SPARSE GRIDS: Turning High Dimensional Data into Information

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

.....

Computational Mathematics Seminar
Mathematics Department
University of Pittsburgh

.....

11 November 2008

.....

https://people.sc.fsu.edu/~jburkardt/presentations/...
sparse_2008_pitt.pdf

SPARSE GRIDS: High Dimensional Data => Information

INTRODUCTION: Physical Laws and Regions

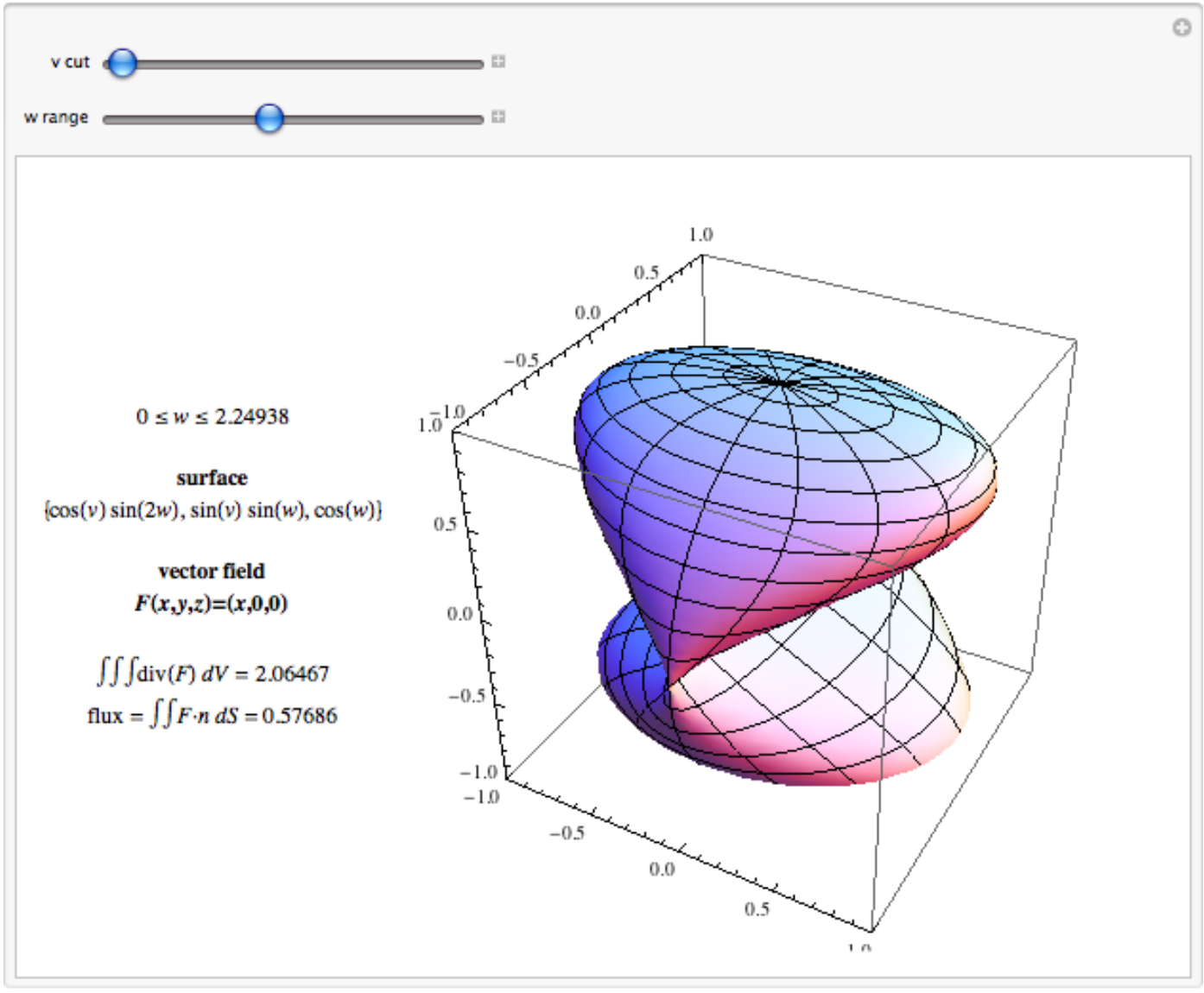
Computational science begins with the simulation of physical laws.

These laws often involve an integral operator applied to a function $f(x)$ over some integration region Ω.

Physical spaces are typically 2D or 3D, so Ω can make our life difficult because of its

- **shape** (polyhedral or curved boundaries)
- **embedding** (on the surface of a sphere, say).
- but not **high dimensionality!**

INTRODUCTION: Physical Laws and Regions



INTRODUCTION: Nonphysical Laws and Regions

Mathematical metaphor creates nonphysical spaces.

A point p in a 10-dimensional space may represent the values of 10 physical parameters that affect an output quantity $g(p)$.

If these values p vary over a space Ω, perhaps with a weighting function $w(\omega)$, we extend the idea of the integral to this **nonphysical integration region**, expressing the expected value as:

$$\overline{g(p)} = \frac{\int_\Omega \quad g(p(\omega)) \quad w(\omega) \quad d\omega}{\int_\Omega w(\omega)\, d\omega}$$

INTRODUCTION: Nonphysical Laws and Regions

Mathematical metaphor treats physical and nonphysical problems the same way.

The metaphor ignores some characteristic features of nonphysical integrals. Computationally, these features cannot be ignored!

- **smoothness** of $f(x)$ is more likely;
- **geometry** of the integration region is simpler;
- but the **dimensionality** is free to explode!

This is especially true in **probabilistic** and **stochastic** settings.

INTRODUCTION: The High Dimensional Challenge

The mere fact of high dimensionality can guarantee that approximate integration in a given region is difficult or impossible.

We will discuss how computational approaches to multidimensional quadrature either break down, or are unable to produce accurate results, when the dimension becomes too high.

We will show that, for a particular kind of integrand and integration region, **sparse grids** can reach far into high dimensional space and extract the information we want.

We will discuss some software that is publicly available and (we hope) both useful and usable.

SPARSE GRIDS: High Dimensional Data => Information

HIGH DIMENSIONAL PROBLEMS

Computational scientists are setting up high dimensional problems (and claim to be solving them, as well!).

- Financial mathematics: 30D or 360D
- ANOVA decompositions: 10D or 20D
- Evolutionary biology (integration over evolutionary trees)
- Signal processing
- Queue simulation (expected average wait)
- Stochastic differential equations: 10D, 20D, 50D
- Particle transport (repeated emission/absorption)
- Light transport (scattering)
- Path integrals over a Wiener measure (Brownian motion)
- Quantum properties (Feynman path integral)

HIGH DIMENSIONAL PROBLEMS: Biology

Peter Beerli's MIGRATE program:

Suppose we have observed sample data D, representing distinct but related populations, such as frogs on neighboring islands.

We assume the population dynamics depend on parameters P such as initial population sizes and migration rates,

The likelihood of a given set of values P can be computed as

$$L(P) = Prob(D|P) = \int_{\mathcal{G}} Prob(D|G) \quad Prob(G|P) \quad dG$$

Here $\mathcal{G}$ represents all possible genealogies!

HIGH DIMENSIONAL PROBLEMS: Physics

Atomic properties:

In order to calculate a physical property P of an atom with 10 electrons, we must evaluate that property for every configuration of the electrons, multiplied by the probability of that configuration. Each electron has 3 spatial coordinates, so we are working in a space of dimension 30.

This is a relatively small example, of course! There are elements with as many as 100 electrons; more electrons (and other degrees of freedom) must be modeled if we look at simple molecules.

HIGH DIMENSIONAL PROBLEMS: Finance

Asian Option Pricing:

We estimate the price of an Asian option, with strike price K, volatility σ, risk free interest rate r and expiration date T.

A standard approach takes d equally spaced time intervals Δt and arrives at an integral over the **d**-dimensional space C^d

$$Price = \int_{C^d} e^{-rT} f(x, r, \sigma, \Delta t) \quad dx$$

Money is at stake here, so accuracy is important,
the value of d can easily be 30 or higher.

HIGH DIMENSIONAL PROBLEMS: Uncertainty

Stochastic Collocation Techniques for "Black Box" models:

UNCERTAIN INPUT X => SYSTEM => OUTPUT Y

- the Yucca Mountain nuclear waste facility
- the Airbus 380 wing
- spacecraft reentering atmosphere
- groundwater flow and pollutant transmission

We have a model for *how* the inputs become outputs, but we don't actually understand the complex system.

We need plausible statistical estimates for the outputs, that is, the **expected values** and **variances**.

A single evaluation of $y = f(x)$ can be very costly!

HIGH DIMENSIONAL PROBLEMS: Uncertainty

Each input parameter x_i may have its own probability distribution: **uniform**, **normal**, **Jacobi**, **Hermite**, **Laguerre** or **generalized Laguerre** distributions.

The integration region will have a dimension as great as the number of interacting input parameters.

$$\overline{y} = \int_{\Theta} f(x) \quad x(\theta) \quad d\theta$$

If an input parameter is not a scalar but an uncertain **field**, we must add even more dimensions to the integral!

SPARSE GRIDS: High Dimensional Data => Information

INTEGRATION RULES

When approximating multidimensional integrals, there are several common techniques:

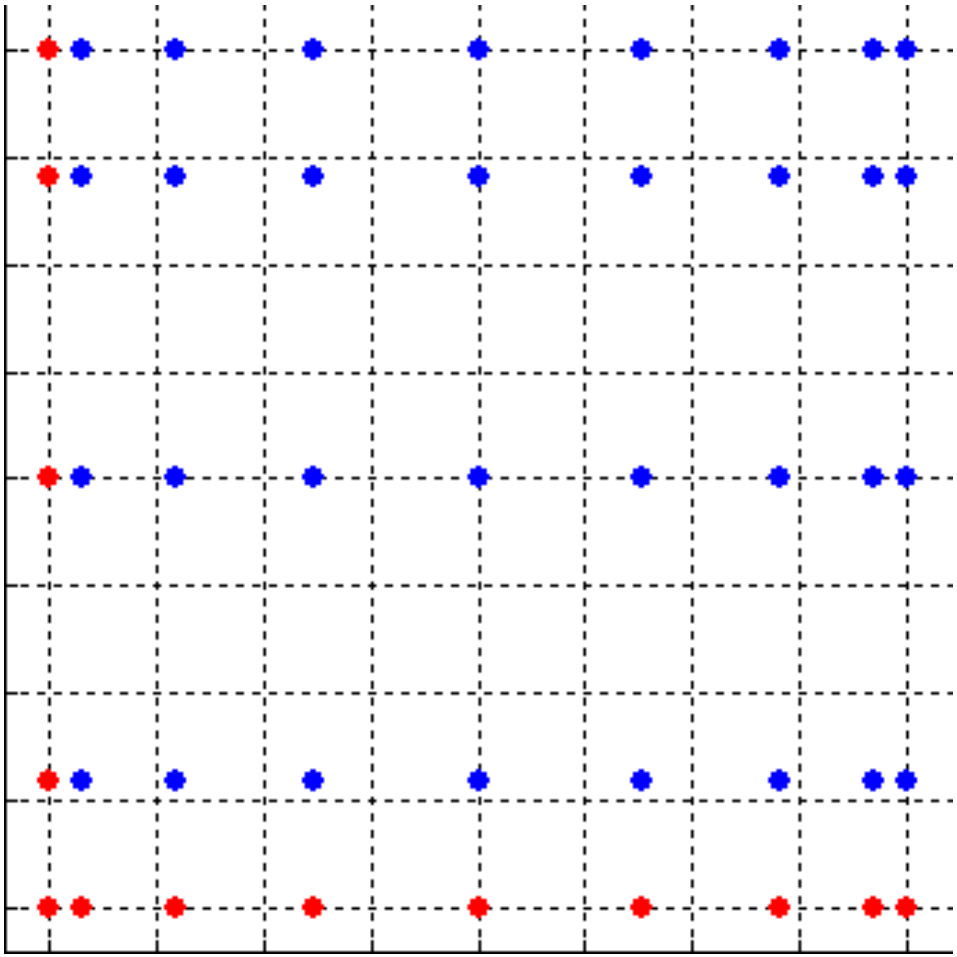
- **interpolatory product**, integrating an interpolant;
- **Gaussian product**, maximal 1D polynomial exactness;
- **minimal**, minimal number of points for ND polynomial exactness;
- **Monte Carlo**, uncorrelated sample points;
- **quasi Monte Carlo**, using a space filling sequence.

INTEGRATION RULES: Issues

Common issues for a multidimensional quadrature rule:

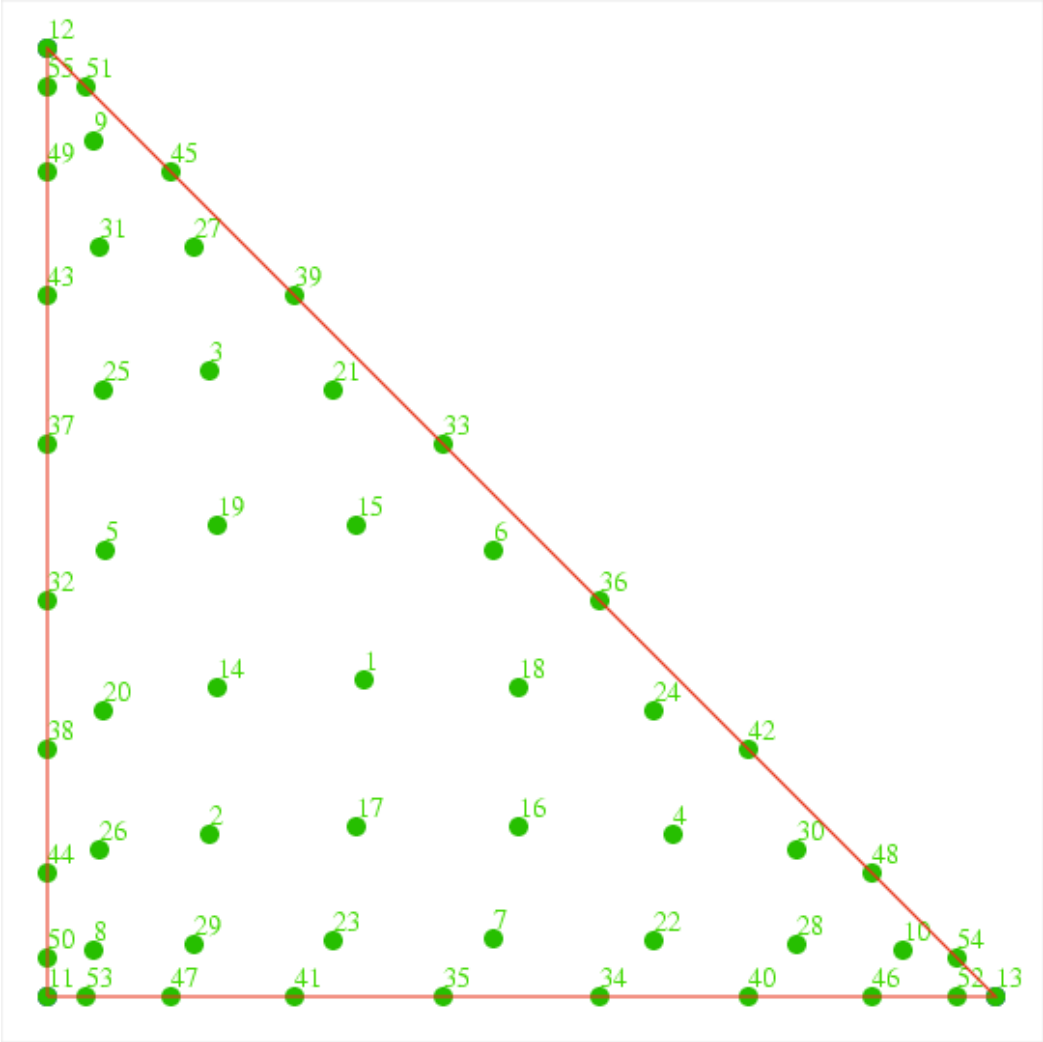
- **Automation**: can we easily generate variations of this rule, of any order, dimension, region, weight function?
- **Accuracy**: is the rule exact for some polynomial space? can we estimate the error as we go?
- **Robustness**: can it handle "misbehaving" $f(x)$?
- **Efficiency**: how many points required versus the theoretical minimum? Does the number explode with dimension?;
- **Reuse**; if an estimate is unsatisfactory, and we saved the function values, can we reuse them?

INTEGRATION RULES: Product Rules



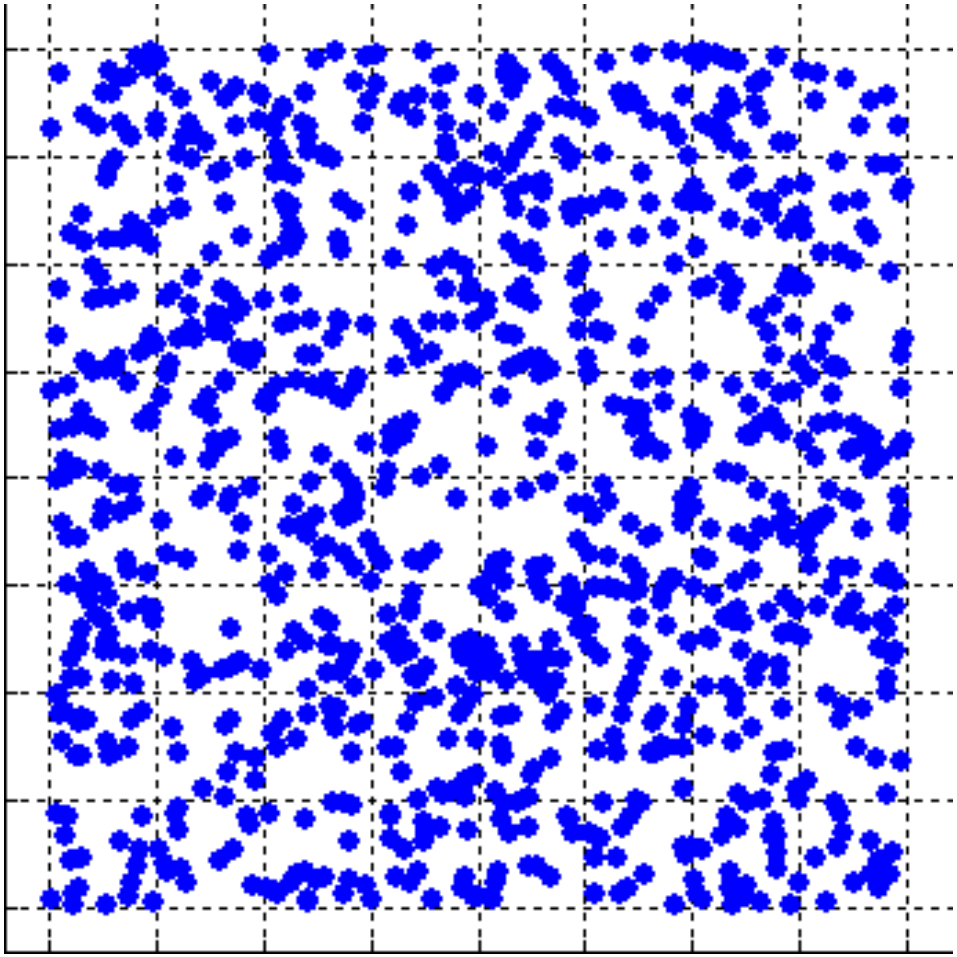
A product of 9 point and 5 point Clenshaw Curtis rules.

INTEGRATION RULES: Minimal Rules

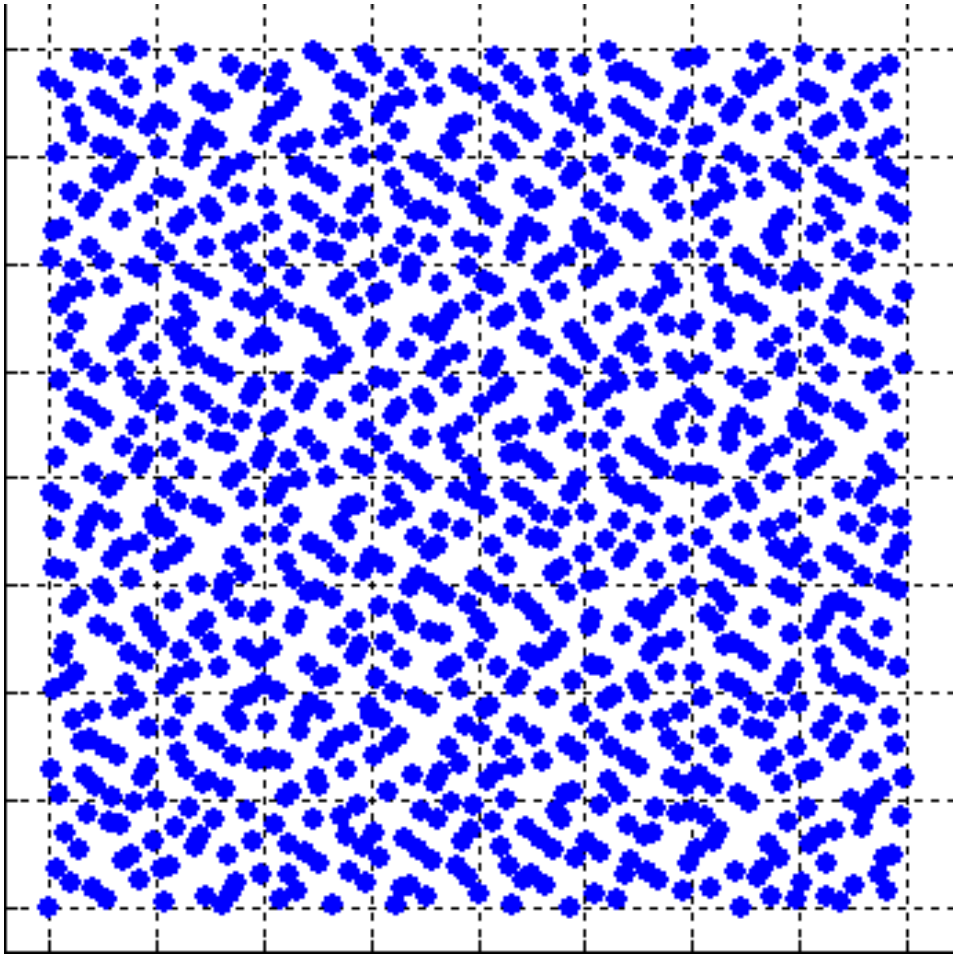


55 point Fekete rule, exact to degree 9 (55 x, y monomials).

INTEGRATION RULES: Monte Carlo

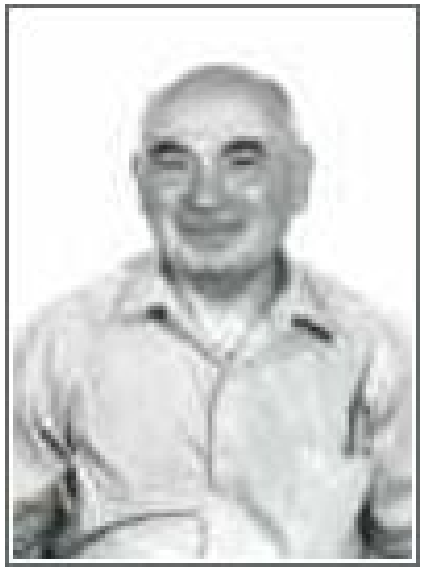


INTEGRATION RULES: Quasi Monte Carlo



SPARSE GRIDS: High Dimensional Data => Information

SMOLYAK QUADRATURE



Sergey Smolyak (1963) suggested sparse grids, an algebraic combination of low order rules:

- had the same polynomial accuracy as a product grid.
- looks like a product grid with points missing;
- formed by collecting lower order product grids.
- used *far fewer* points than a product grid.

SMOLYAK QUADRATURE: Construction

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

$$\mathcal{A}(q,d) = \sum_{q-d+1 \le |\mathrm{i}| \le q} (-1)^{q-|\mathrm{i}|} \left(\begin{array}{c} d-1 \\ q-|\mathrm{i}| \end{array} \right) (\mathcal{U}^{i_1} \otimes \cdots \otimes \mathcal{U}^{i_d})$$

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

SMOLYAK QUADRATURE

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

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: Exactness

A common choice is 1D Clenshaw-Curtis rules U^i, based on interpolation at the extrema of the Chebyshev polynomials.

The rules are nested, with U^i using $m_i = 2^{i-1} + 1$ points.
U^1 is the exception, using a single point.

Theorem

The Clenshaw-Curtis Smolyak formula $A(q, d)$ is exact for all polynomials of degree $2 * (q - d) + 1$ or less.

(Lowest rule has $q = d$, hence exactness 1.
Second rule has exactness 3, and so on.)

SMOLYAK QUADRATURE: Asymptotic Error

Let n be the number of points used in the rule $A(q, d)$,
let I_d be the integral of $f(x)$,
$f(x) : [-1, 1]^d \rightarrow R | D^\alpha$ continuous if $\alpha_i \leq r$ for all i;

Then the error satisfies:

$$||I_d - A(q, d)|| = O(n^{-r} \cdot (\log n)^{(d-1)(r+1)})$$

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

SMOLYAK QUADRATURE: Optimality

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

For large d, Clenshaw-Curtis Smolyak uses about $\frac{(2d)^k}{k!}$ 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^k that is independent of d).

SMOLYAK QUADRATURE

Number of points in a Clenshaw-Curtis Smolyak grid, for dimensions 1 to 5, and 10, and 1D point counts 1, 3, 5, ...65.

Dim 1D Rule	1	2	3	4	5	10
1	1	1	1	1	1	1
3	3	5	7	9	11	21
5	5	13	25	41	61	221
9	9	29	69	137	241	1581
17	17	65	177	401	801	8801
33	33	145	441	1105	2433	41265
65	65	321	1073	2929	6993	171425

SMOLYAK QUADRATURE

1D point count = 1D degree of precision.

Level	1D count	10D count	10D Accuracy
0	1	1	1
1	3	21	3
2	5	221	5
3	9	1581	7
4	17	8801	9
5	33	41265	11
6	65	171425	13

Multidimensional precision = 2 * LEVEL + 1.

SMOLYAK QUADRATURE

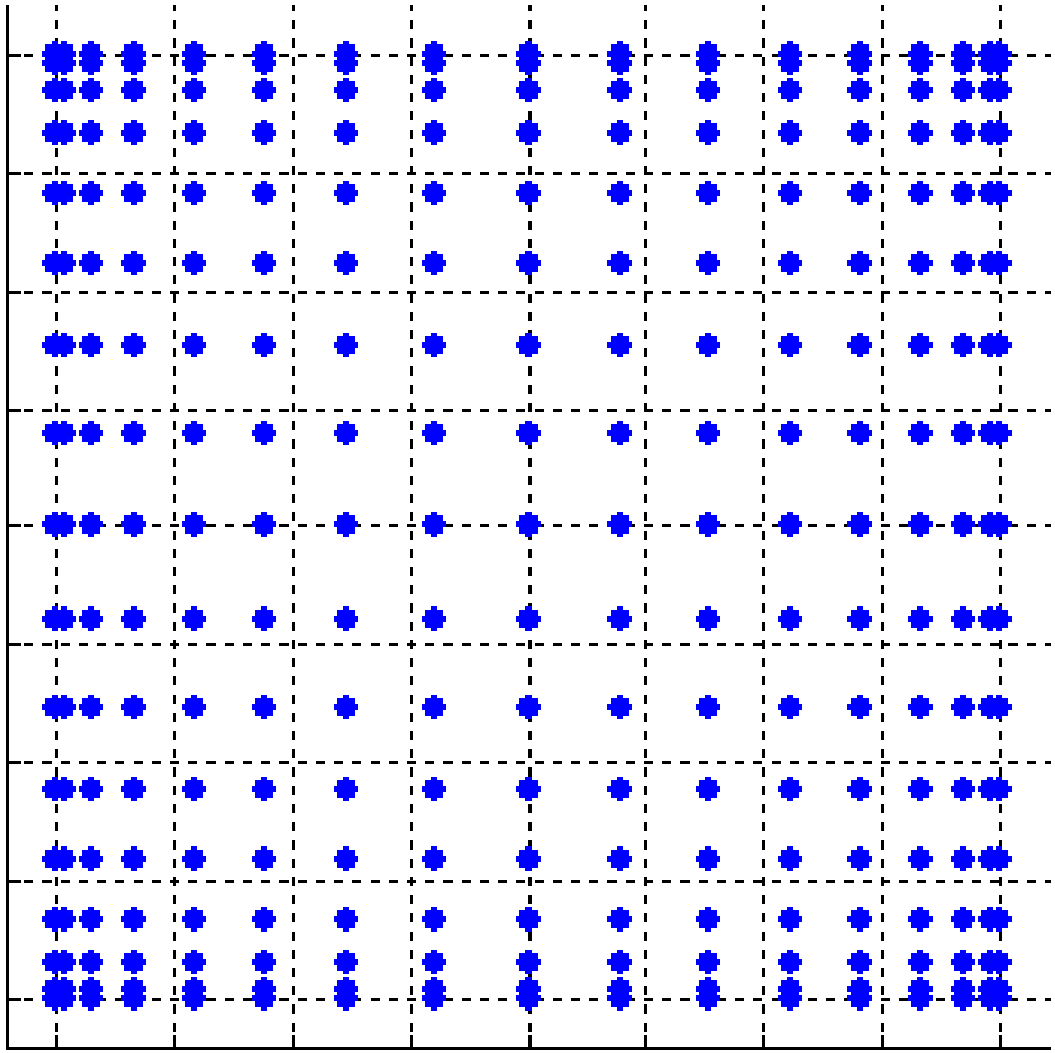
To capture only "desirable" monomials, we essentially add product grids which are sparse in one direction if dense in the other.

Because of nesting, the grids reuse many points.

The big savings comes from entirely eliminating most of the points of the full product grid.

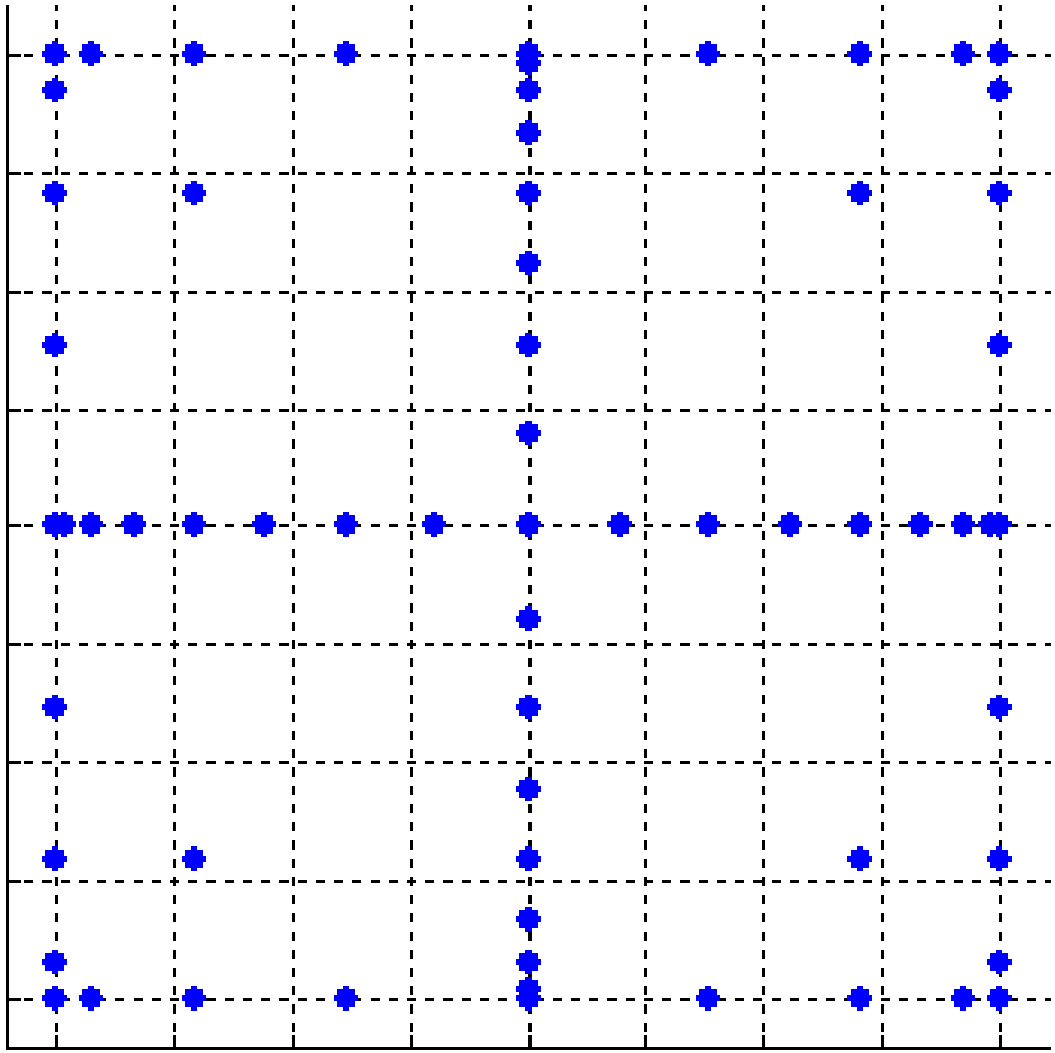
The improvement is greater as the dimension or level increases.

SMOLYAK QUADRATURE: 2D Order 17 Product Rule



A 17x17 product grid (289 points).

SMOLYAK QUADRATURE: 2D Level4 Smolyak Grid



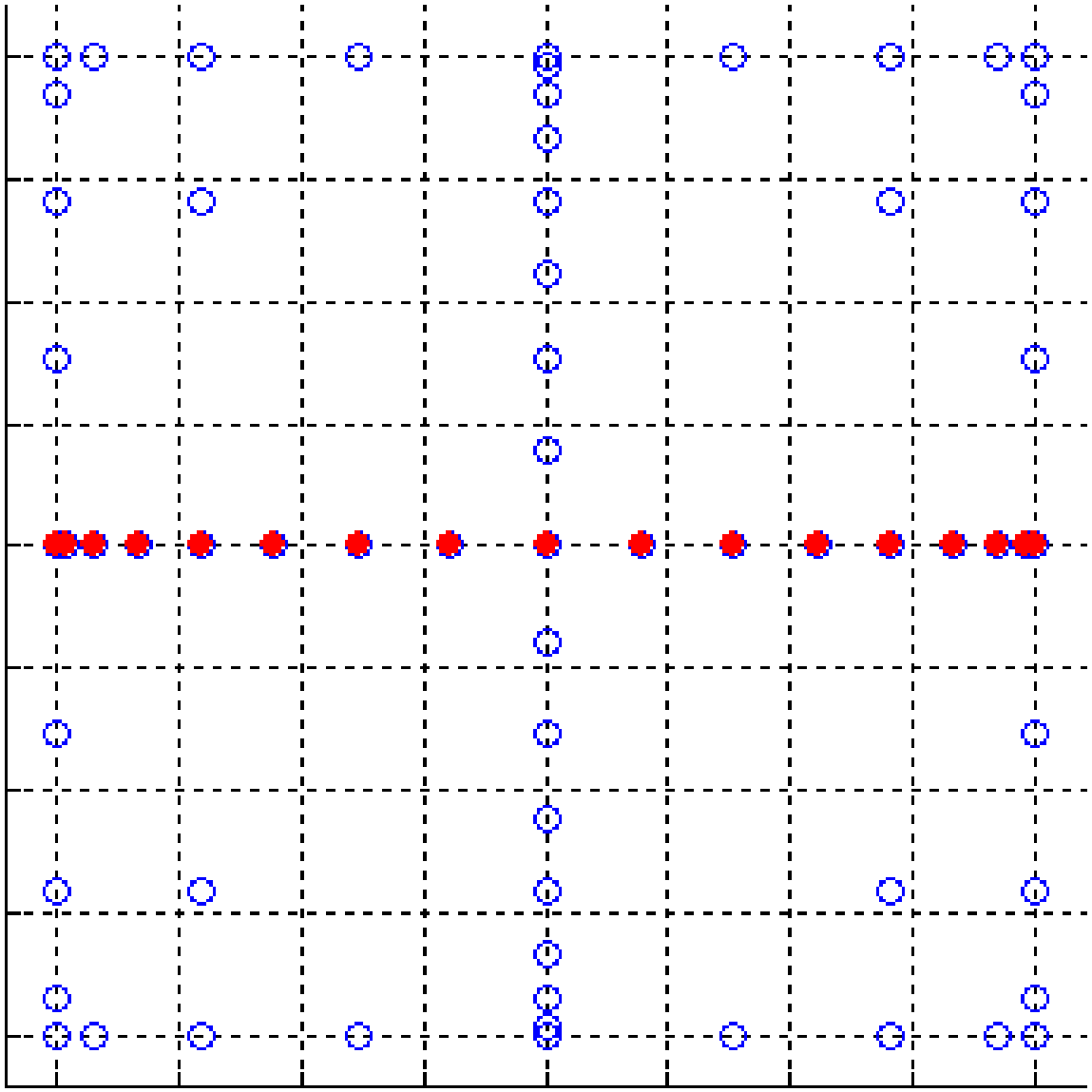
An "equivalent" sparse grid (65 points).

SMOLYAK QUADRATURE

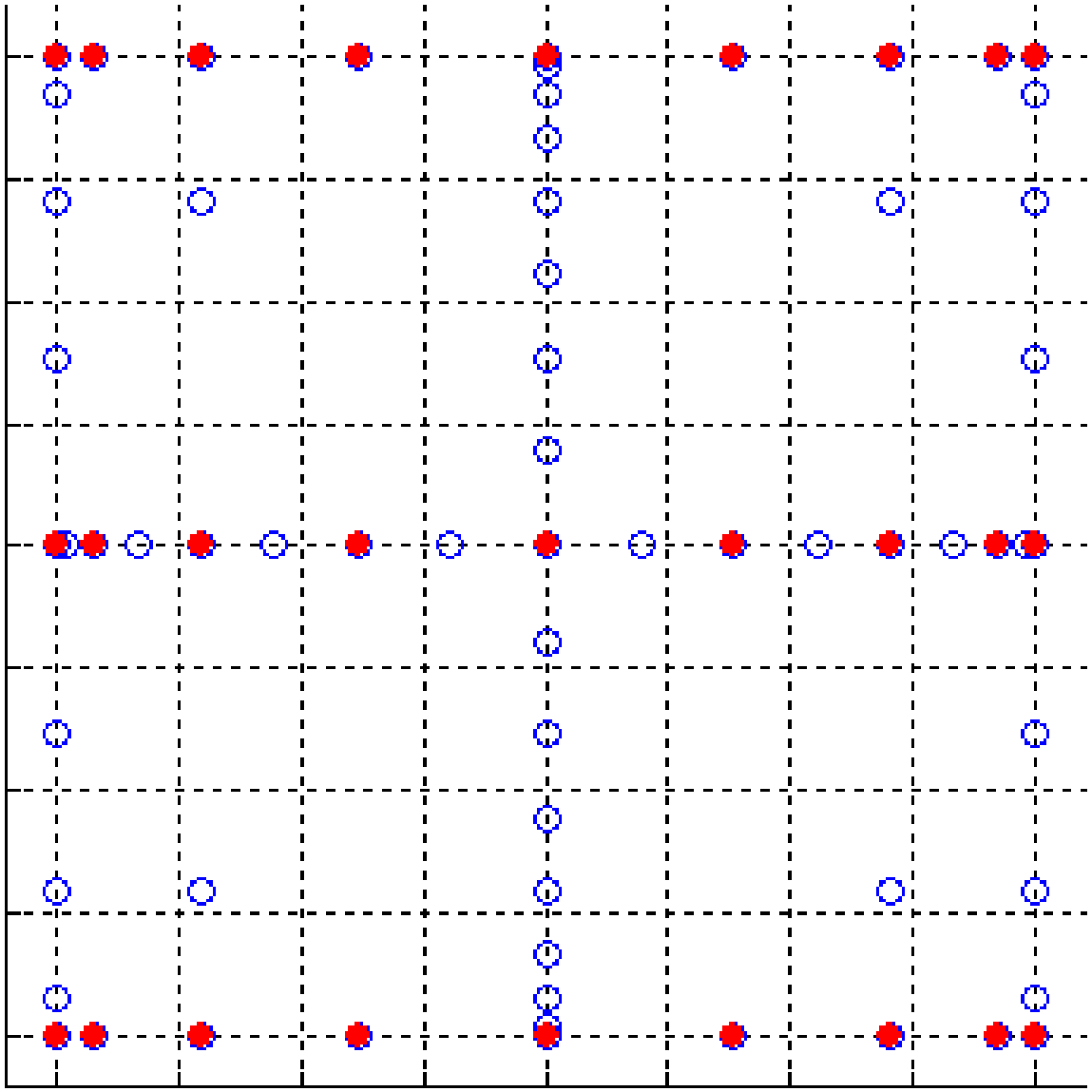
$$\mathcal{A}(6,2) = \sum_{6-2+1 \le |\mathbf{i}| \le 6} (-1)^{6-|\mathbf{i}|} \begin{pmatrix} 2-1 \\ 6-|\mathbf{i}| \end{pmatrix} (\mathcal{U}^{i_1} \otimes \mathcal{U}^{i_2})$$

$$\begin{aligned}
&= -\mathcal{U}^1 \otimes \mathcal{U}^4 \quad (1x9) \\
&\quad -\mathcal{U}^2 \otimes \mathcal{U}^3 \quad (3x5) \\
&\quad -\mathcal{U}^3 \otimes \mathcal{U}^2 \quad (5x3) \\
&\quad -\mathcal{U}^4 \otimes \mathcal{U}^1 \quad (9x1) \\
&\quad +\mathcal{U}^1 \otimes \mathcal{U}^5 \quad (1x17) \\
&\quad +\mathcal{U}^2 \otimes \mathcal{U}^4 \quad (3x9) \\
&\quad +\mathcal{U}^3 \otimes \mathcal{U}^3 \quad (5x5) \\
&\quad +\mathcal{U}^4 \otimes \mathcal{U}^2 \quad (9x3) \\
&\quad +\mathcal{U}^5 \otimes \mathcal{U}^1 \quad (17x1)
\end{aligned}$$

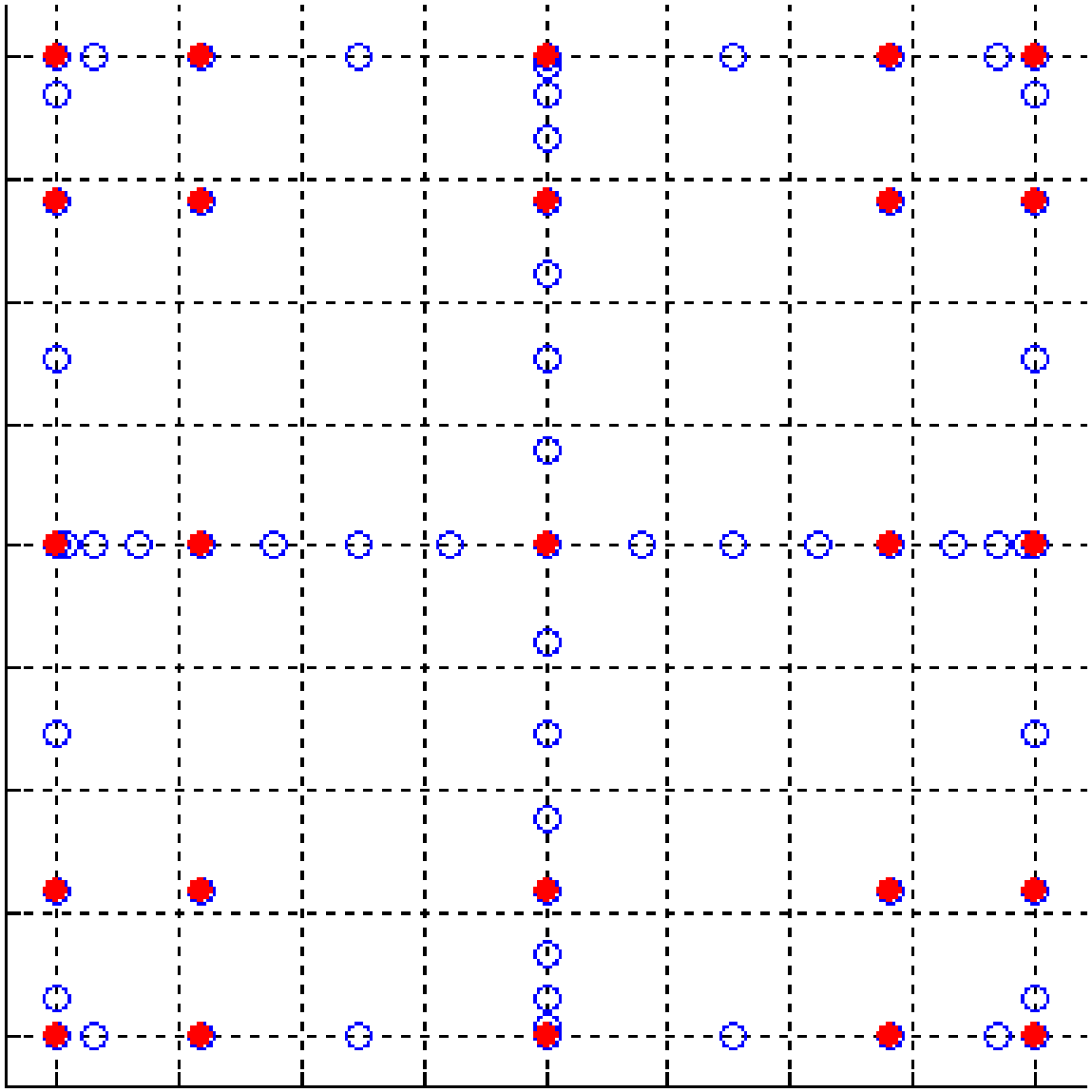
SMOLYAK QUADRATURE: 2D Level4 17x1 component



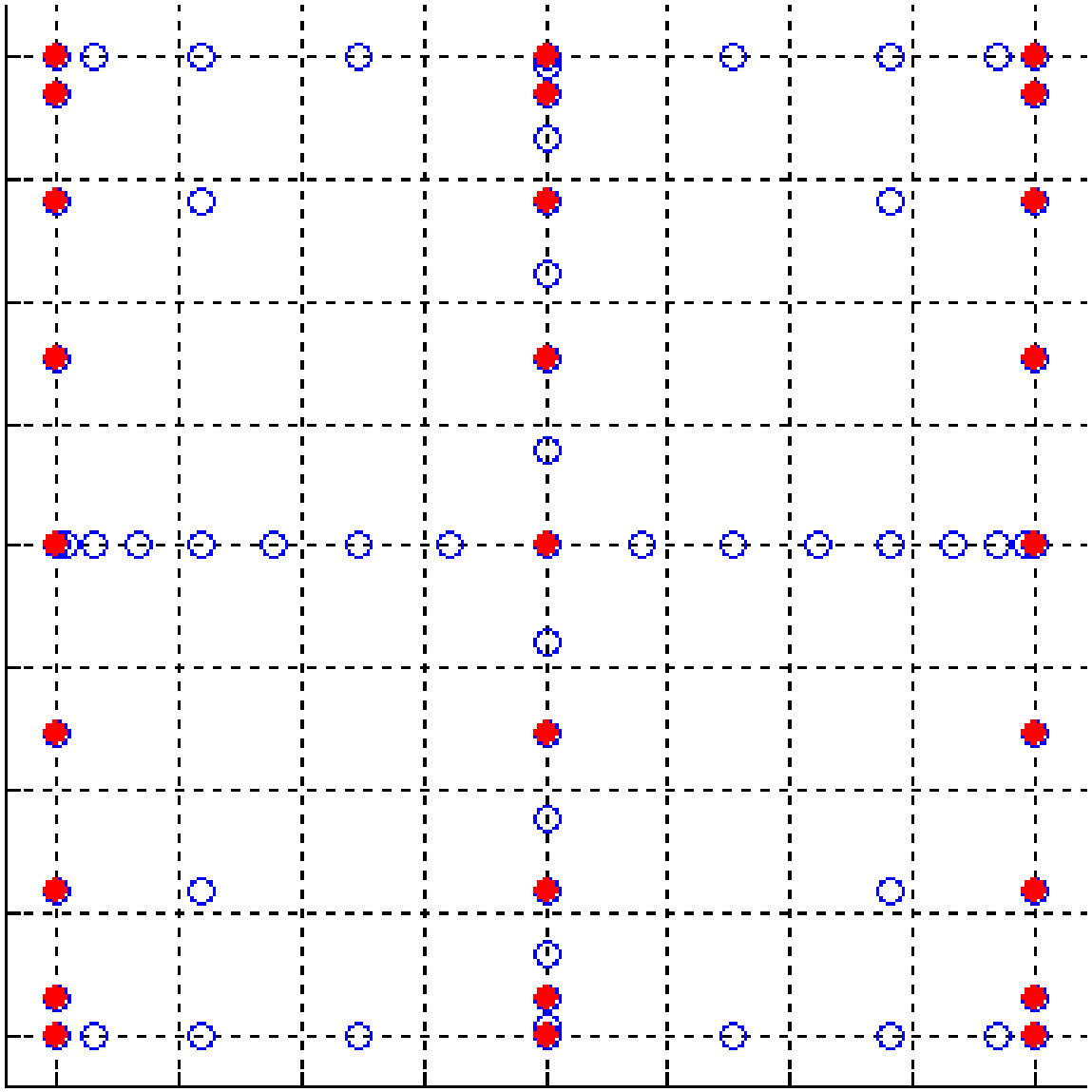
SMOLYAK QUADRATURE: 2D Level4 9x3 component



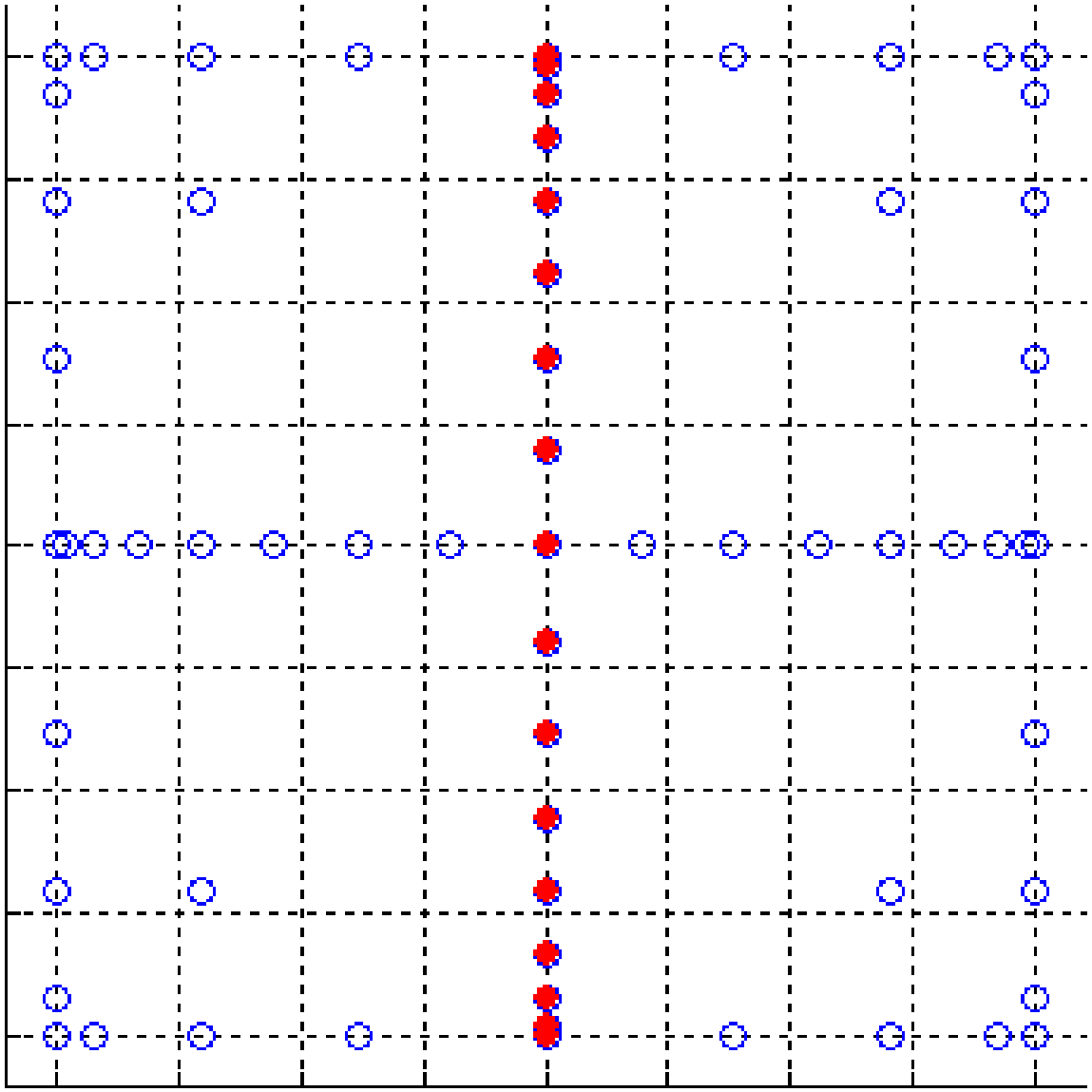
SMOLYAK QUADRATURE: 2D Level4 5x5 component



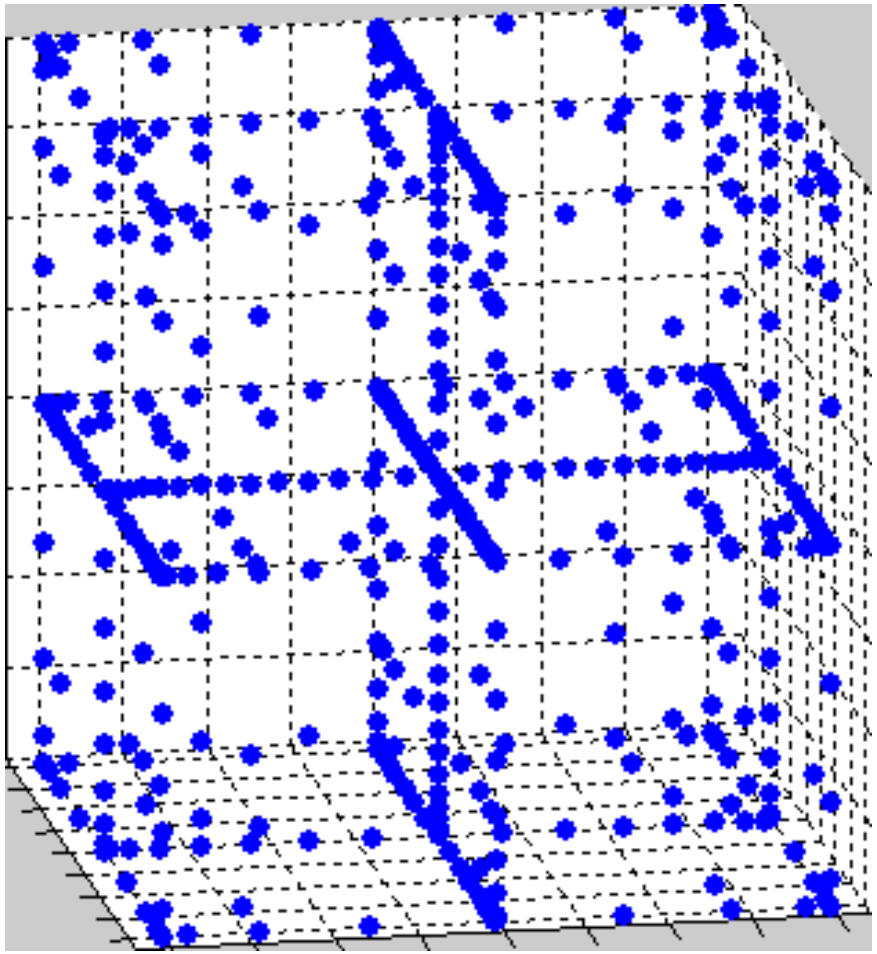
SMOLYAK QUADRATURE: 2D Level4 3x9 component



SMOLYAK QUADRATURE: 2D Level4 1x17 component



SMOLYAK QUADRATURE: 3D Level5 Smolyak Grid



Sparse grid = 441 points; Product grid would have 35,937.

SMOLYAK QUADRATURE

The orders of the 9 product rules forming $A(6,2)$ sum to 136 abscissas. But if we choose a nested family of rules, such as Clenshaw Curtis, the non-duplicated total is 65.

If we are computing a series of estimates, and have already calculated $A(5,2)$, then the 29 abscissas from that calculation are included in $A(6,2)$ (although the function values are multiplied by different weights) so we'd only have to evaluate the function at 36 new abscissas.

Economy in the number of abscissas can be vital when function evaluations are expensive. $f(x)$ might be available as a trivial formula, or as the output of a lengthy simulation.

SMOLYAK QUADRATURE

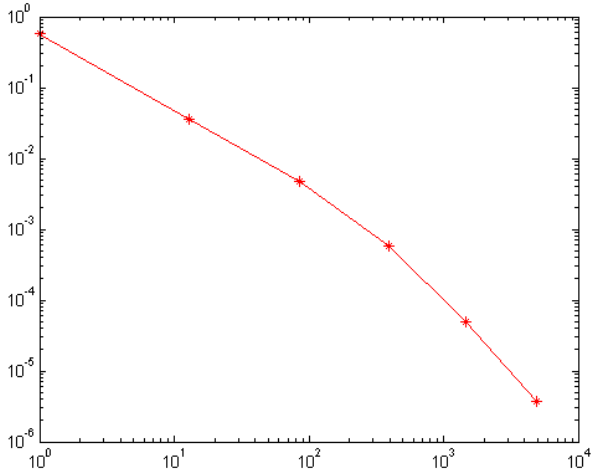
The careful observer will note that even if the product rules have positive weights, this is not necessarily the case for the resulting Smolyak rule.

In fact, it is common to have many negative weights in such a rule.

Among other issues, this means that it is possible for a Smolyak rule to return a *negative* estimate for the integral of a *uniformly positive* quantity!

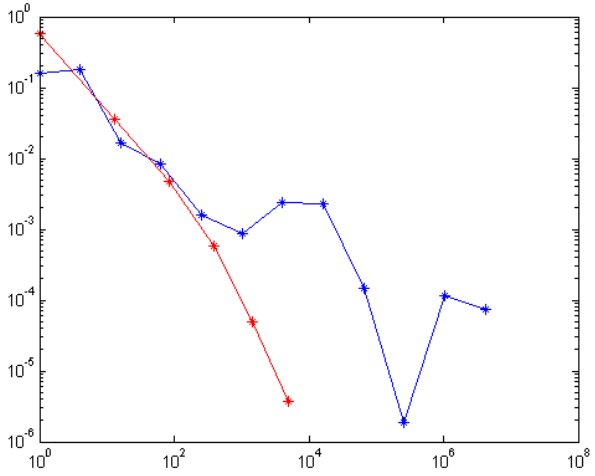
We will see an example of this behavior when we (inappropriately) try to integrate a discontinuous function.

SMOLYAK QUADRATURE: 6D Sparse Grid



N	Estimate	Error
1	0.062500	0.573282
13	0.600000	0.0357818
85	0.631111	0.00467073
389	0.636364	0.000582152
1457	0.635831	0.0000492033
4865	0.635778	0.00000375410
∞	0.635782	0.0000

SMOLYAK QUADRATURE: Monte Carlo vs Sparse Grid



SG N	SG Estimate	—	MC N	MC Estimate
1	0.062500	—	1	0.796541
13	0.600000	—	16	0.652621
85	0.631111	—	256	0.637351
389	0.636364	—	4096	0.633428
1457	0.635831	—	65536	0.635926
4865	0.635778	—	1048576	0.635666
∞	0.635782	—	∞	0.635782

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!

SPARSE GRIDS: High Dimensional Data => Information

DEGREE & PRECISION

In multi-dimensions, what is the **DEGREE** of a monomial?

If we consider the **component degree**, (the maximum of the degrees of the component variables) then monomials of component degree 4 include x^4 and x^3y^2 and even x^4y^4.

If we consider the **total degree**, we sum all the exponents. Then monomials of total degree 4 are exactly

$$x^4, x^3y, x^2y^2, xy^3, y^4.$$

The asymptotic accuracy of a quadrature rule is determined by the highest **total degree** N for which we can guarantee that all monomials will be integrated exactly.

As soon as we miss one monomial of a given total degree, our rule will have "run out of accuracy".

DEGREE & PRECISION

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				

Monomials up to 4th degree. Those below the line **are not needed** ..they don't help the asymptotic accuracy.

DEGREE & PRECISION

As the dimension increases, the useless monomials predominate.

Suppose we take products of a 10 point with accuracy 9. In a spatial dimension of N, how many monomials are there of total degree 9 or less? If we estimate that we need 1 point for each such monomial, how many points are we wasting?

Dim	Points	Good	Useless	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	2002	97,998	97%
6D	1,000,000	5005	994,995	99%

In 5D, there are only 2002 items to search for.
Can't we find a quadrature rule of roughly that order?

DEGREE & PRECISION

Another way to understand what the power of a quadrature rule may be is to ask for its **degree of precision**.

A typical precision property states that a given quadrature rule will produce the exact answer (zero error) when the integrand function is a polynomial whose degree is no more than a given limit.

If a family of quadrature rules has a precision property (which improves as you increase the order of the rule), you can assume the rule will eventually "capture" the integral of any function that is polynomial, and will "chase" (reliably drive down the error) the integral of any function well approximated by polynomials (bounded derivatives up to the degree of precision).

DEGREE & PRECISION

Monte Carlo rules have a (weak) precision property: every Monte Carlo rule is exact for polynomials of degree 0...but for no higher degree.

Interpolatory quadrature rules of order N typically have precision up to degree $N-1$ or N, and Gauss rules have precision of $2N-1$; in both cases, this property extends naturally to product rules.

Sparse grids inherit a precision property based on the property of the rules forming the underlying product grids. Because Gauss rules generally don't nest, a less precise but nested rule like Clenshaw Curtis may be preferable if available.

DEGREE & PRECISION: A Rule for 100D

As an extreme example, let us consider constructing quadrature rules for the hypercube $[-1,+1]^{100}$.

The family of **product rules** for such a region would start out with a 1 point rule, followed immediately by a rule of order $2^{100} \approx 10^{30}$. The **number** of points is too large to store as a 32 bit integer on a computer!

The **sparse grid** family also begins with a 1 point rule. The second rule contains...201 points.

DEGREE & PRECISION; A Rule for 100D

The 201 point sparse grid rule has a degree of precision of 3, so it **exactly** integrates

- any constant function
- combinations of linear monomials, x_1, x_2, ...,x_{100}
- quadratic combinations such as x_1^2 or $x_{37}x_{93}$;
- cubic monomials such as x_1^3, $x_4^2x_8$ or $x_1x_2x_3$;

This rule is precise for $1 + 100 + 10{,}000 + 1{,}000{,}000$ integrands. This is less amazing when we realize that linears and cubics integrate to zero.

Nonetheless, there are 201 "interesting" integrands left, and we get them all, efficiently and precisely.

DEGREE & PRECISION

I'm not suggesting that integrating in 100D is a good idea.

However, I've shown that it's easy to generate an extreme case for which product rules cannot even be constructed, and for which Monte Carlo rules cannot be run long enough to achieve any accuracy, but for which sparse grids can pull out **some** information.

By the way, the next sparse grid rule for 100D is of size 20,201, certainly a jump from 201 points, but not enormous. This rule adds quartics and quintics to the precision.

SPARSE GRIDS: High Dimensional Data => Information

SMOOTHNESS

Interpolatory and Gauss-type rules provide improved answers by increasing the order (number of points), thereby raising the degree of interpolation, and leaving an error whose leading term involves the derivatives of one higher degree.

This assumes all such derivatives exist and are bounded.

Sparse grids based on product rules of interpolatory or Gauss-type rules inherit this limitation.

Integrands with badly behaved derivatives, singularities, or discontinuities will **poison** the calculation.

SMOOTHNESS: the 6D Sphere

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

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

Apply (foolishly) 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	:	...	...

Do you remember 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!

SPARSE GRIDS: High Dimensional Data => Information

STOCHASTIC PROBLEMS: A Diffusion Equation

Consider a diffusion equation with a stochastic component in the diffusion coefficient $a(\omega; x, y)$.

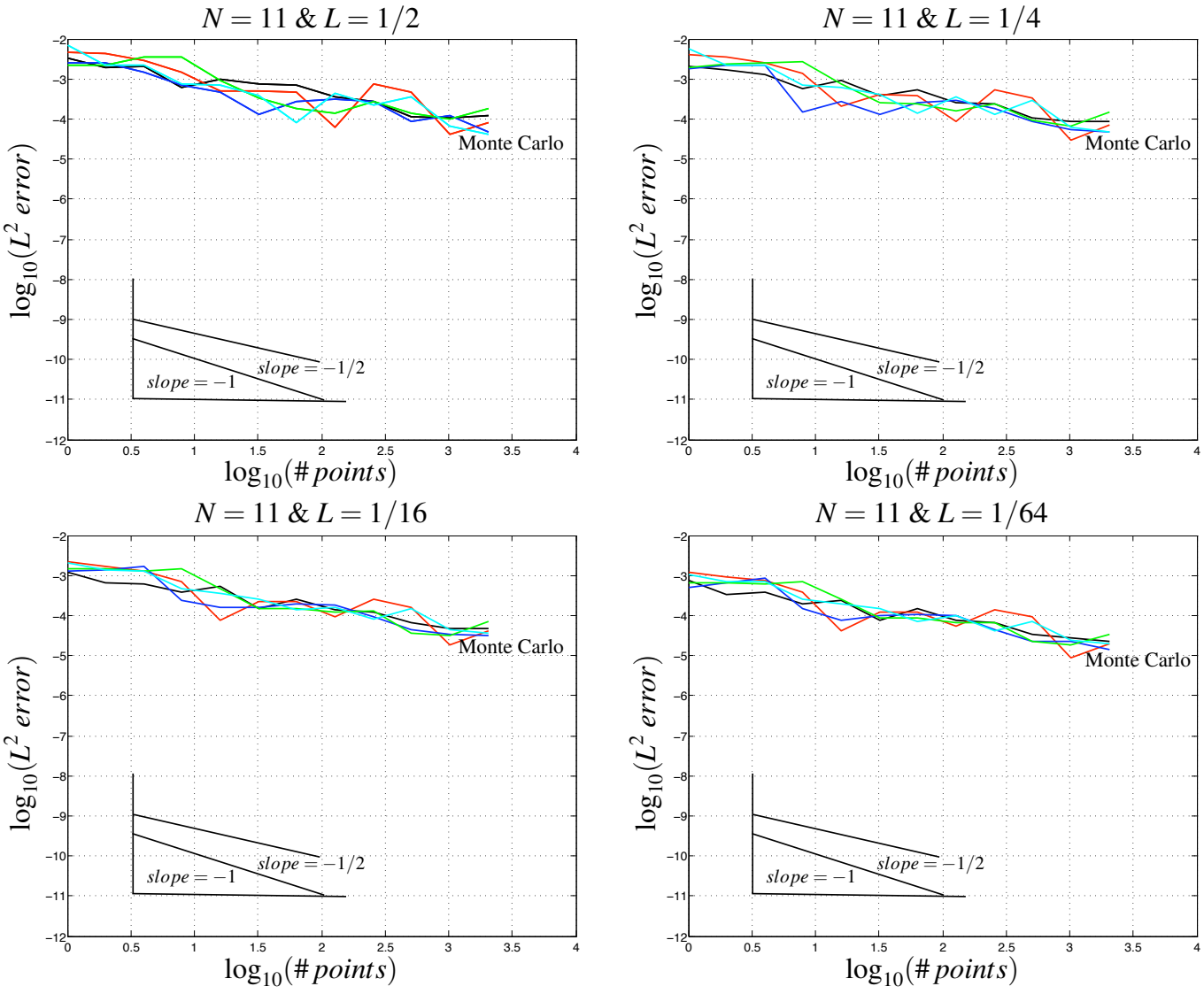
$$-\nabla \cdot (a(\omega; x, y)\, \nabla u(\omega; x, y)) = f(x, y)$$

Integrating over the probability space gives us expected value information (or moments, and so on).

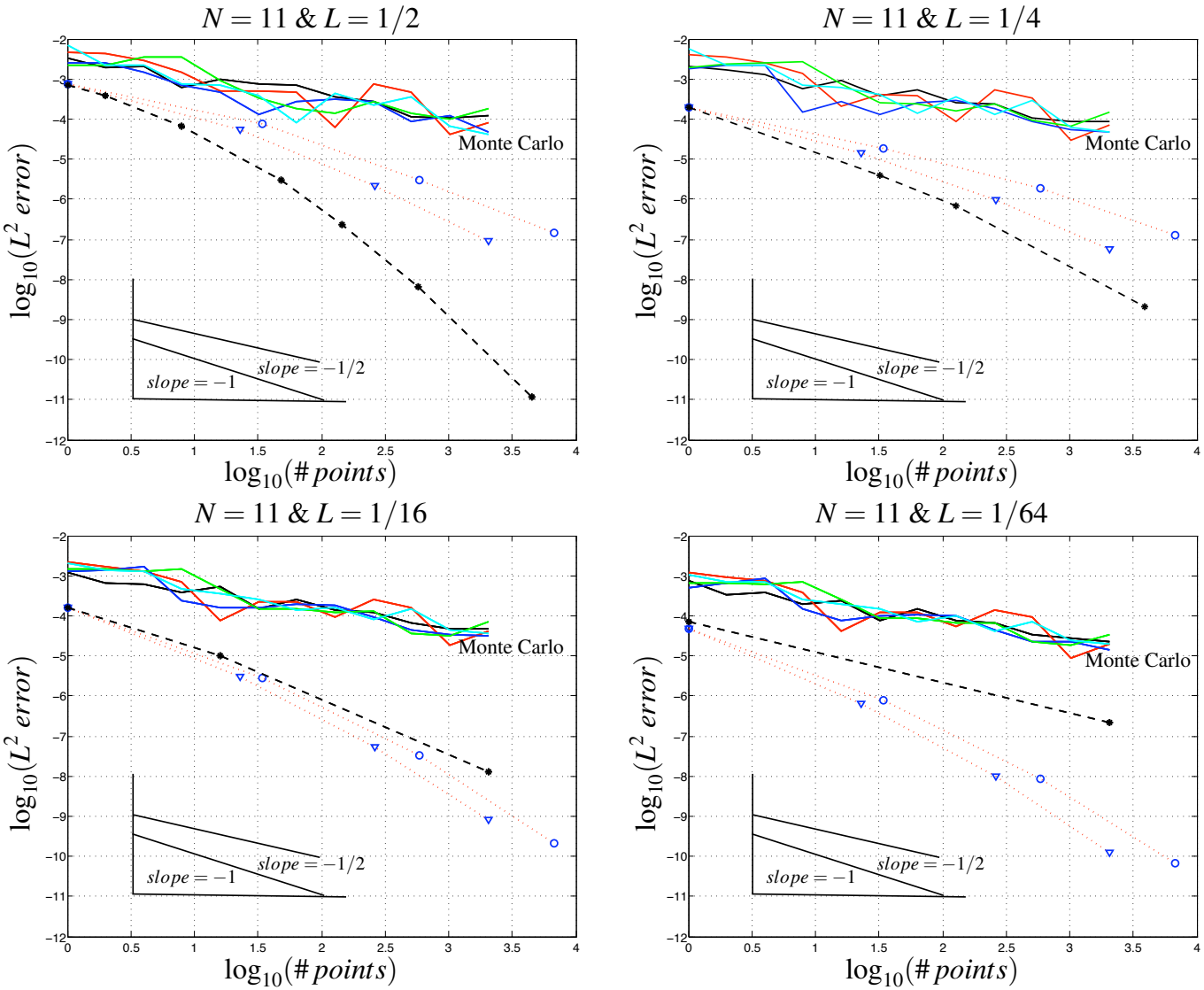
$$E(u(x, y)) = \int_{\Omega} u(\omega; x, y)\, pr(\omega)\, d\omega$$

[Clayton Webster, Thesis, 2007]

STOCHASTIC PROBLEMS: Monte Carlo



STOCHASTIC PROBLEMS: Monte Carlo vs Smolyak



SPARSE GRIDS: High Dimensional Data => Information

Software Implementation: 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 Implementation: 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;
```

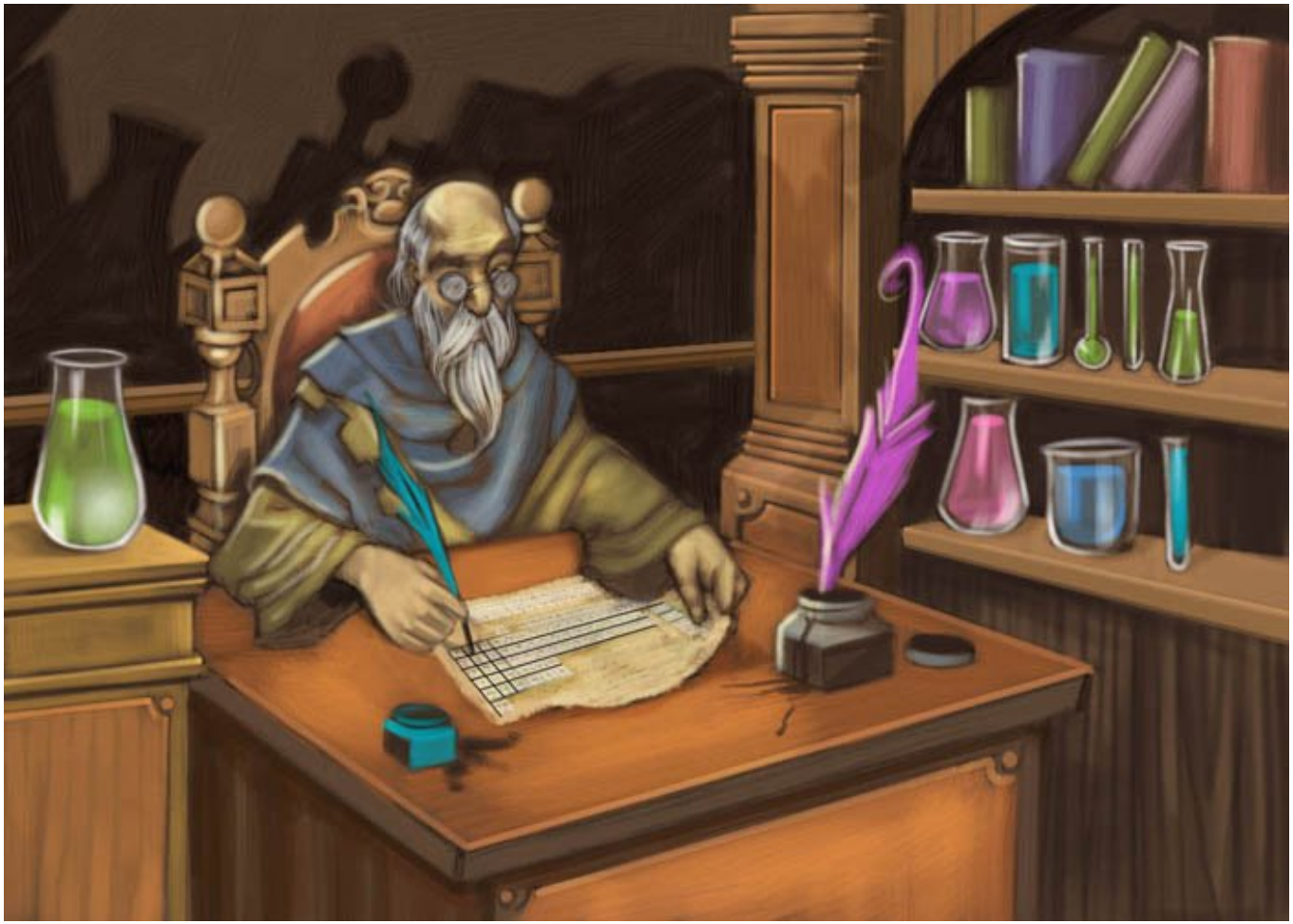
Software Implementation: Sparse Grid Mixed Weight

```
    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
```

SPARSE GRIDS: High Dimensional Data => Information

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

```
------
0.111
0.444
0.111
0.444
1.777
0.444
0.111
0.444
0.111
```

X file

```
-----------
-1.0  -1.0
-1.0   0.0
-1.0  +1.0
 0.0  -1.0
 0.0   0.0
 0.0  +1.0
+1.0  -1.0
+1.0   0.0
+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
0.0416666666666665	4	4 0
0.0000000000000001	4	3 1
0.0000000000000000	4	2 2
0.0000000000000001	4	1 3
0.0277777777777779	4	0 4

SPARSE GRIDS: High Dimensional Data => Information

CONCLUSION: A few observations

Sparse grids are based on product rules.

They achieve the accuracy of a high order product rule using a combination of lower order rules.

Sparse grid rules are suitable for integrands with bounded mixed derivatives.

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

CONCLUSION: Software and Data

SMOLPACK, a C library by Knut Petras for sparse integration.

SPINTERP, ACM TOMS Algorithm 847, a MATLAB library by Andreas Klimke for sparse grid interpolation.

http://people.sc.fsu.edu/∼burkardt...
.../cpp_src/sparse_grid_mixed/sparse_grid_mixed.html C++
.../f_src/sparse_grid_mixed/sparse_grid_mixed.html F90
.../m_src/sparse_grid_mixed/sparse_grid_mixed.html MATLAB
.../datasets/sparse_grid_mixed/sparse_grid_mixed.html

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.