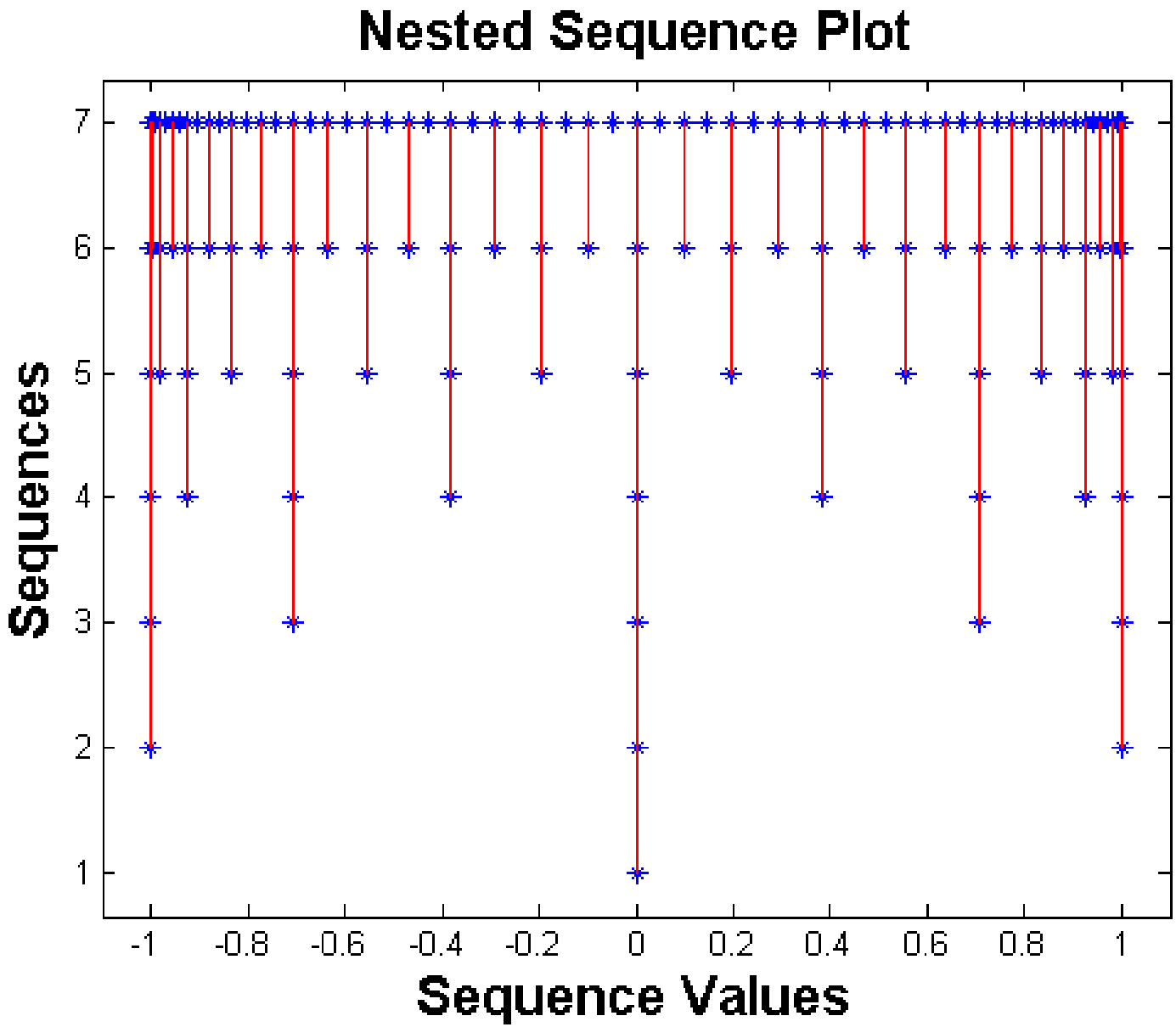
Most quadrature rules are available in any order **N**.

Generally, increasing **N** produces a more accurate result
(more about this in a minute!)

Under that assumption, a cautious calculation uses a sequence of increasing values of **N**.

An efficient calculation chooses the sequence of **N** in such a way that previous function values can be reused. This is called **nesting**.

1D Quadrature: Interpolatory



QUADRATURE: Precision

If a quadrature rule is exact when applied to any polynomials of degree **P** or less, the rule has **precision P**.

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

- Interpolatory rules: **P = N.** (assuming symmetry);
- Gauss rules **P = 2 * N - 1** ;
- Monte Carlo and Quasi-Monte Carlo rules, **P = 0** ;
- "transform rules": tanh, tanh-sinh, erf rules **P = 1** .

High precision is a property of interpolatory and Gauss rules.

QUADRATURE: Precision Can Mean Accuracy

Using a rule with **P = N** on a smooth function, low order terms get integrated, leaving an error that is $O(\frac{1}{N}^{N+1})$,

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

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

Keep in mind that precision:

- *is not necessary* - after all, Monte Carlo rules work.
- *is not a guarantee* - Newton Cotes rules are unstable;
- *can be harmful* - f(x) = step or piecewise or singular!

QUADRATURE: Accuracy is the Goal

Accuracy simply measures the error in the result.

A rule is accurate for a given class of integrands if we can show that the error (or expectation of the error) goes to zero as $h \to 0$ or $N \to \infty$.

A rule can be accurate without being precise.

The $\frac{1}{\sqrt{N}}$ accuracy of the Monte Carlo rule depends on the Law of Large Numbers.

The (exponential) accuracy of tanh, tanh-sinh and erf rules is essentially anecdotal.

QUADRATURE: Efficiency is the Number of Abscissas

Efficiency measures the amount of work expended for the result.

For quadrature, we measure our work in terms of the number of function evaluations, which in turn is the number of abscissas.

Since it is common to use a sequence of rules, it is important, for efficiency, to take advantage of nestedness, that is, to choose a family of rule for which the function values at one level can be reused on the next.

Accuracy, Precision and Efficiency in Sparse Grids

PRODUCT RULES: Formed from 1D Rules

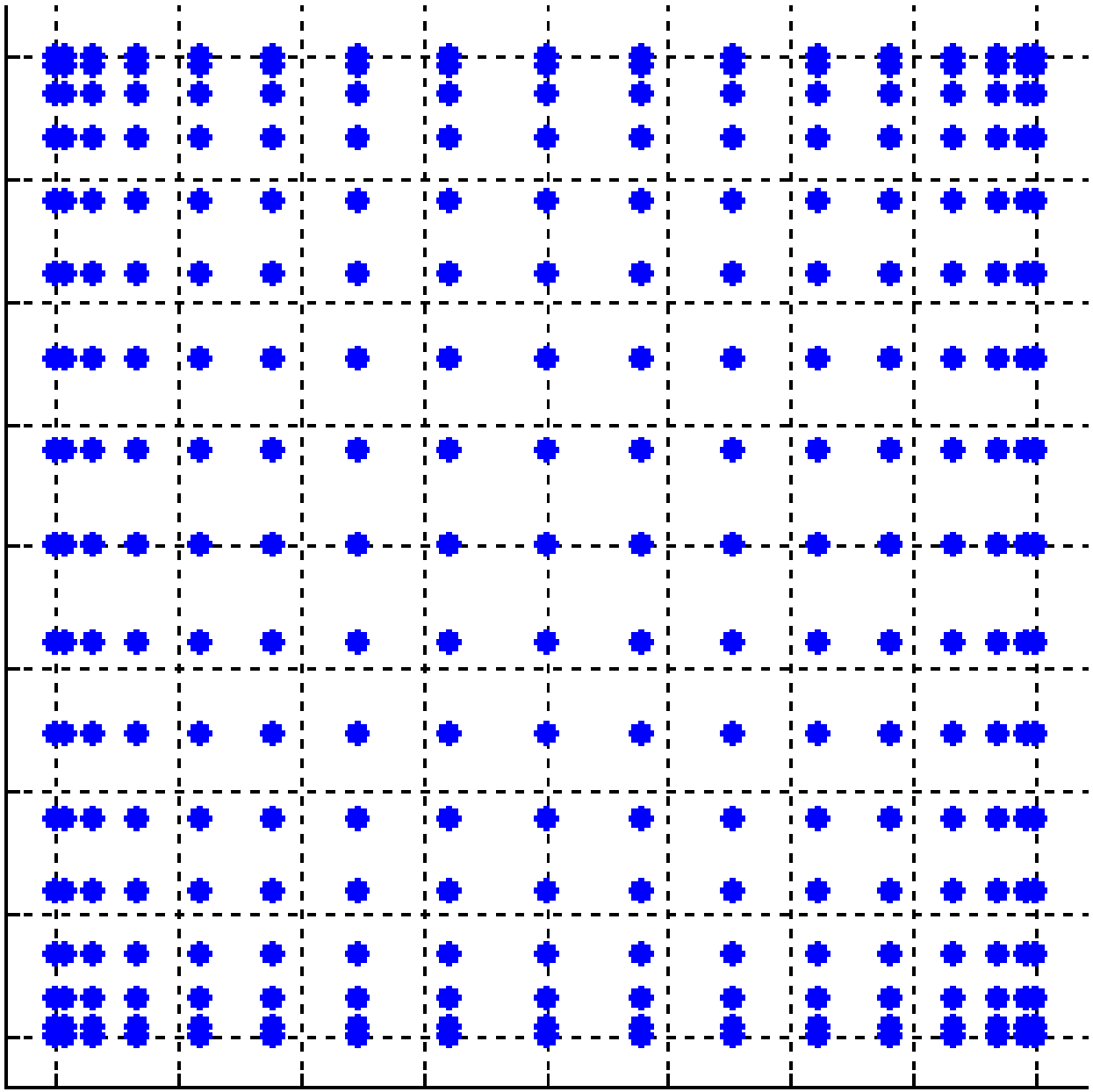
Let Q_i be the i-th member of a family of nested quadrature rules, with order N_i and precision P_i.

We can construct a corresponding family of 2D product rules as $Q_i \otimes Q_i$, with order N_i^2 and precision P_i.

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

Everything extends to the general M-dimensional case... except that the order growth N_i^M is unacceptable!

PRODUCT RULES: 17x17 Clenshaw-Curtis



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

Suppose we "square" a rule which is precise for monomials 1 through x^4.

We get a product rule 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

Precision for only part of a row of Pascal's triangle **is not useful!**. If we can't integrate x^5 or y^5 exactly, errors in those monomials determine our accuracy.

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

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 row 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?

PRODUCT RULES: What Do We Want?

Product rules can't take us where we want to go.

But where is that, exactly?

We want rules with faster convergence than Monte Carlo rules.

We are hoping to buy faster convergence using polynomial precision (smooth integrand **f(x)**)

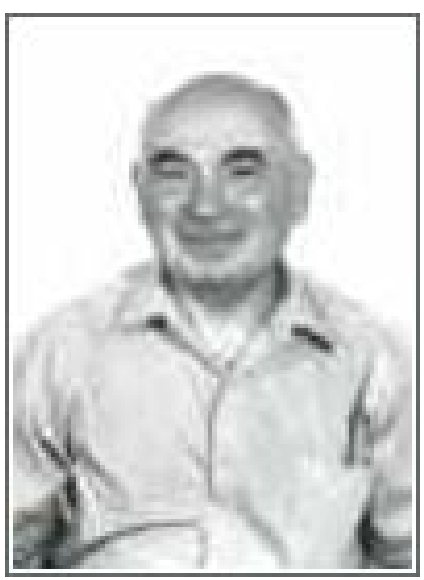
We'd like a family of rules, of increasing precision, so we can control our work and estimate our accuracy.

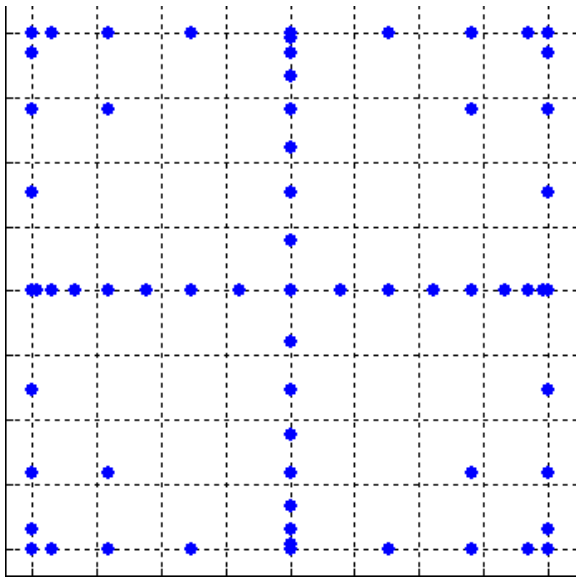
But we can't afford rules for which, at a fixed precision **P**,
the number of abscissas **N** grows exponentially with spatial dimension **M**.

Accuracy, Precision and Efficiency in Sparse Grids

SMOLYAK QUADRATURE

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





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

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

SMOLYAK QUADRATURE: Construction

We have an indexed family of 1D quadrature rules $\mathcal{Q}^i$.
We form rules for dimension **M**, indexed by "level" **q** starting at **M**.
Here $\mathrm{i} = i_1 + \cdots + i_M$.

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

Thus, the rule $\mathcal{A}(q, M)$ 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_i 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 number of points used in the rule $A(q, M)$.

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;

Then the error satisfies:

$$||I - A(q, M)|| = O(N^{-r} \cdot (\log N)^{(M-1)(r+1)})$$

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, Clenshaw-Curtis Smolyak 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**).

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.

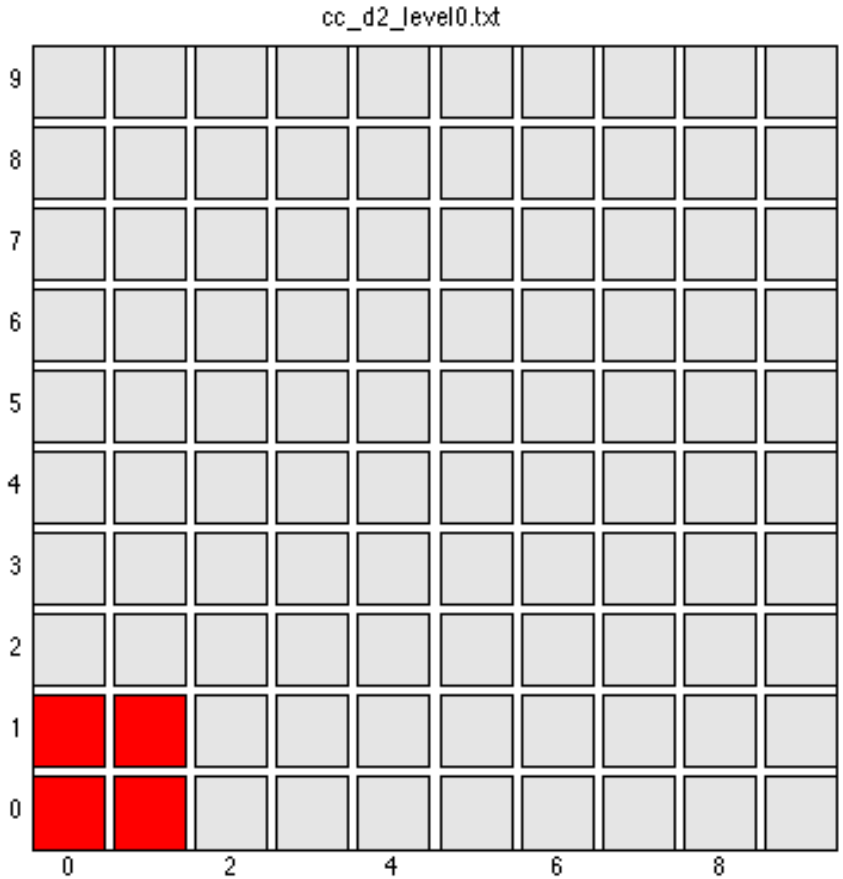
I've had to turn Pascal's triangle sideways, to an XY grid. If we count from 0, then box (I,J) represents $x^i y^j$.

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 is the precision of the rule.

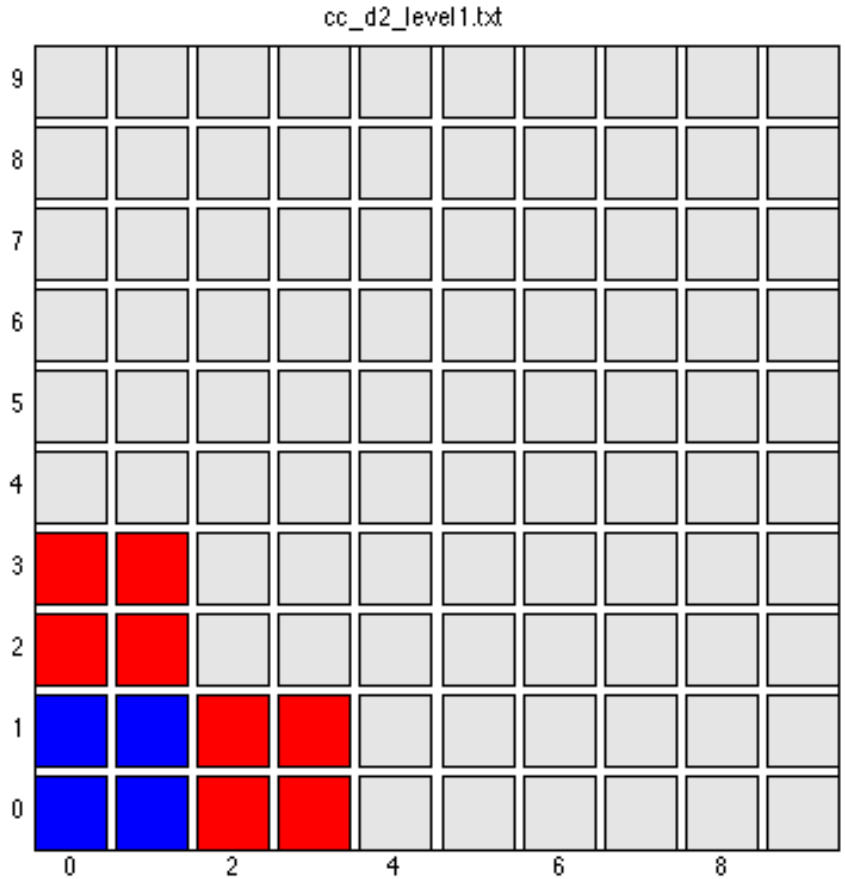
We will see levels 0 through 4 and expect precisions 1 through 11 by 2's.

COVERING PASCAL'S TRIANGLE: 2D Level 0



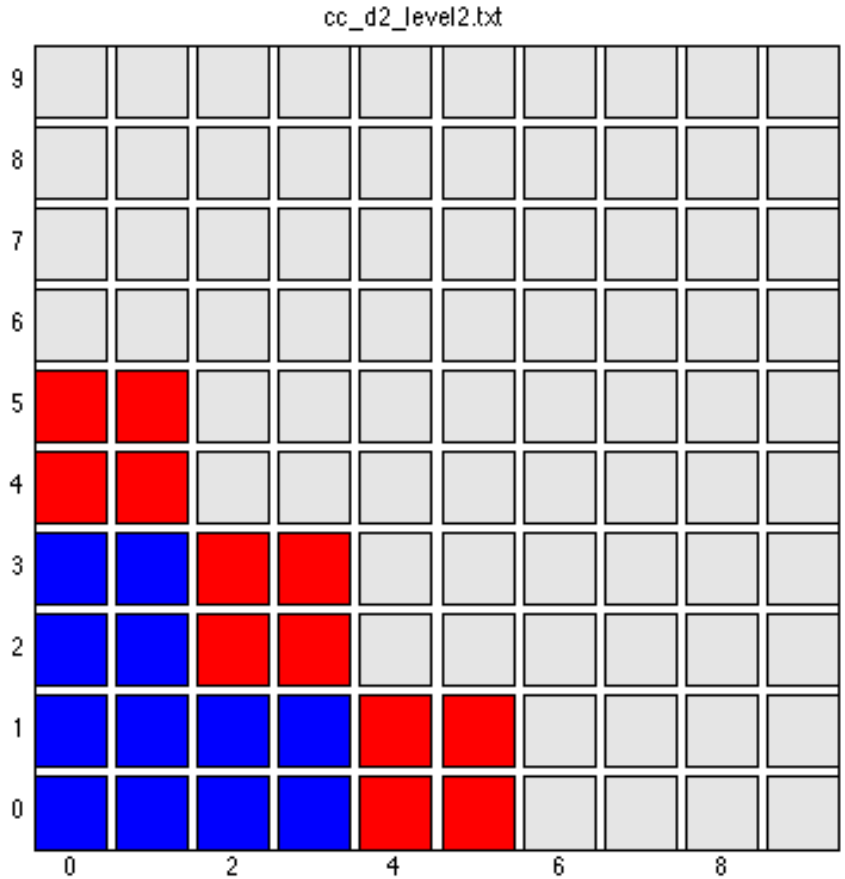
$Q_1 \otimes Q_1$

COVERING PASCAL'S TRIANGLE: 2D Level 1



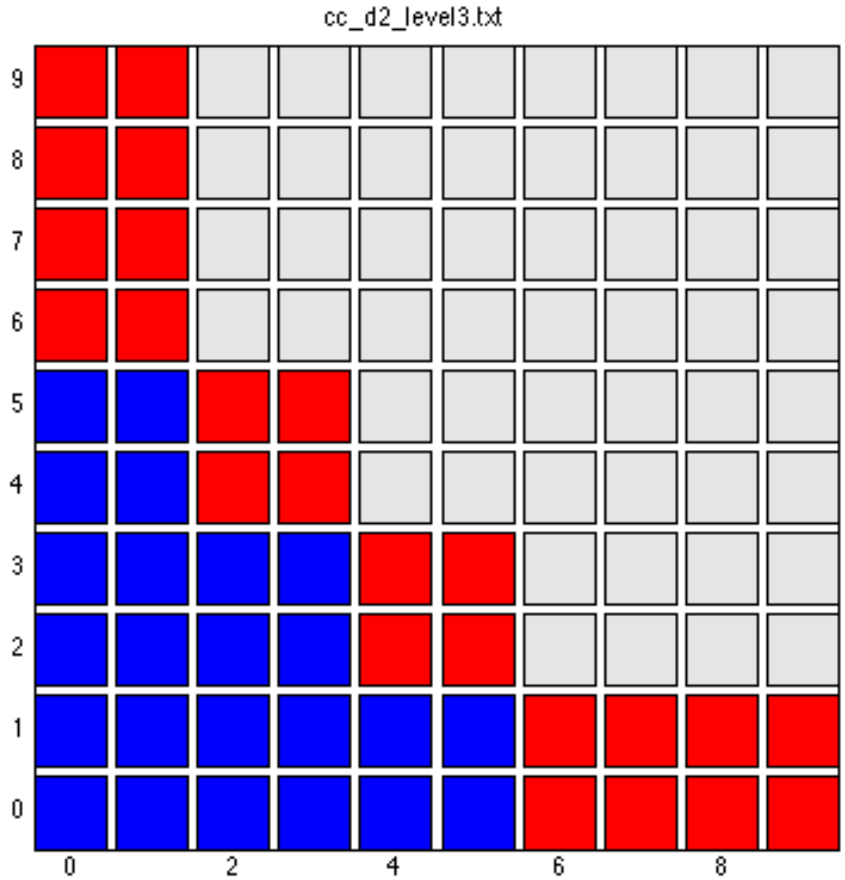
$Q_3 \otimes Q_1 + Q_1 \otimes Q_3$ - old

COVERING PASCAL'S TRIANGLE: 2D Level 2



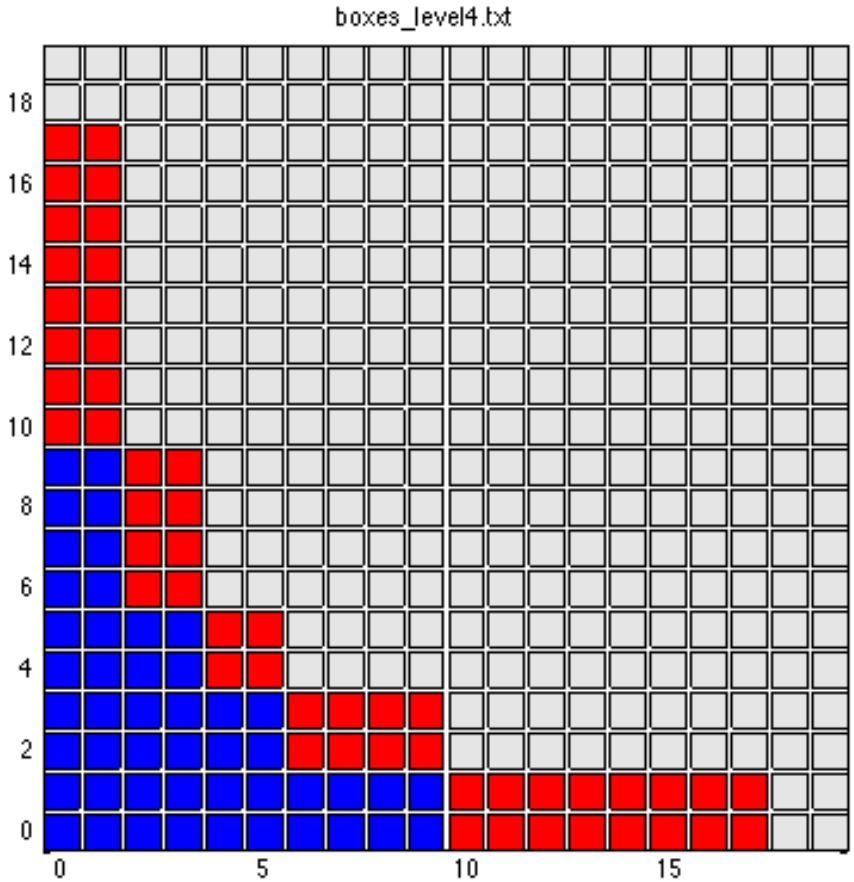
$Q_5 \otimes Q_1 + Q_3 \otimes Q_3 + Q_1 \otimes Q_5$ - old.

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;

Accuracy, Precision and Efficiency in Sparse Grids

HOW GRIDS COMBINE

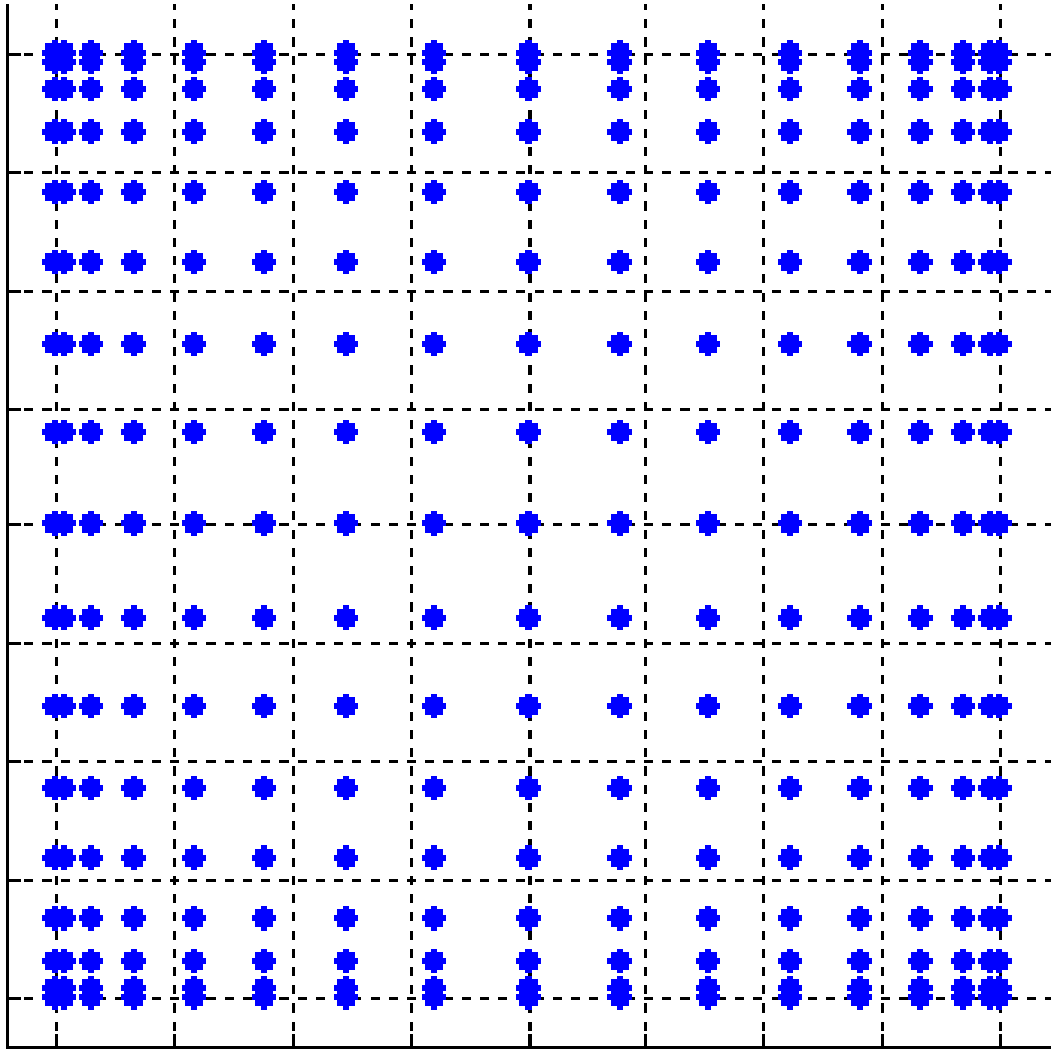
We said that the Smolyak construction combines low order product rules, and that the result can be regarded as a single rule.

Let's look at the construction of the Smolyak grid of level 4 in 2D.

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

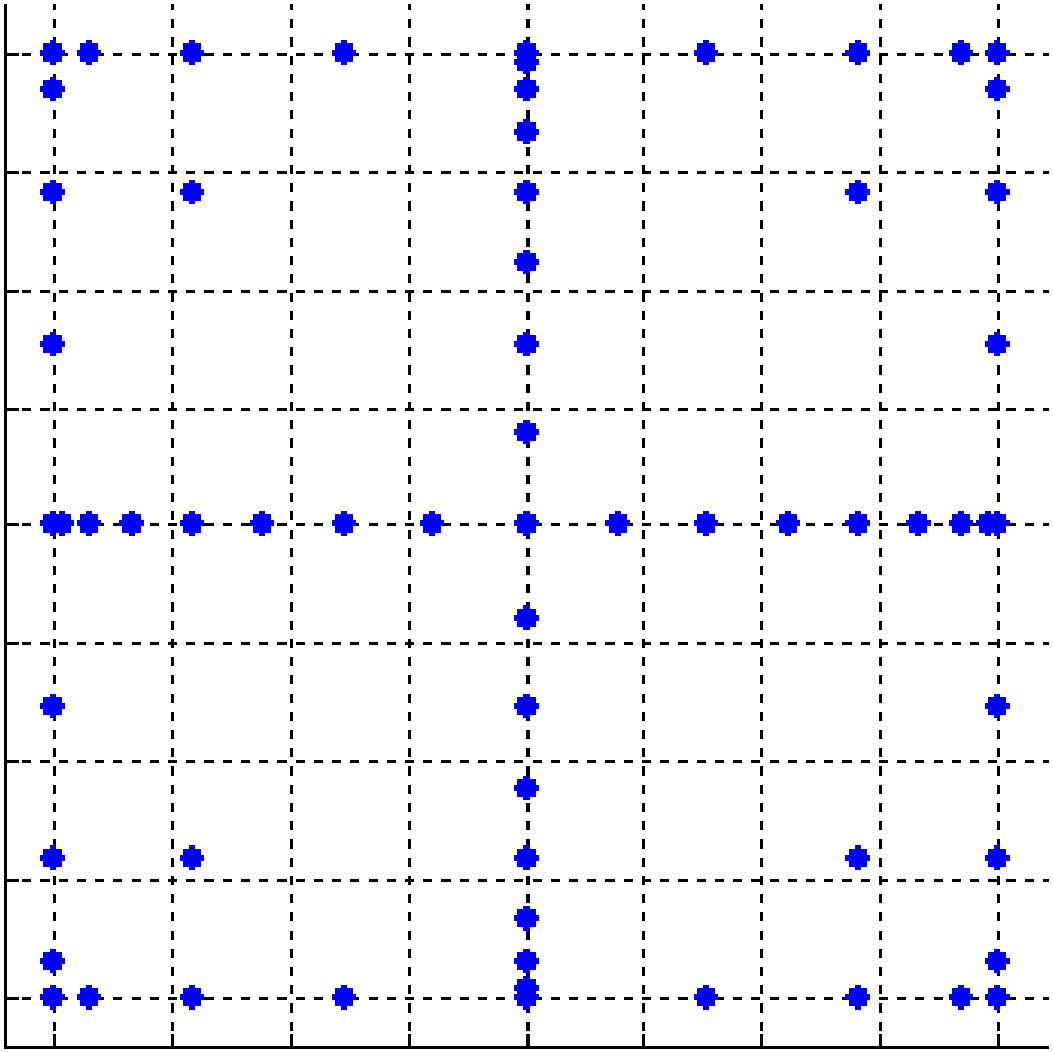
Because of nesting, every product rule we form will be a subset of the 17x17 full product grid.

HOW GRIDS COMBINE: 2D Order 17 Product Rule



A 17x17 product grid (289 points).

HOW GRIDS COMBINE: 2D Level4 Smolyak Grid



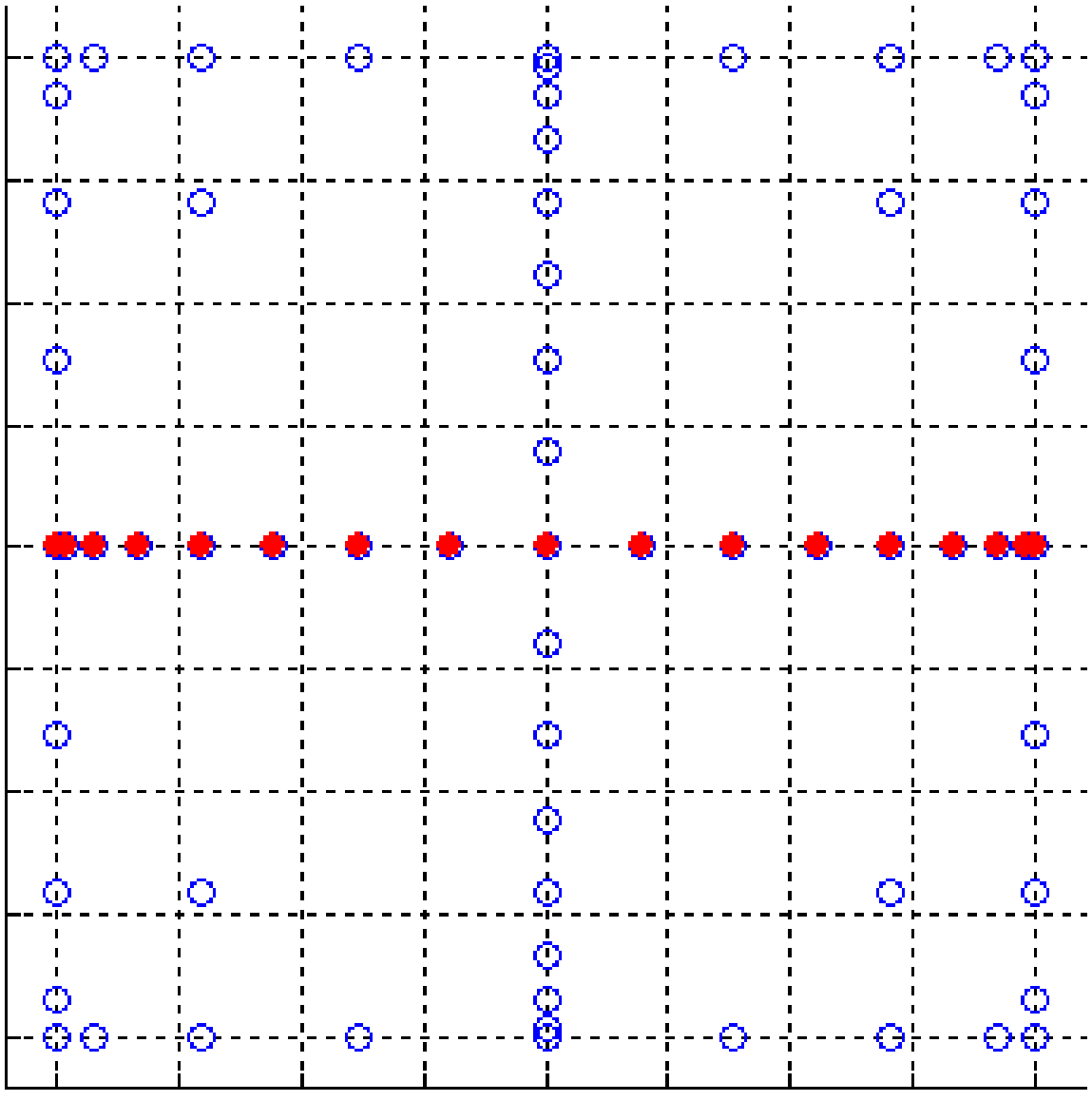
An "equivalent" sparse grid (65 points).

HOW GRIDS COMBINE

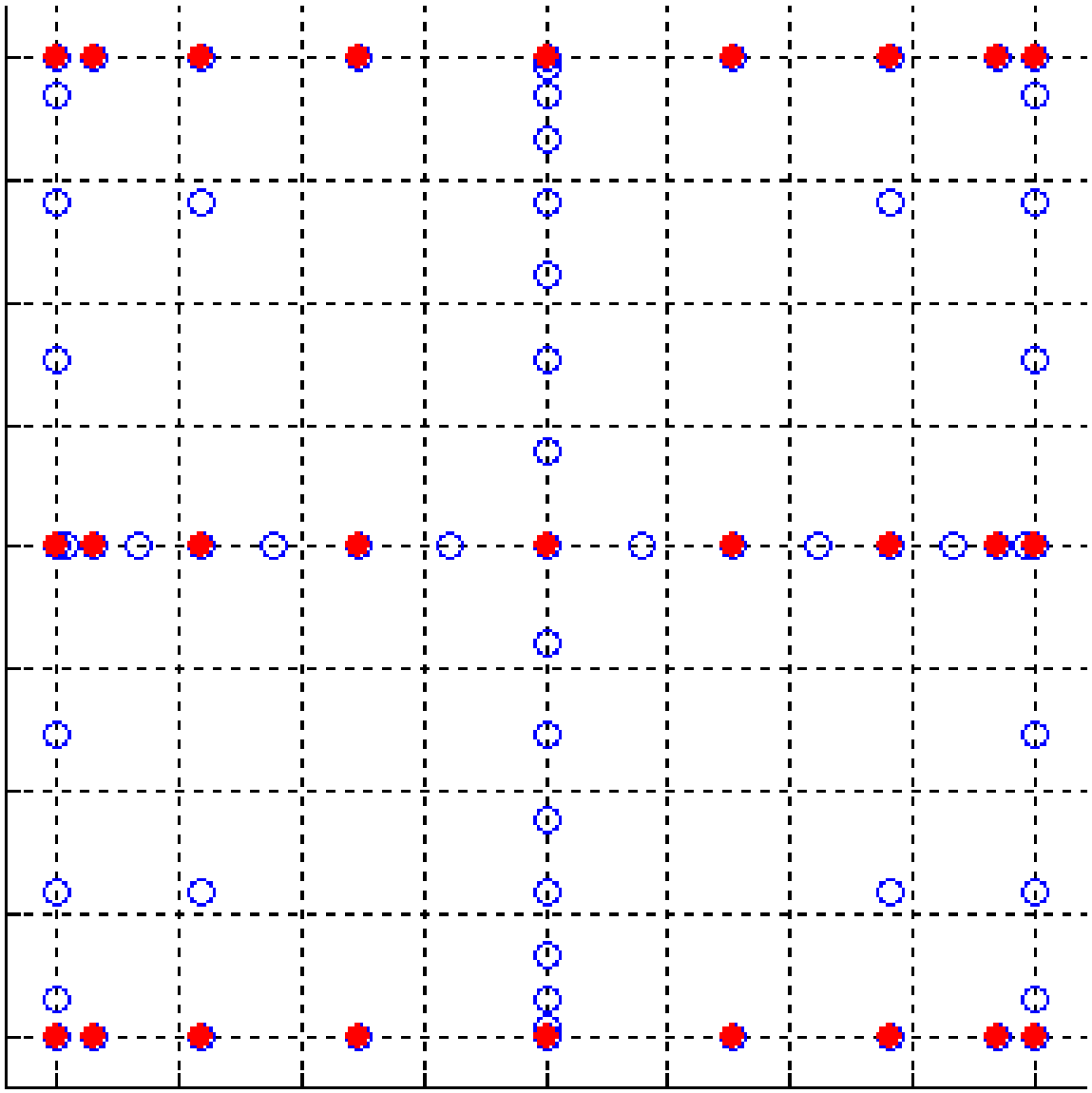
$$\mathcal{A}(6,2) = \sum_{6-2+1 \le |\mathbf{i}| \le 6} (-1)^{6-|\mathbf{i}|} \begin{pmatrix} 2-1 \\ 6-|\mathbf{i}| \end{pmatrix} (\mathcal{Q}^{i_1} \otimes \mathcal{Q}^{i_2})$$

$$\begin{aligned}
= &- \mathcal{Q}^1 \otimes \mathcal{Q}^4 \quad (Q_1 \otimes Q_9) \\
&- \mathcal{Q}^2 \otimes \mathcal{Q}^3 \quad (Q_3 \otimes Q_5) \\
&- \mathcal{Q}^3 \otimes \mathcal{Q}^2 \quad (Q_5 \otimes Q_3) \\
&- \mathcal{Q}^4 \otimes \mathcal{Q}^1 \quad (Q_9 \otimes Q_1) \\
&+ \mathcal{Q}^1 \otimes \mathcal{Q}^5 \quad (Q_1 \otimes Q_{17}) \\
&+ \mathcal{Q}^2 \otimes \mathcal{Q}^4 \quad (Q_3 \otimes Q_9) \\
&+ \mathcal{Q}^3 \otimes \mathcal{Q}^3 \quad (Q_5 \otimes Q_5) \\
&+ \mathcal{Q}^4 \otimes \mathcal{Q}^2 \quad (Q_9 \otimes Q_3) \\
&+ \mathcal{Q}^5 \otimes \mathcal{Q}^1 \quad (Q_{17} \otimes Q_1)
\end{aligned}$$

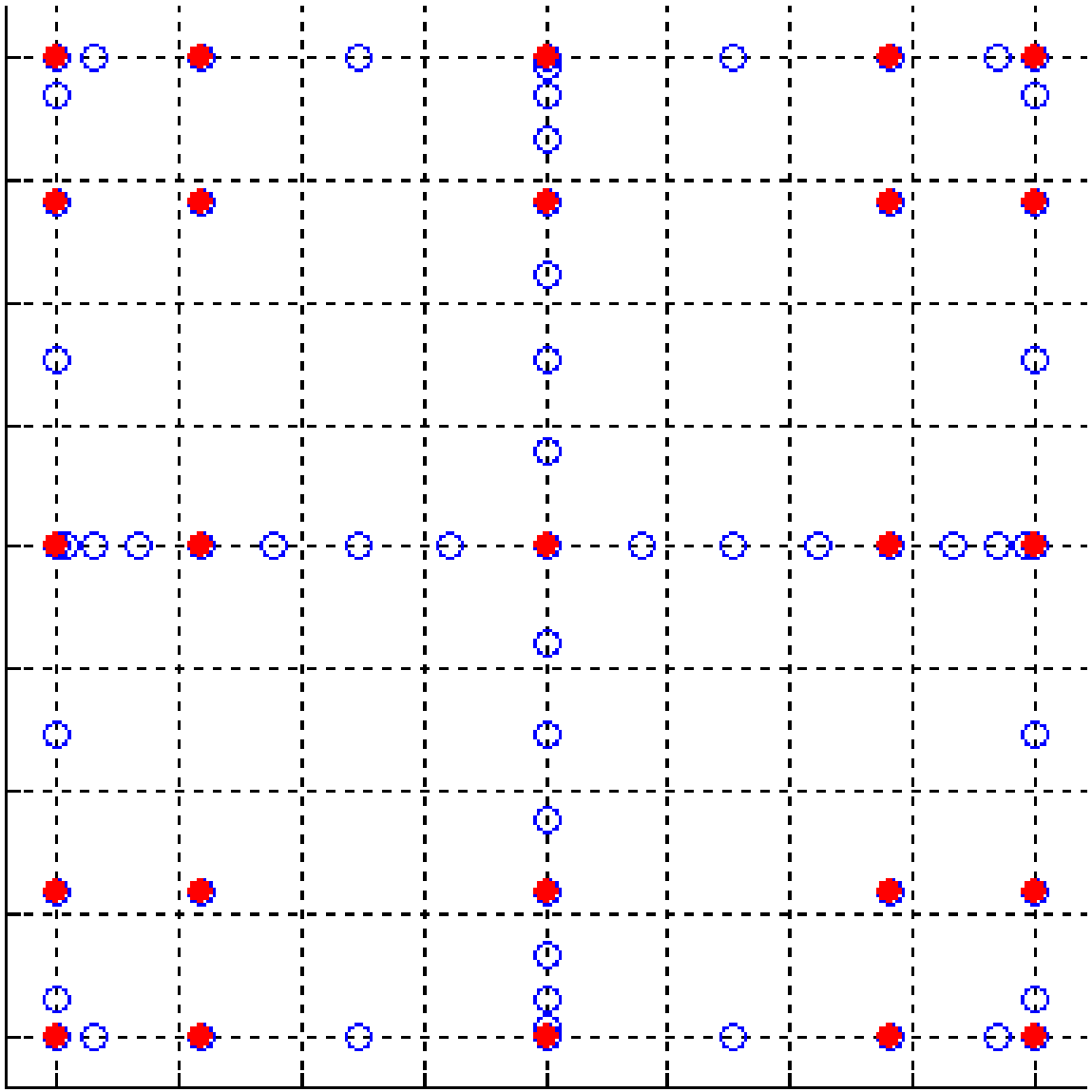
HOW GRIDS COMBINE: 2D Level4 17x1 component



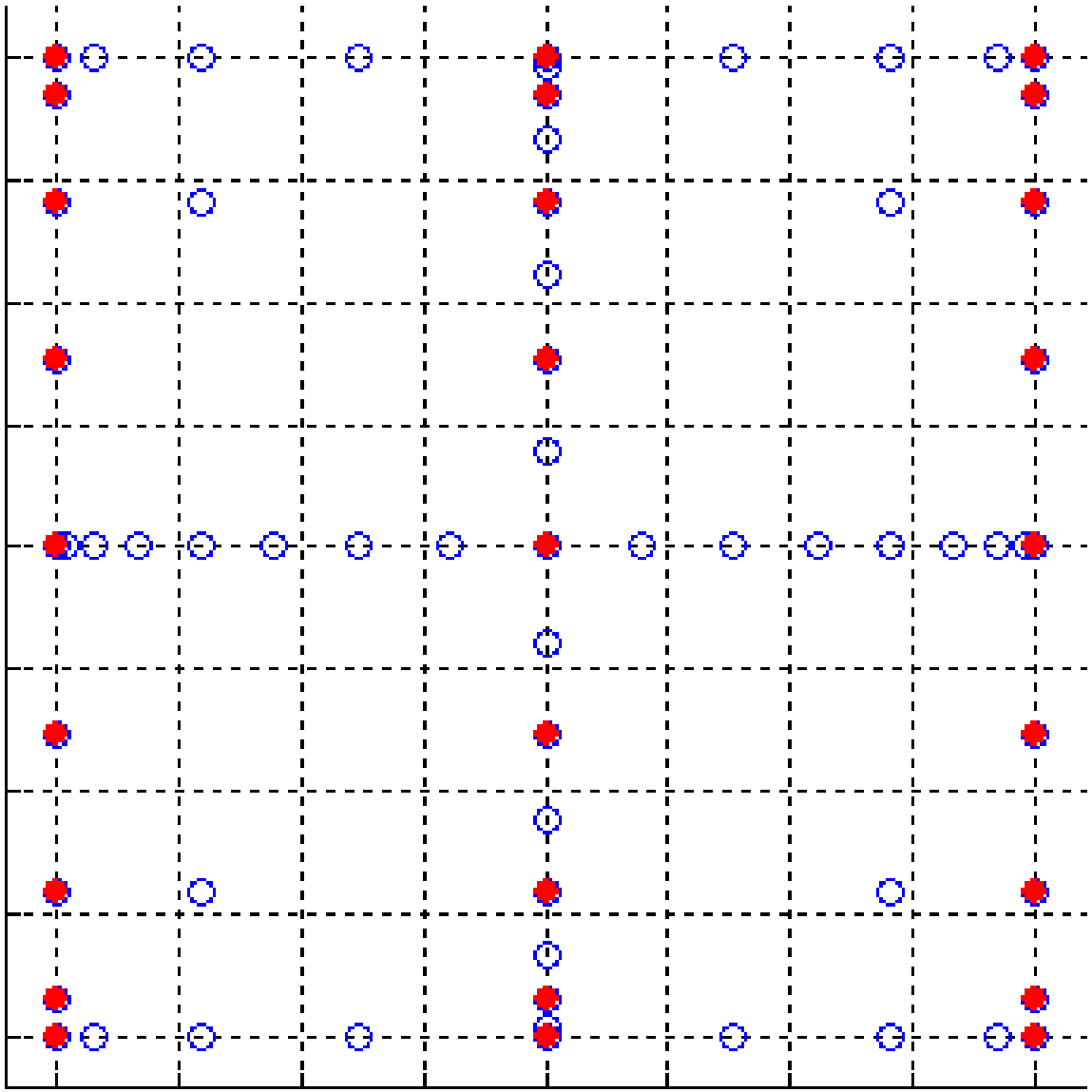
HOW GRIDS COMBINE: 2D Level4 9x3 component



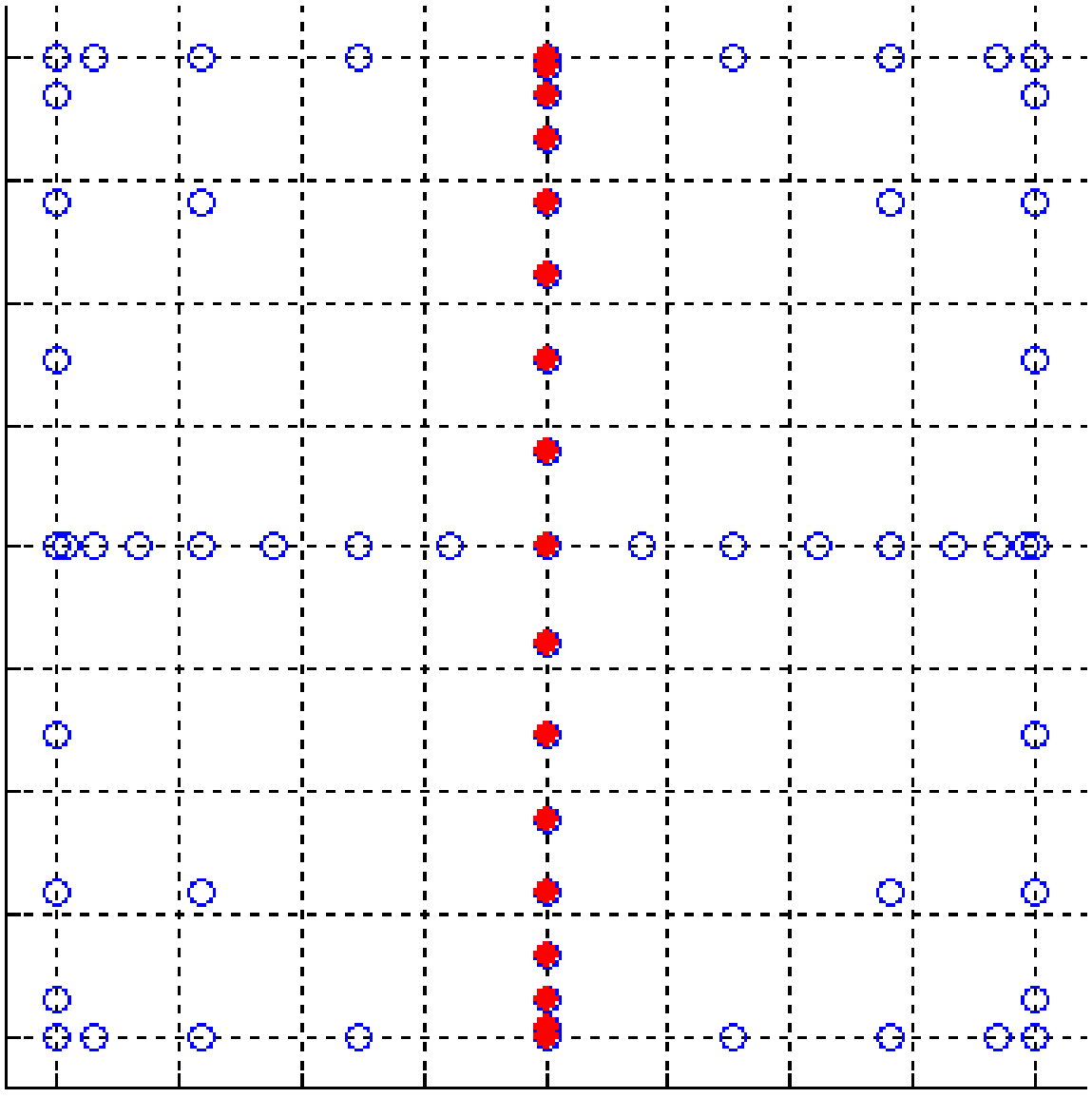
HOW GRIDS COMBINE: 2D Level4 5x5 component



HOW GRIDS COMBINE: 2D Level4 3x9 component



HOW GRIDS COMBINE: 2D Level4 1x17 component



HOW GRIDS COMBINE

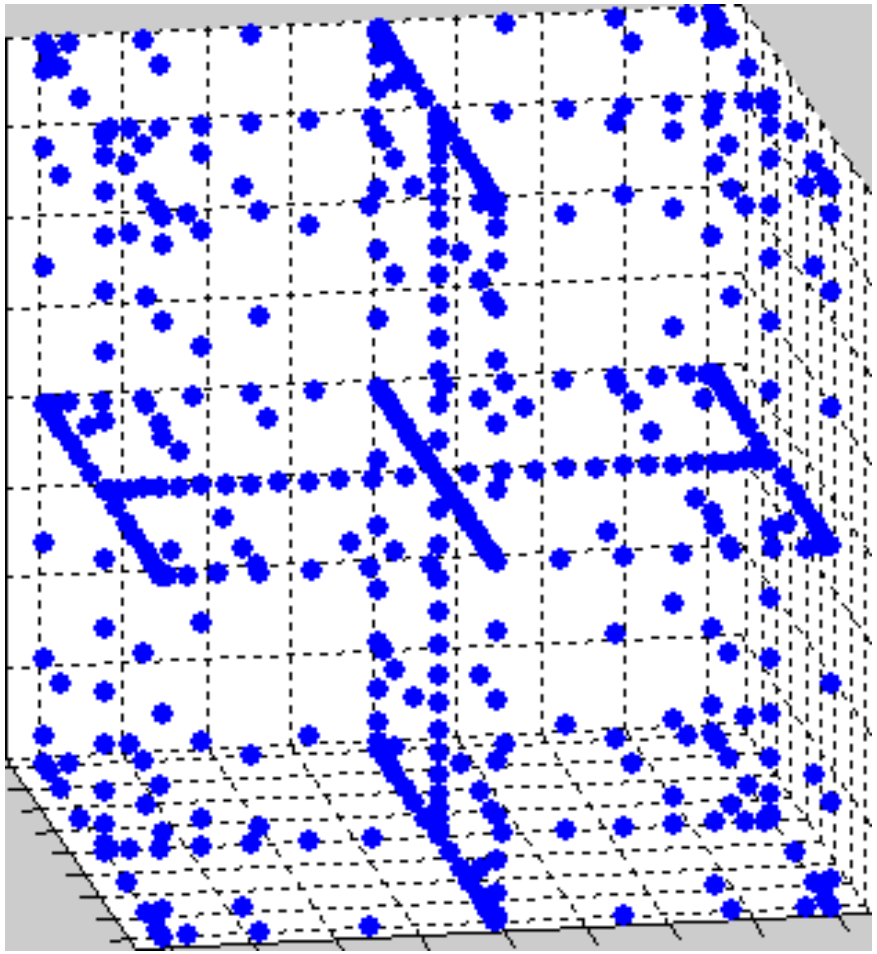
It's easy to get confused, and think that our Smolyak grid might have the same precision **P=17** as the 17x17 product rule.

But the level 4 rule has precision **P = 2*4+1 = 9**.

So in terms of efficiency, we should count the 65 points in the Smolyak grid against the 81 points in a product rule of precision 9.

This is not a huge savings; however, the advantage improves with increasing precision **P** or spatial dimension **M**.

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.

Accuracy, Precision and Efficiency in Sparse Grids

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**:

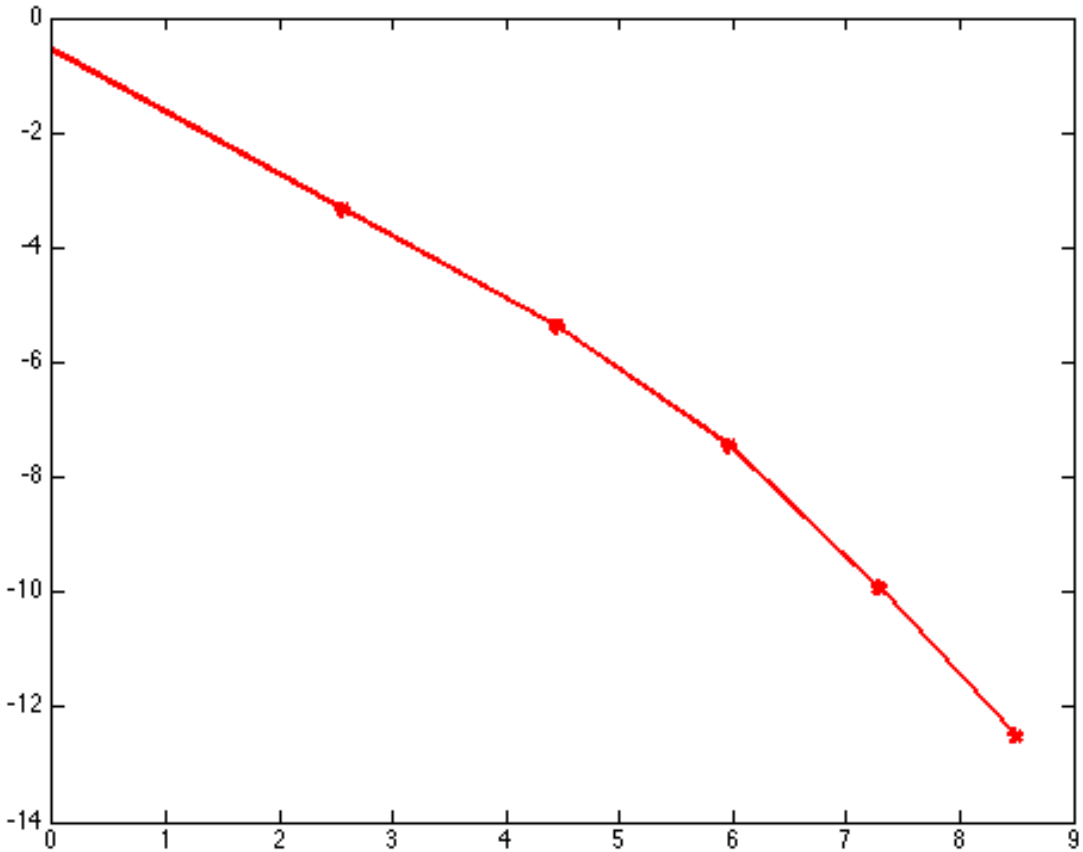
$$f(X) = \frac{1}{\prod_{i=1}^{M}(C_i^2 + (X_i - Z_i)^2)}$$

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



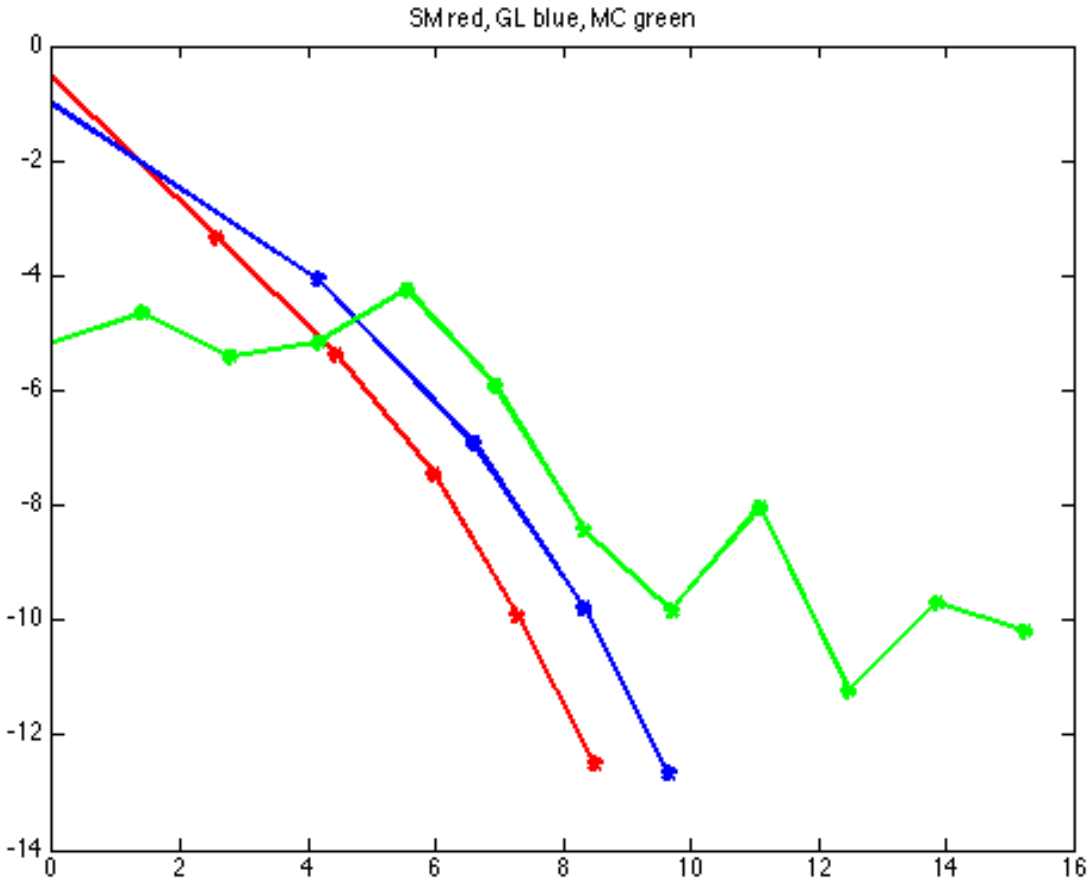
SPARSE GRIDS IN ACTION: 6D Gauss-Legendre

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

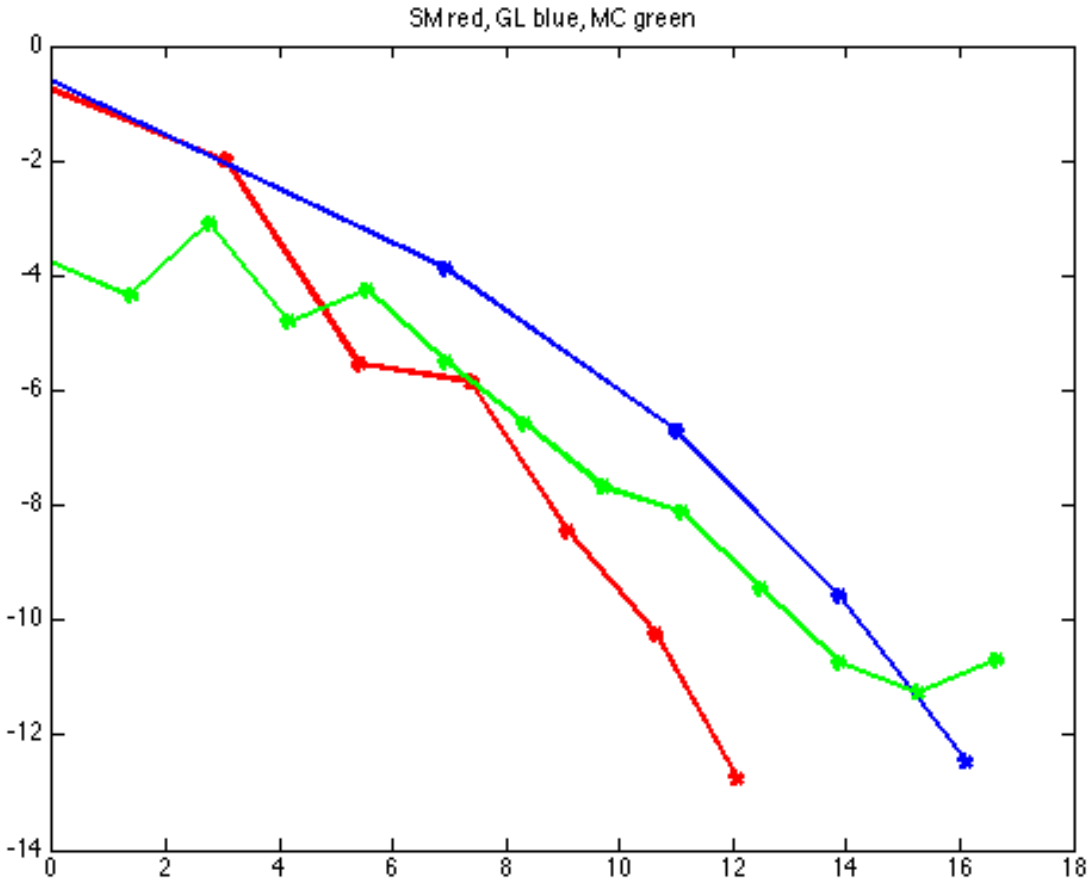
SPARSE GRIDS IN ACTION: 6D Monte Carlo

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.

Accuracy, Precision and Efficiency in Sparse Grids

A Few Words of Wisdom

"When good results are obtained in integrating a high-dimensional function, we should conclude first of all that an especially tractable integrand was tried and not that a generally successful method has been found.

"A secondary conclusion is that we might have made a very good choice in selecting an integration method to exploit whatever features of f made it tractable."

Art Owen, Stanford University.

A Few Words of Wisdom

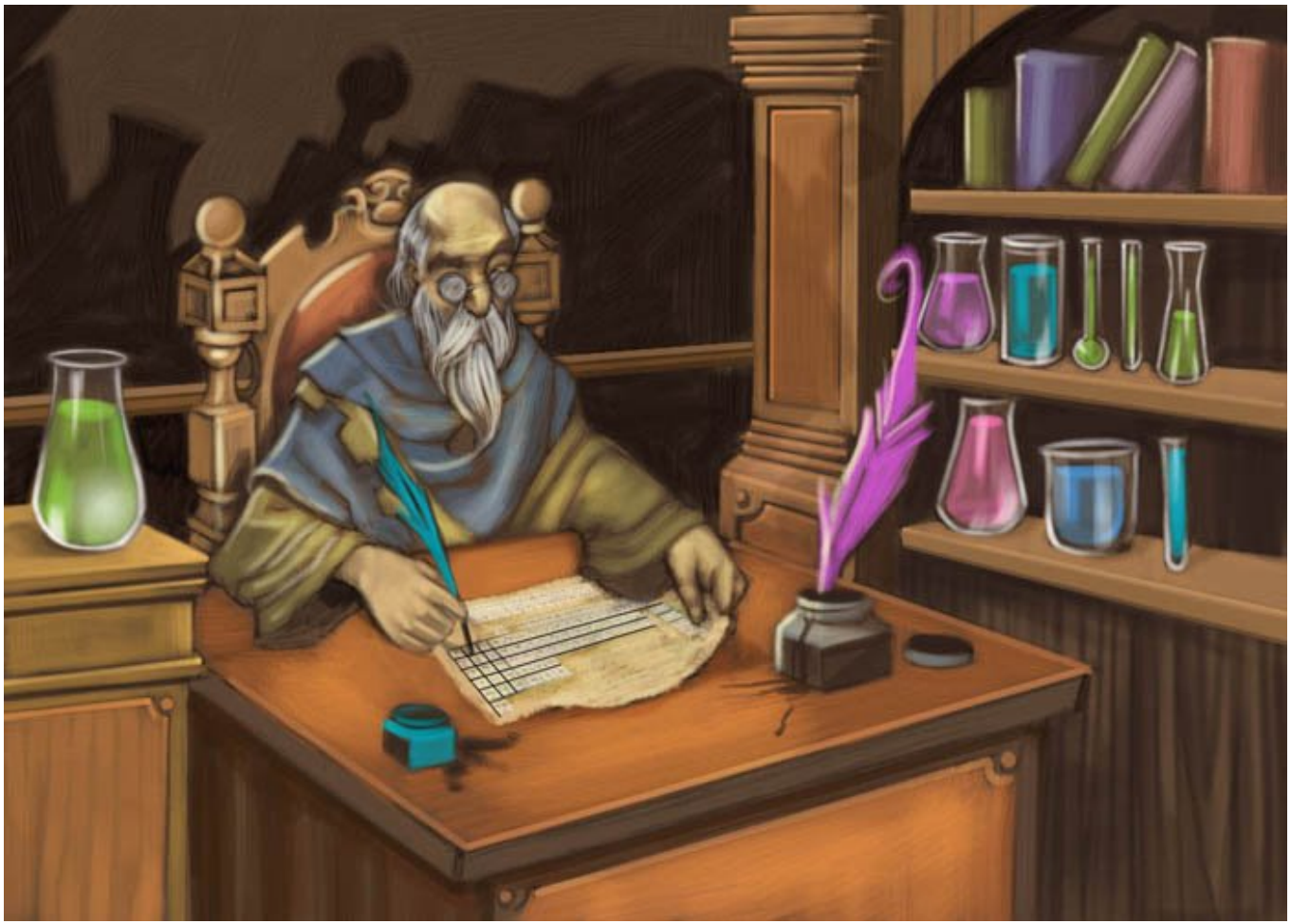
Art Owen's words apply here. A sparse grid approach is the right choice when the function to be integrated is known to be polynomial, or to have bounded derivatives up to the order of the rule we are applying.

In those cases, a sparse grid can extract extra information from the function values, to provide an answer that is exact for polynomials, and highly accurate for other smooth functions.

In order to ruin everything, however, we can simply suppose that f(x) is a step function!

SOFTWARE PRODUCTS

Smolyak's definition of sparse grids is almost magical; but it can take the novice a while to master the tricks. So it's important to bottle some of that magic in accessible tools!



SOFTWARE PRODUCTS: 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 PRODUCTS: 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 PRODUCTS: 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 PRODUCTS: 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!

SOFTWARE PRODUCTS: 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 PRODUCTS: 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 PRODUCTS: 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 PRODUCTS: 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
```

SOFTWARE PRODUCTS: 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

Accuracy, Precision and Efficiency in Sparse Grids

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: References

Volker **Barthelmann**, Erich **Novak**, Klaus **Ritter**, *High Dimensional Polynomial Interpolation on Sparse Grids*, Advances in Computational Mathematics, Volume 12, Number 4, March 2000, pages 273-288.

John **Burkardt**, Max **Gunzburger**, Clayton **Webster**, *Reduced Order Modeling of Some Nonlinear Stochastic Partial Differential Equations,* International Journal of Numerical Analysis and Modeling, Volume 4, Number 3-4, 2007, pages 368-391.

Thomas **Gerstner**, Michael **Griebel**, *Numerical Integration Using Sparse Grids*, Numerical Algorithms, Volume 18, Number 3-4, January 1998, pages 209-232.

Sergey **Smolyak**, *Quadrature and Interpolation Formulas for Tensor Products of Certain Classes of Functions*, Doklady Akademii Nauk SSSR, Volume 4, 1963, pages 240-243.