Integration, Quadrature, and Sparse Grids

John Burkardt
Department of Computational Science
Florida State University

...

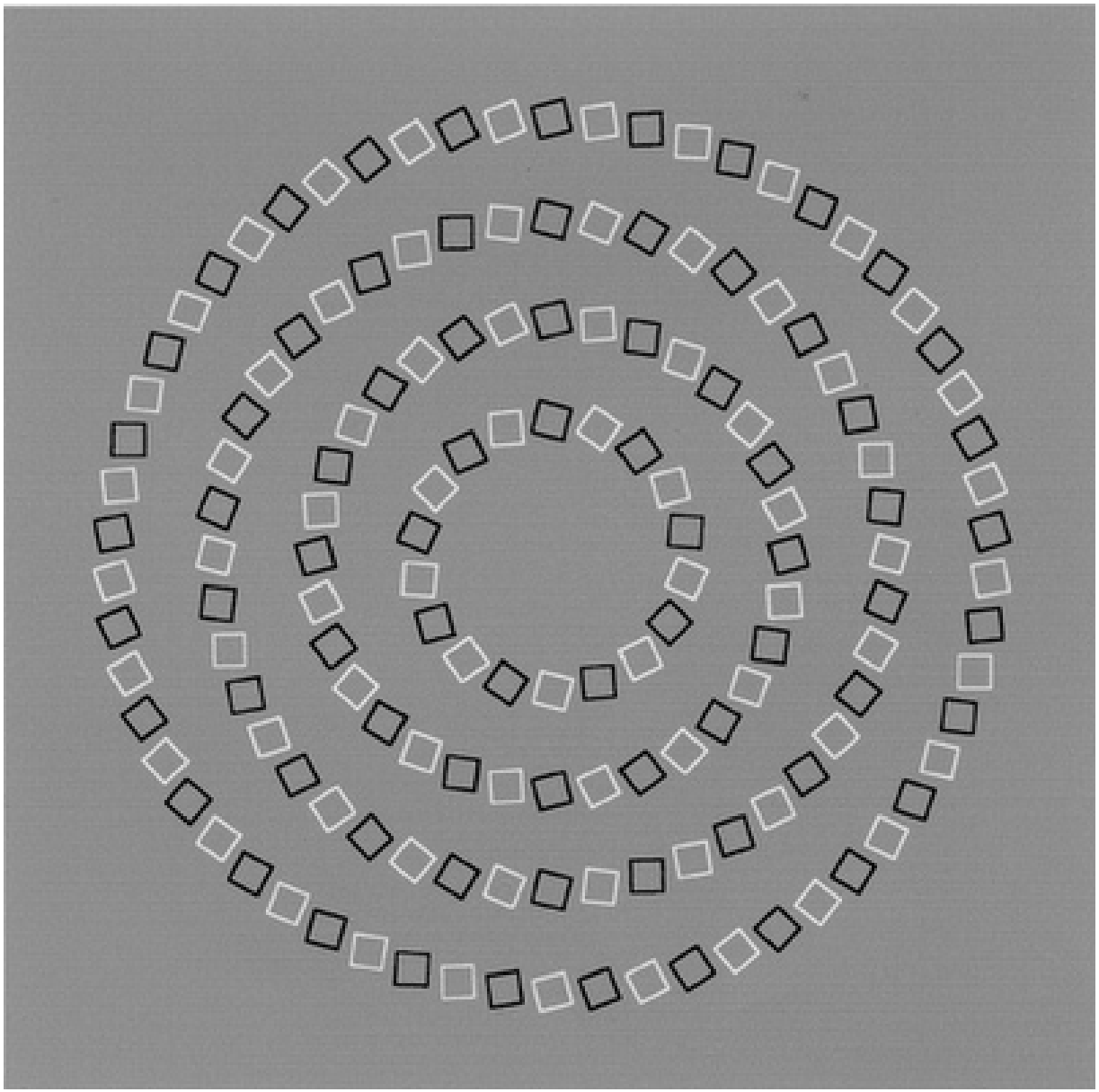
Research Group Seminar

...

http://people.sc.fsu.edu/~jburkardt/presentations/...
sparse_2010_fsu.pdf

10 September 2010

I Promise That Things Will Become Clear!



Integration: Begin

1. **INTEGRATION**
2. Quadrature
3. Interpolatory Quadrature
4. Product Rules
5. Smolyak Quadrature
6. Sparse Grid Software
7. Conclusion

Integration

Integration is is used for the mathematical expression of physical laws.

A complicated thing is understood by **adding** tiny components.

Integration is the limit of a summing operation.

$$\int_a^b f(x)\,dx = \lim_{h\to 0} \sum_{\text{Partition}_{[a,b]}} f(x_i)\Delta x_i$$

which (we will need to remember!) suggests that a sum can approximate an integral if the maximum subinterval size is small.

Integration

Geometric interpretation: area under the curve $y = f(x)$.

$$G[a, b] = \int_a^b f(x)\, dx$$

Sampling interpretation: average value (integral divided by length).

$$\overline{f(x)} = \frac{\int_a^b f(x)\, dx}{(b - a)}$$

Integration: Multiple Dimensions

Integration: Multiple Dimensions

We are so used to seeing double integrals as iterated integrals that we don't even think about it.

If the integration region is regular enough, then the integral over an area can be treated as the integral along one dimension of the integral along the other.

$$Volume[pool] = \int_{\text{base}} depth(x, y)\, dA = \int_c^d \left(\int_a^b depth(x, y)\, dx \right) dy$$

This means that if we can figure out how to approximate an integral in 1D, then we automatically have a way to approximate in 2D, 3D, or any dimension. (It will work, but it might not be the best way!)

Integration: Multiple Dimensions: More than 3!

Modern computations now involve integrals over high dimensions.

- Financial mathematics: 30D or 360D
- ANOVA decompositions: 10D or 20D
- 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)
- Modeling unobservable groundwater flow

Integration: No Formulas for Interesting Problems!

Freshman memorize "antiderivatives" of formulas $f(x)$.

$$\int x^3 \, dx = \frac{x^4}{4} + C$$

But most formulas have no antiderivative!

And most things we want to integrate are not formulas but implicit functions!

$$\int_\Omega \nabla \mathbf{v}^h \cdot \nabla \psi_i + \nabla p^h \, \psi_i \, d\Omega = ?$$

and computationally, the only thing we can do is ask for the value of such an implicit function at various points.

Integration: Many functions are implicit

In many cases, the function we want to integrate is not a formula. Its value is only obtainable by an expensive indirect computation.

For example, we might have a temperature function $u(x, y, z; \omega)$ which depends on low dimensional spatial parameters and a stochastic parameter ω.

If we choose a value of ω, we can solve the state equations for u as a function of (x,y,z) by integrating over the geometric space (2D, 3D?).

If we want to know the influence of the stochastic parameter, we have to integrate over all possible value of ω, to get an expected value function $\overline{u}(x, y, z)$. But ω may lie in a 10D or 50D space.

Quadrature: Begin

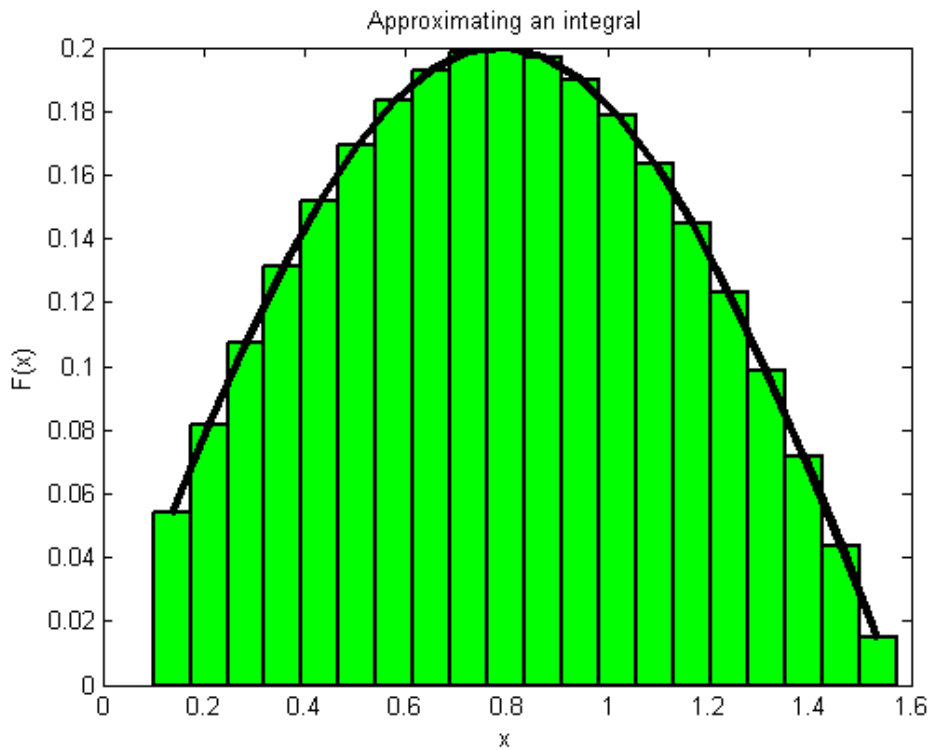
1. Integration
2. **QUADRATURE**
3. Interpolatory Quadrature
4. Product Rules
5. Smolyak Quadrature
6. Sparse Grid Software
7. Conclusion

Quadrature: Approximating an 1D Integral

Quadrature allows us to estimate integrals by weighted sums.

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

where the w_i and x_i may be chosen by a variety of criteria.



Quadrature: 1D Monte Carlo

The *Monte Carlo* algorithm views the integral as an average:

$$\int f(x)\,dx \approx \frac{\sum_{i=1}^{N} f(x_i)}{N}$$

and concentrates on choosing x values that are well spread out.

Quadrature: 1D Monte Carlo

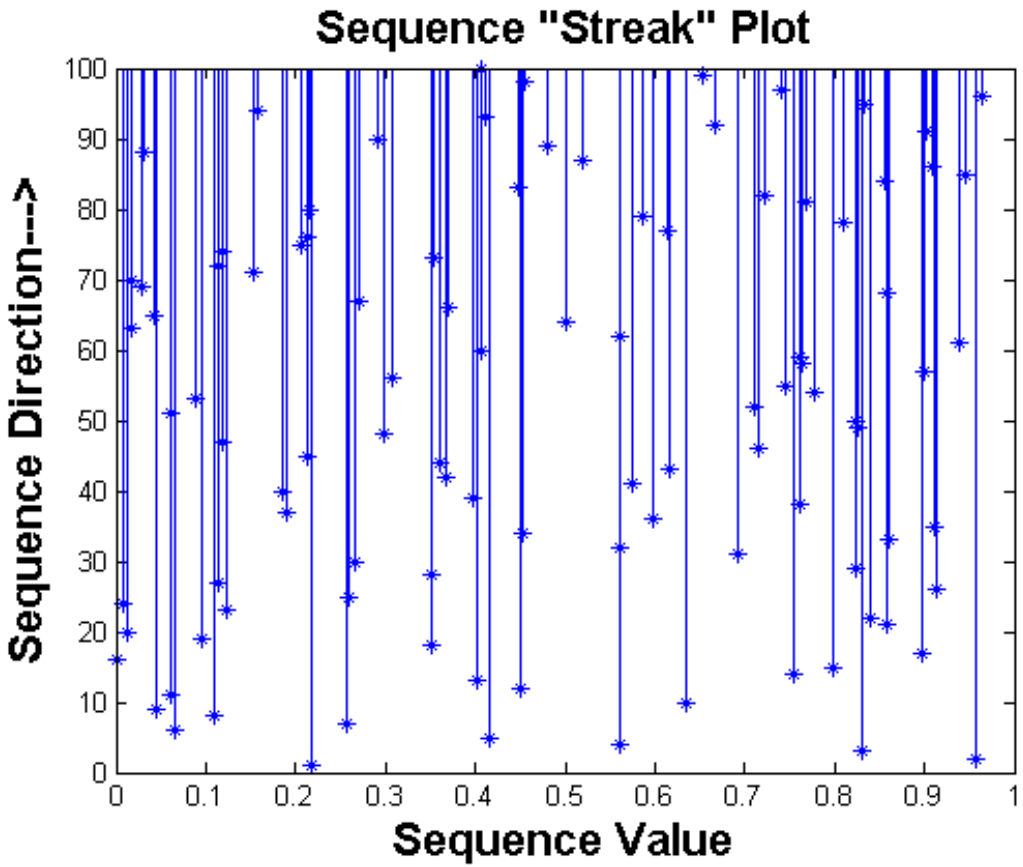
To improve an MC estimate, increase **N**, the size of your sample.

The Law of Large Numbers says that convergence will be like $\sqrt{N}$. To reduce the error by a factor of 10 (one more decimal place) requires 100 times the data.

- If more accuracy needed, current values can be included;
- Accuracy hampered because of large "gaps" in sampling.
- Accuracy improvement rate is independent of spatial dimension.

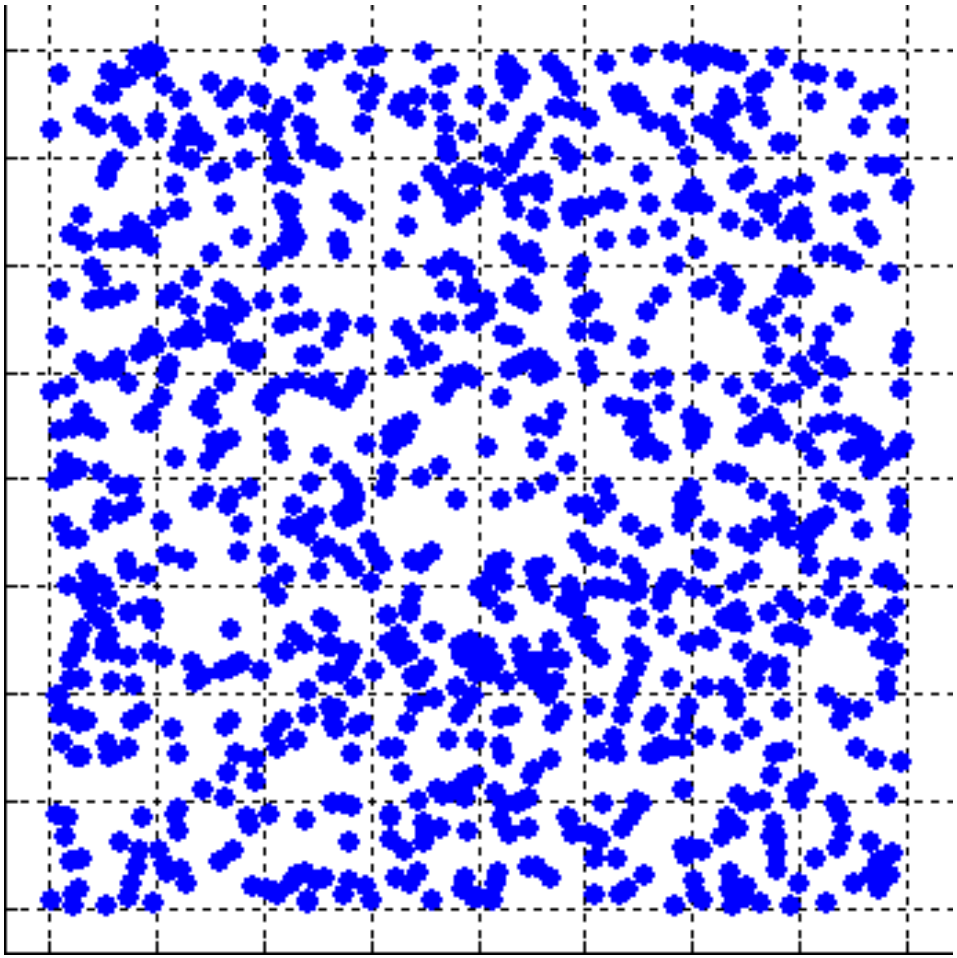
Quadrature: 1D Monte Carlo

Notice the "gaps" and "clusters" as this 1D sequence fills in.



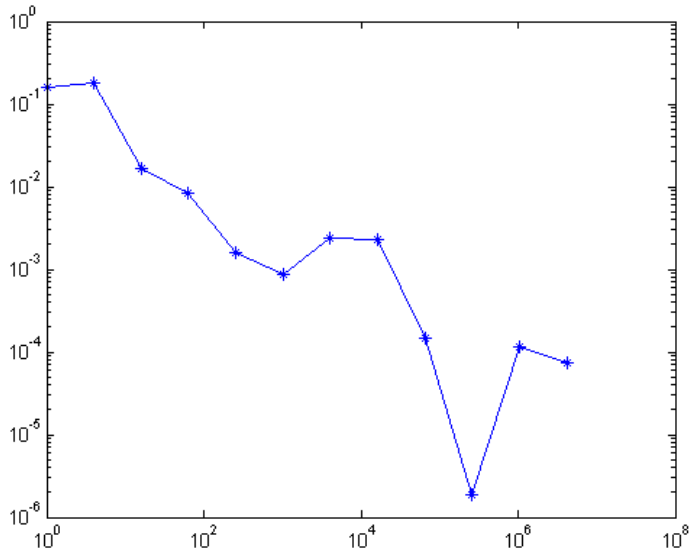
Quadrature: 2D Monte Carlo

Notice the "gaps" and "clusters" in this 2D sample.



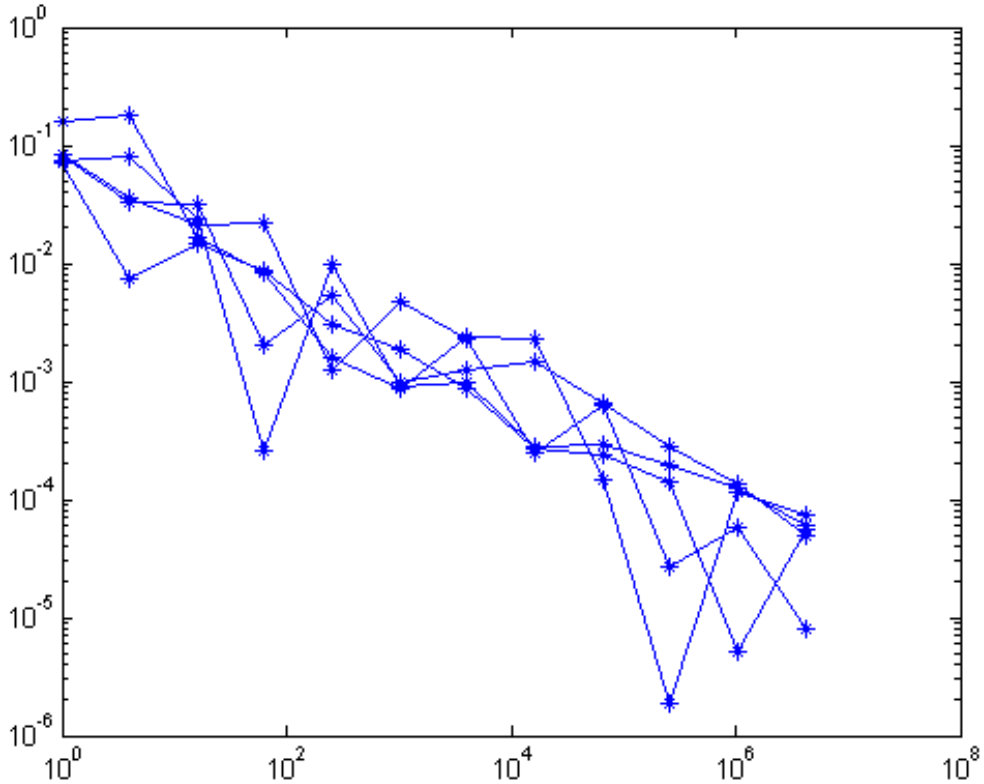
Quadrature: 6D Monte Carlo Error

N	Estimate	Error
1	0.796541	0.160759
16	0.652621	0.016838
256	0.637351	0.001569
65536	0.635926	0.000144
4194304	0.635856	0.000074
∞	0.635782	0.0000



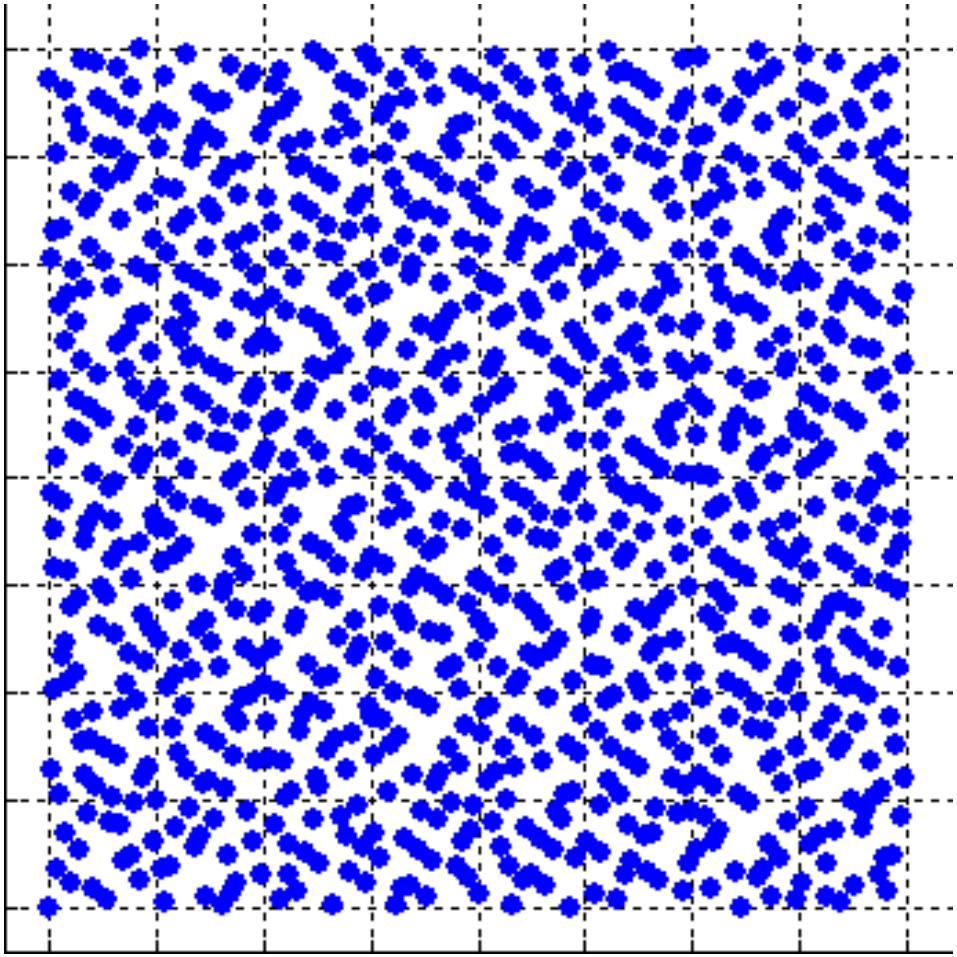
Quadrature: 6D Monte Carlo Error

If we try five times, we get five different sets of results.



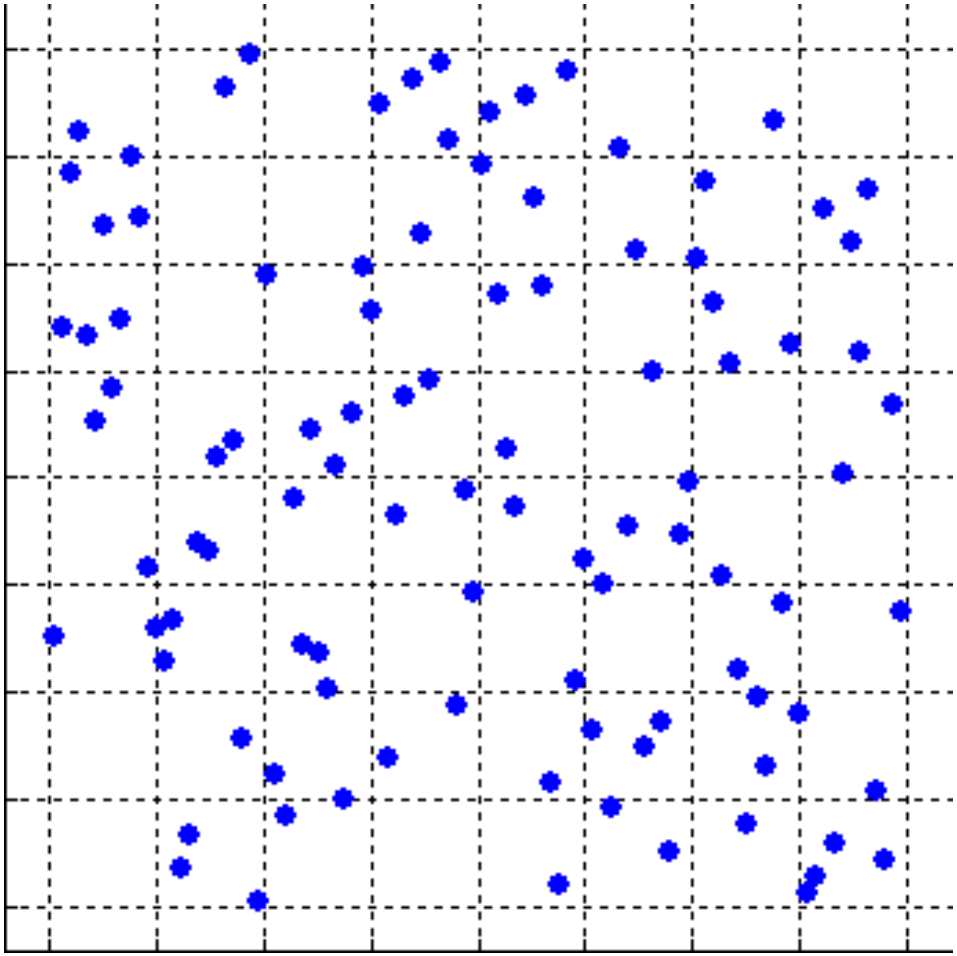
Quadrature: 2D Quasi Monte Carlo

Quasi-Monte Carlo methods produce well spaced sampling.



Quadrature: 2D Latin Hypercube

Latin Hypercube Sampling ensures good spacing in each 1D component (but allows gaps and clusters in multidimensions.)



Interpolatory Quadrature: Begin

1. Integration
2. Quadrature
3. **INTERPOLATORY QUADRATURE**
4. Product Rules
5. Smolyak Quadrature
6. Sparse Grid Software
7. Conclusion

Interpolatory Quadrature: An Alternative to Sampling

Sampling methods focus on the properties of **X**, the set of sample points.

No attempt is made to exploit the information returned in **F(X)**.

(Exception: Importance sampling and variance reduction versions of Monte Carlo methods modify their behavior based on the values of **F(X)** *encountered.*

If the function $f(x)$ is "well-behaved", the sample values **F(X)** contain strong clues about $f(x)$ and its integral.

Interpolatory Quadrature: 1D Example

Is $f(x)$ approximately a sum of monomials (powers of x)?

$$f(x) \approx 4.5 + 6.3x + 0.8x^2 + 2.1x^3 + 0.7x^4 + \ldots$$

If so, the beginning of the formula can be determined and integrated exactly.

This assumption is **not true** *for step functions, piecewise functions, functions with poles or singularities or great oscillation.*

Interpolatory Quadrature: 1D Example

To find the initial part of the representation, sample the function.

Evaluating at one point can give us the constant.

$$\mathbf{f(x)} \approx \mathbf{4.5}... + 6.3x + 0.8x^2 + 2.1x^3 + 0.7x^4 + ...$$

A second evaluation gives us the coefficient of x:

$$\mathbf{f(x)} \approx \mathbf{4.5} + \mathbf{6.3x}... + 0.8x^2 + 2.1x^3 + 0.7x^4 + ...$$

Evaluating at N points gives the first N coefficients.

Interpolatory Quadrature: Integrating Monomials

An approximate formula can be integrated exactly.

With N samples, we can integrate the first N monomials,

$$1, x, x^2, ..., x^{N-1},$$

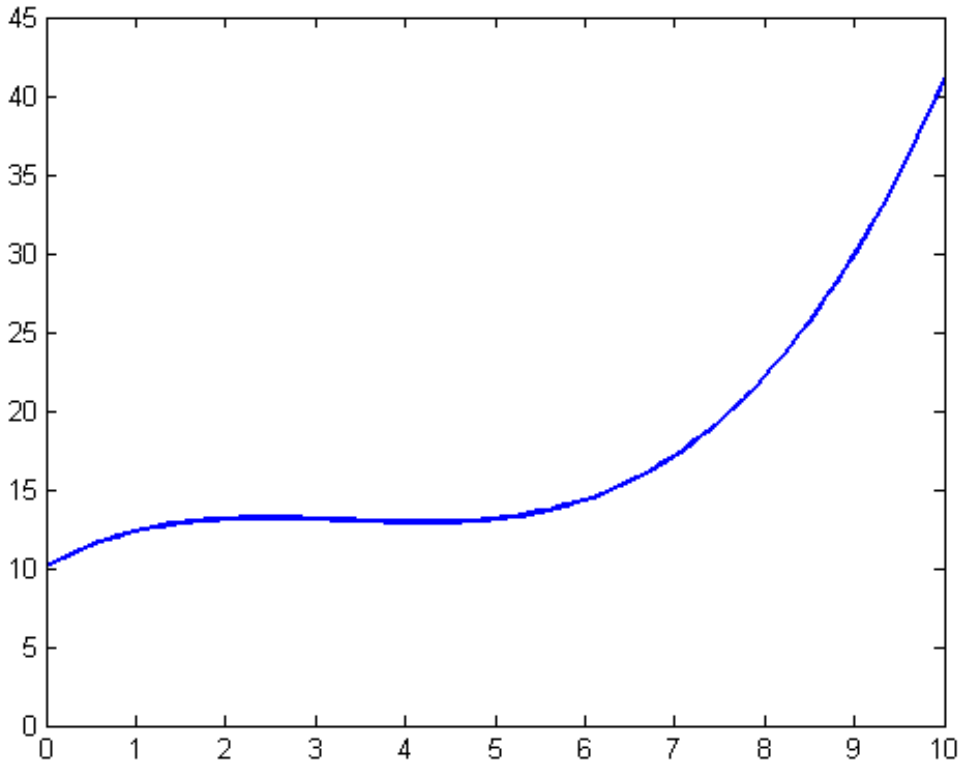
and all functions made up of them.

The error behaves like h^N, where h is the spacing between sample points.

Increasing N increases the monomials we can "capture".

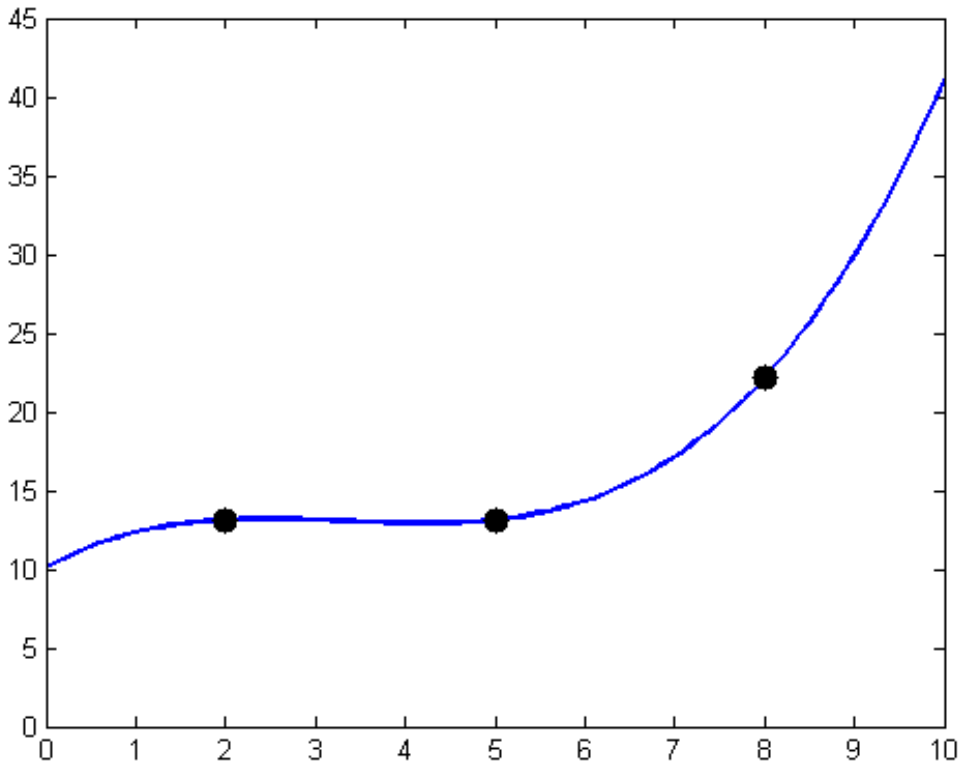
Interpolatory Quadrature: A Function to Integrate

A function f(x) is given.



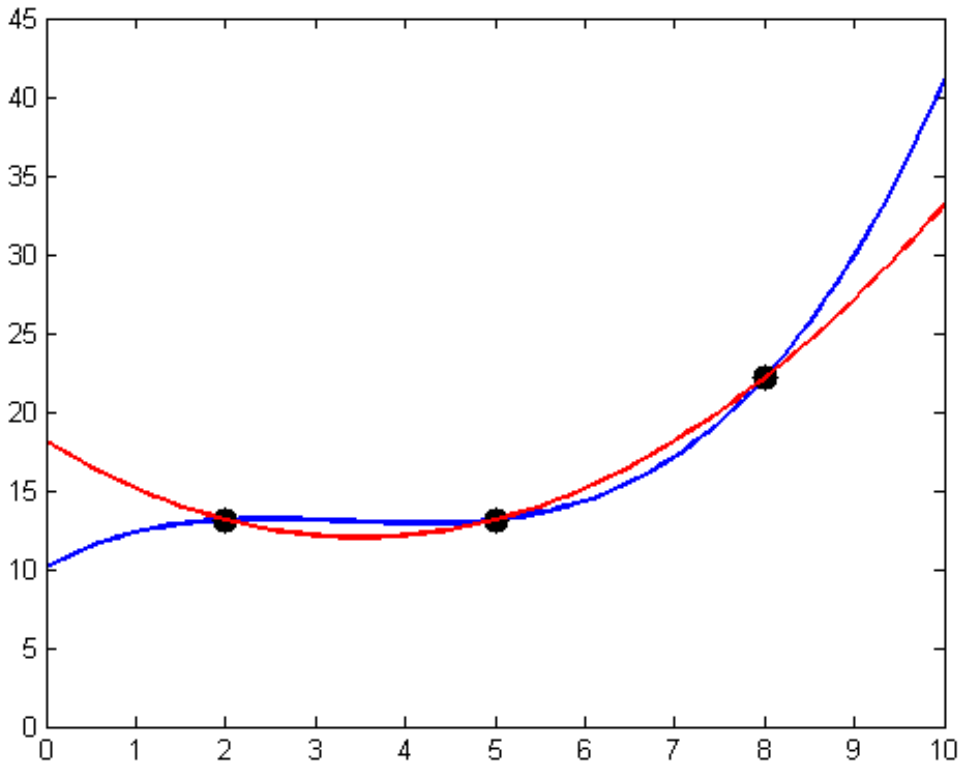
Interpolatory Quadrature: Selected Function Values

We evaluate it at N points.



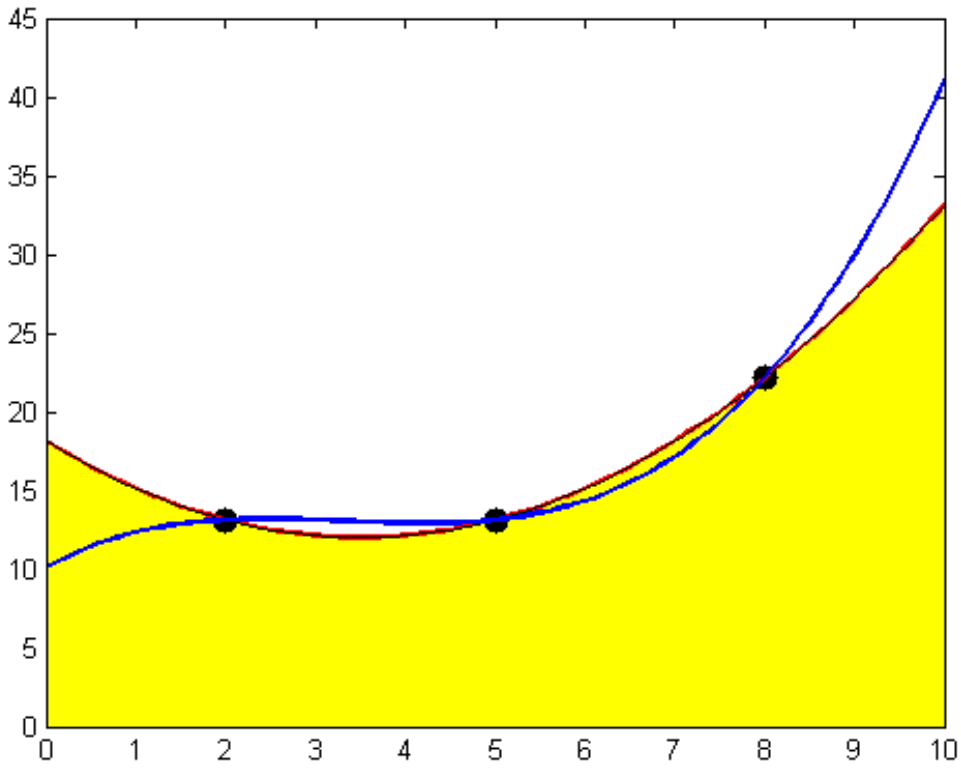
Interpolatory Quadrature: The interpolant

We determine the approximating polynomial.



Interpolatory Quadrature: The integrated interpolant

We integrate the approximating polynomial exactly.



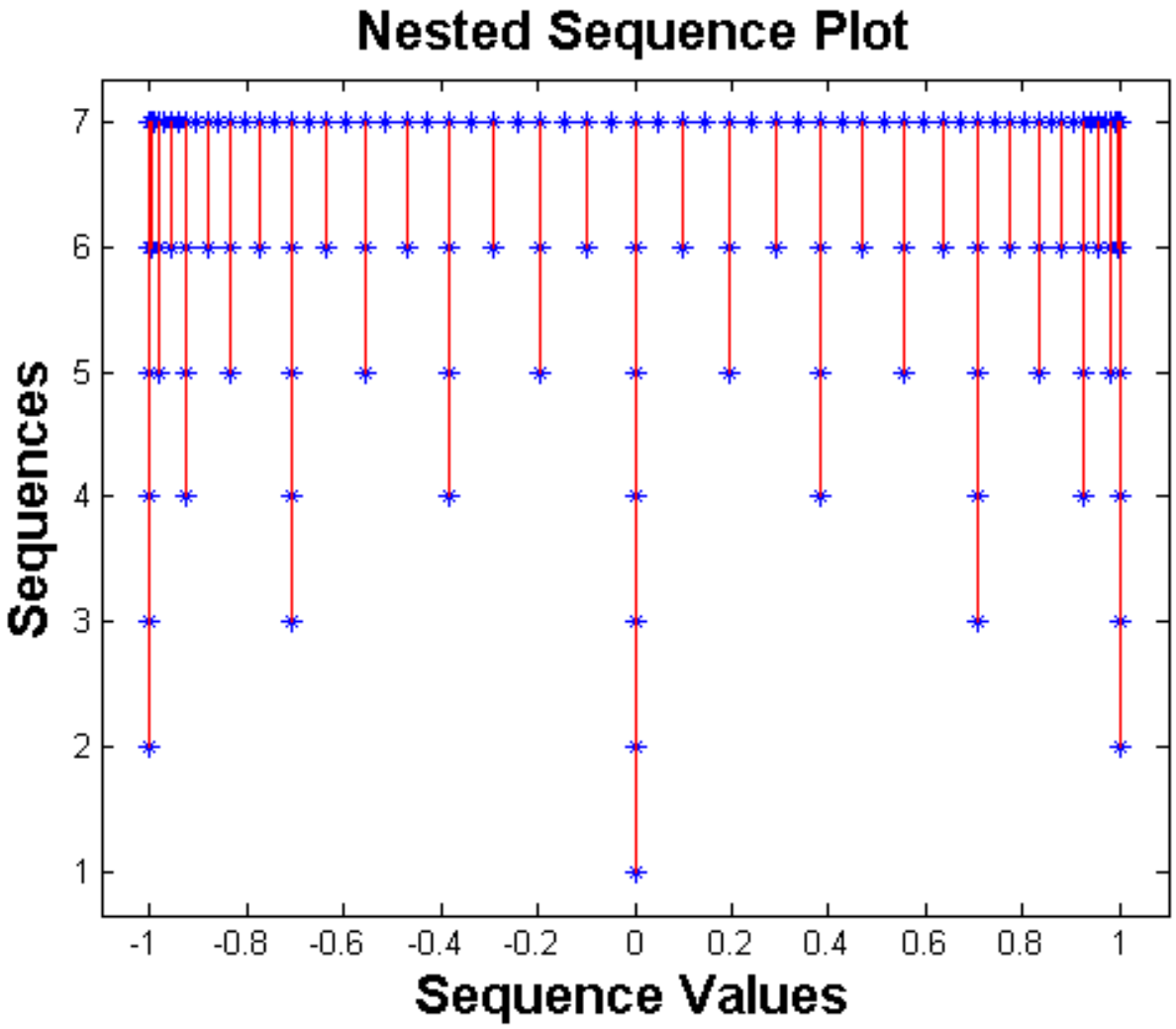
Interpolatory Quadrature: Features

- uses a regular grid of N points;
- Evaluates each $f(x)$;
- Computes a *weighted* average of the function values.

- To reuse data, the grids must be "nested".
- **The error can drop with an exponent of N**

Interpolatory Quadrature: Nested Rules

Our nested rules roughly double the number of points at each step.



Interpolatory Quadrature: Monomial Accuracy

Interpolatory quadrature works well if $f(x)$ can be well approximated by a sum of monomials.

A 1D rule has accuracy N if it "captures" all monomials from $1, x, x^2$, up to x^N.

The lowest monomial we can't capture determines the error. A rule of accuracy N can't capture x^N and so its error behaves like h^{N+1}.

These monomials are the creatures we are "stalking".

Product Rules: Begin

1. Integration
2. Quadrature
3. Interpolatory Quadrature
4. **PRODUCT RULES**
5. Smolyak Quadrature
6. Sparse Grid Software
7. Conclusion

Product Rules

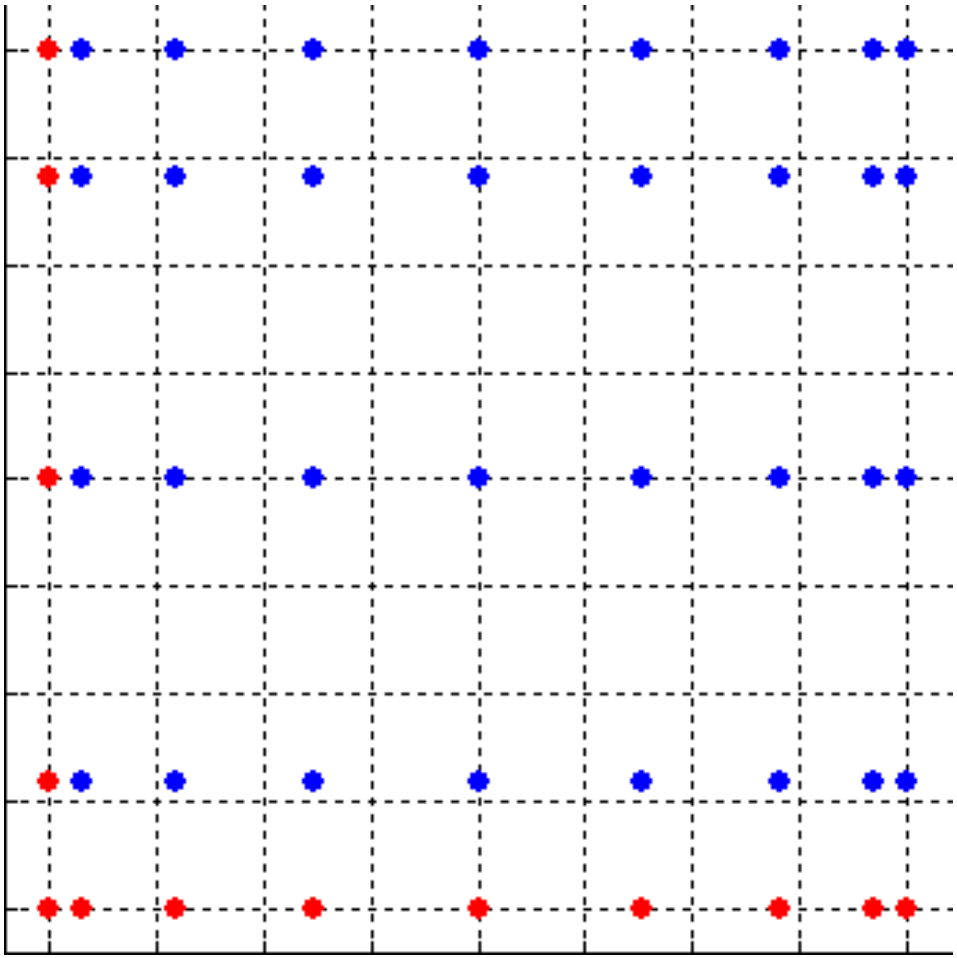
A 2D product rule can be made by taking two 1D rules and combining pairs of values.

The number of points in a product grid is the product of the sizes of the 1D rules.

The resulting rule captures monomials up to $x^{N1}y^{N2}$ where $N1$ and $N2$ are the individual accuracies.

Product Rules: a 9x5 rule

A product of 9 point and 5 point rules.



Product Rules: Point Growth with Dimension

Suppose we take products of a modest 4 point rule:

- 1D: 4 points;
- 2D: 16 points;
- 3D: 64 points;
- 4D: 256 points;
- 5D: 1024 points;
- 10D: a million points;
- 20D: a trillion points.
- 100D: don't ask!

Conclusion: *Product rules can't go very far!*

Product Rules: Component Degree and Total Degree

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".

Product Rules: Monomials of Degree 4

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.

Product Rules

As the dimension increases, the useless monomials predominate.

Suppose we take products of a modest rule of accuracy 10, and limit the exponent total to 10. How many "good" and "useless" monomials do we capture?

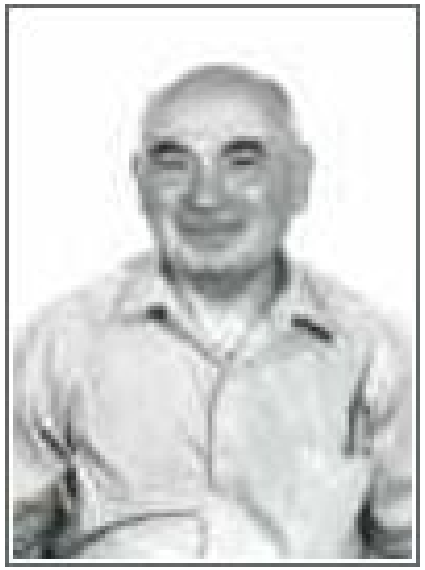
Dim	Good	Useless
1D	10	0
2D	55	45
3D	120	880
4D	210	9790
5D	252	99748

Conclusion: *A "cut down" product rule might work!*

Smolyak Quadrature: Begin

1. Integration
2. Quadrature
3. Interpolatory Quadrature
4. Product Rules
5. **SMOLYAK QUADRATURE**
6. Sparse Grid Software
7. Conclusion

Smolyak Quadrature



Sergey Smolyak (1963) added low order grids together.

Each of his combined "sparse grids":

- had the same asymptotic accuracy as a product grid.
- was a subset of the points of the product grid.
- used *far fewer* points.

Smolyak Quadrature

Number of points in a 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

These point counts and accuracy assume a 1D Clenshaw-Curtis grid.

Smolyak Quadrature

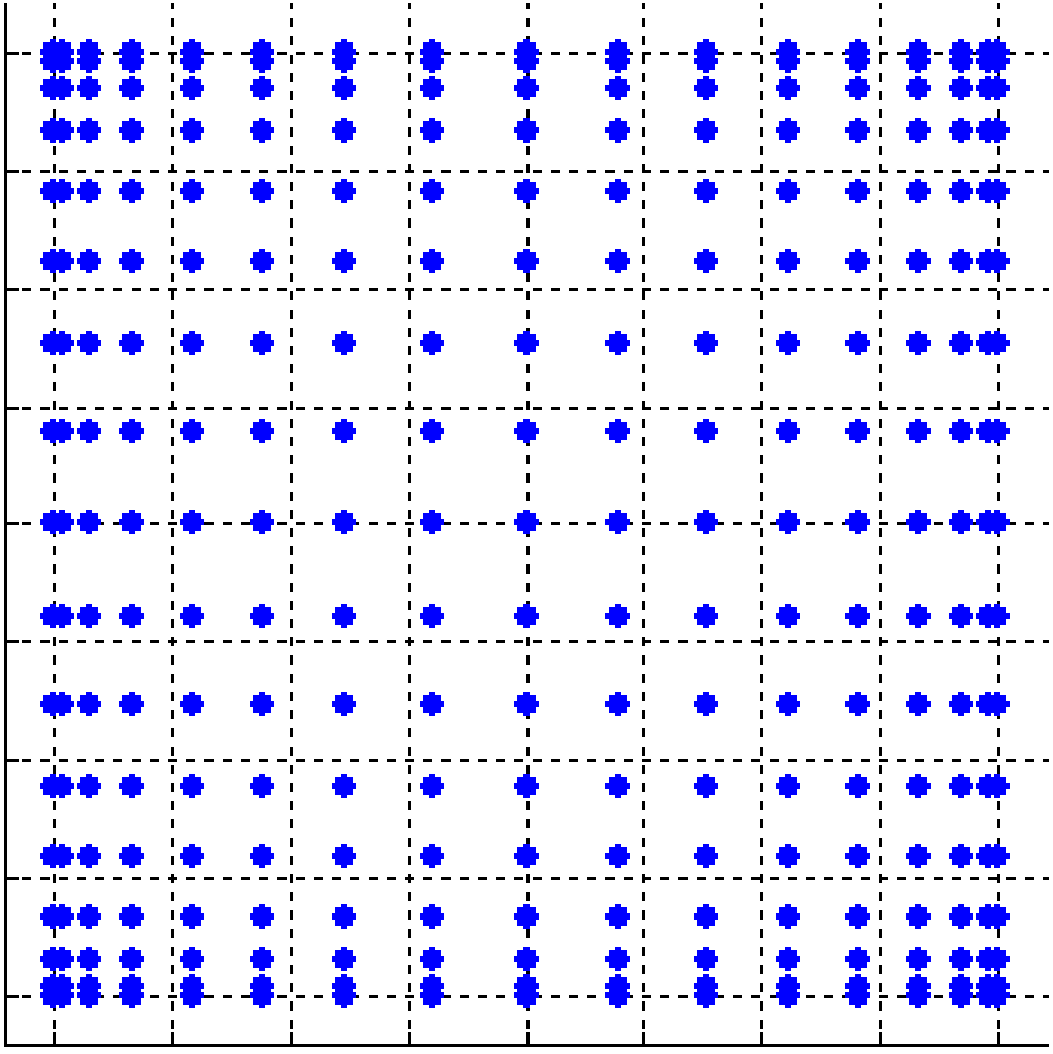
Level and 1D accuracy (= point count) versus Multidimensional accuracy.
(We also include 10D point count.)

Level	1D count	10D count	Accuracy
0	1	1	0,1
1	3	21	0,1,2,3
2	5	221	0,1,2,3,4,5
3	9	1581	0,1,2,3,4,5,6,7
4	17	8801	0,1,2,3,4,5,6,7,8,9
5	33	41265	0,1,2,3,4,5,6,7,8,9,10,11
6	65	171425	0,1,2,3,4,5,6,7,8,9,10,11,12,13

Multidimensional accuracy = 2 * LEVEL + 1.

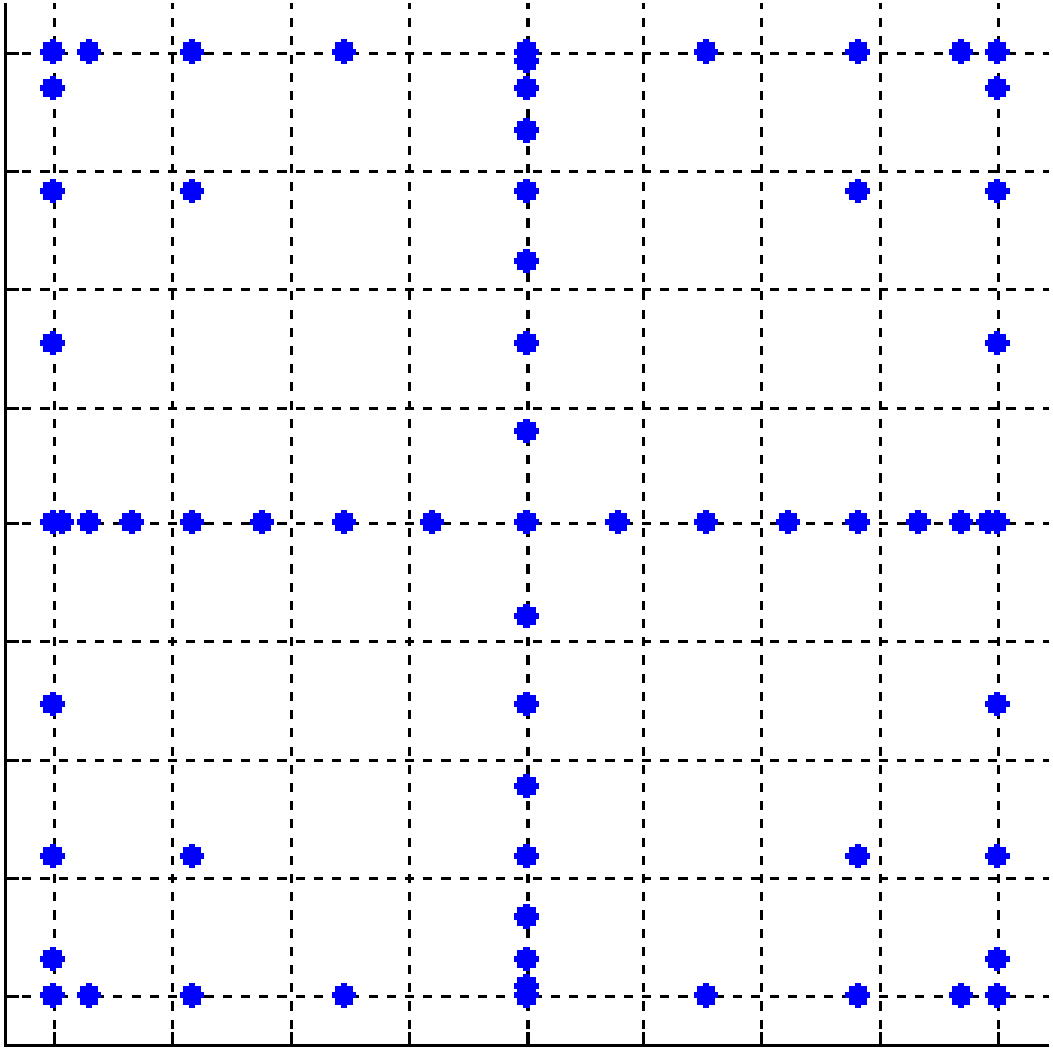
Smolyak Quadrature: 2D Order 17 Product Rule

A 17x17 product grid (289 points).



Smolyak Quadrature: 2D Level4 Smolyak Grid

An "equivalent" sparse grid (65 points).



Smolyak Quadrature

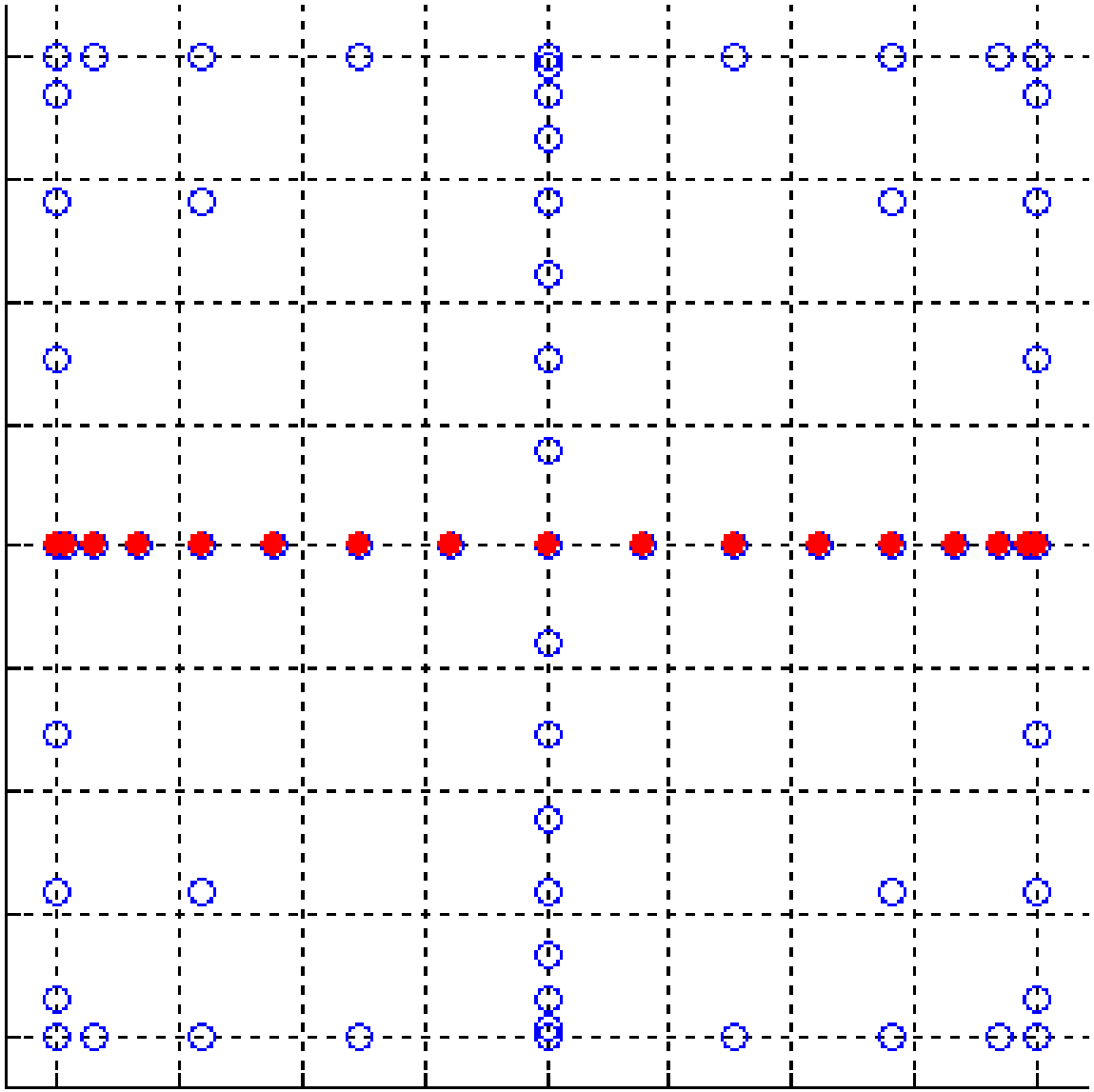
To capture only "desirable" monomials, we essentially add product grids which are sparse in one direction if dense in the other.

Because of nesting, the grids reuse many points.

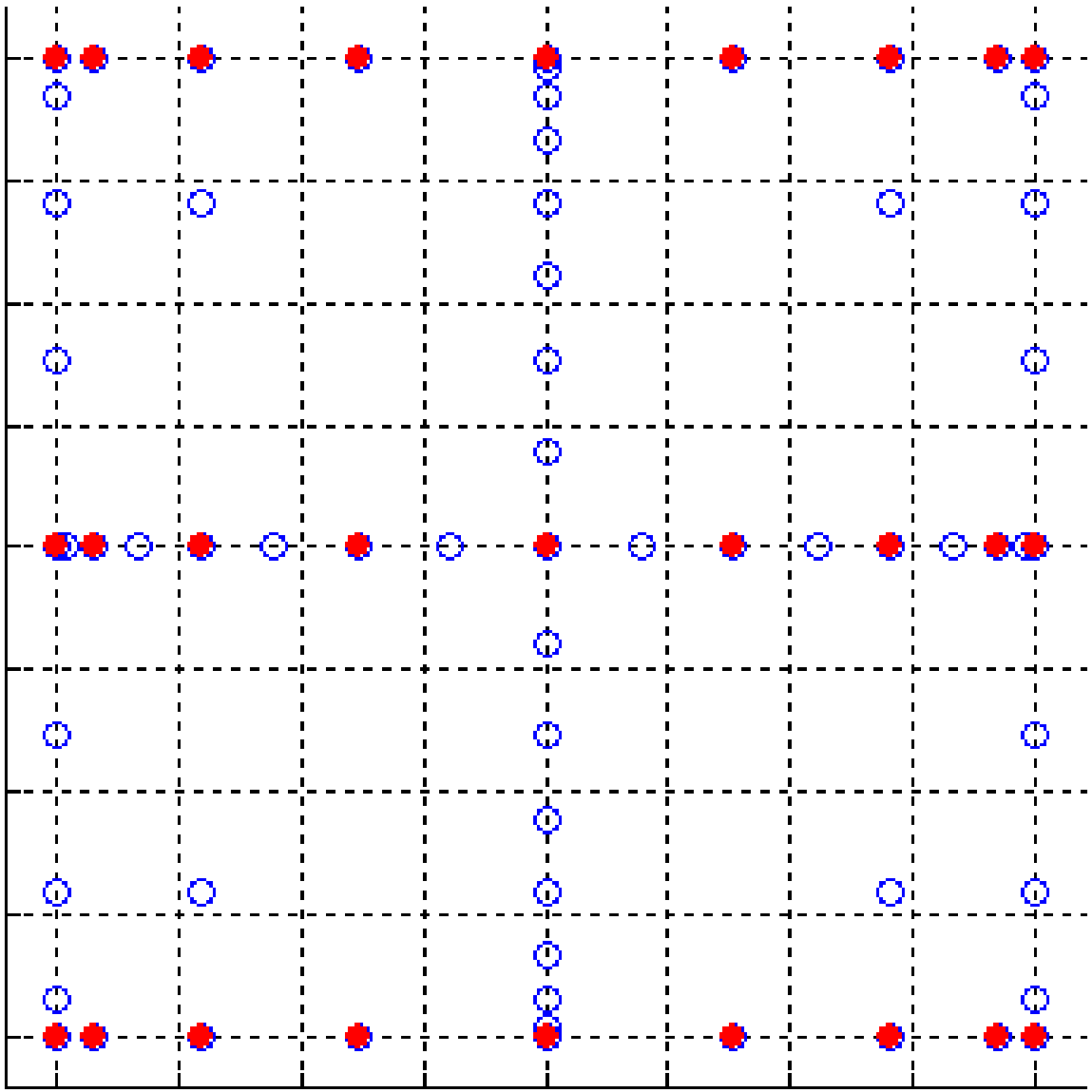
The big savings comes from entirely eliminating most of the points of the full product grid.

The improvement is greater as the dimension or level increases.

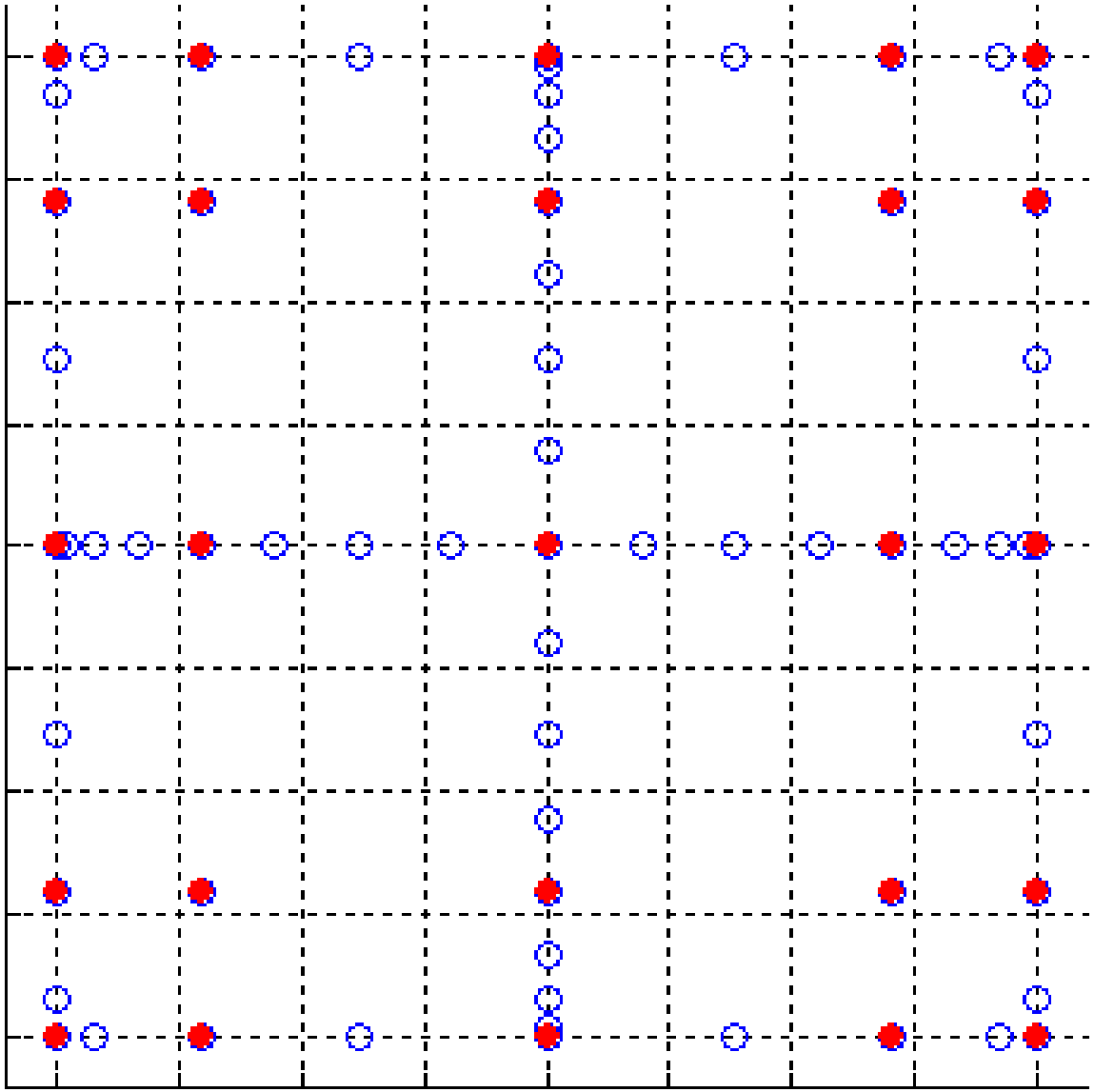
Smolyak Quadrature: 2D Level4 17x1 component



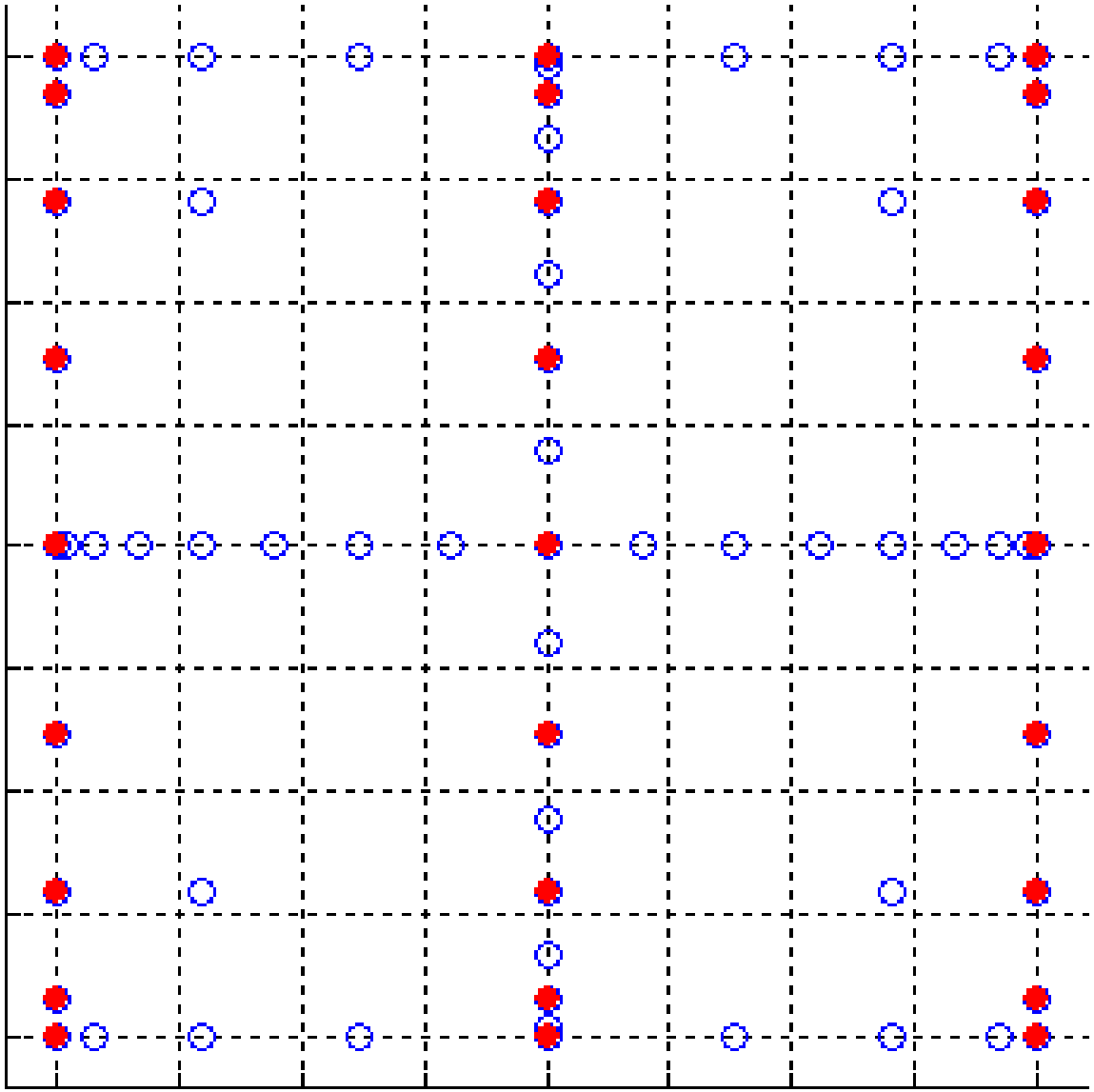
Smolyak Quadrature: 2D Level4 9x3 component



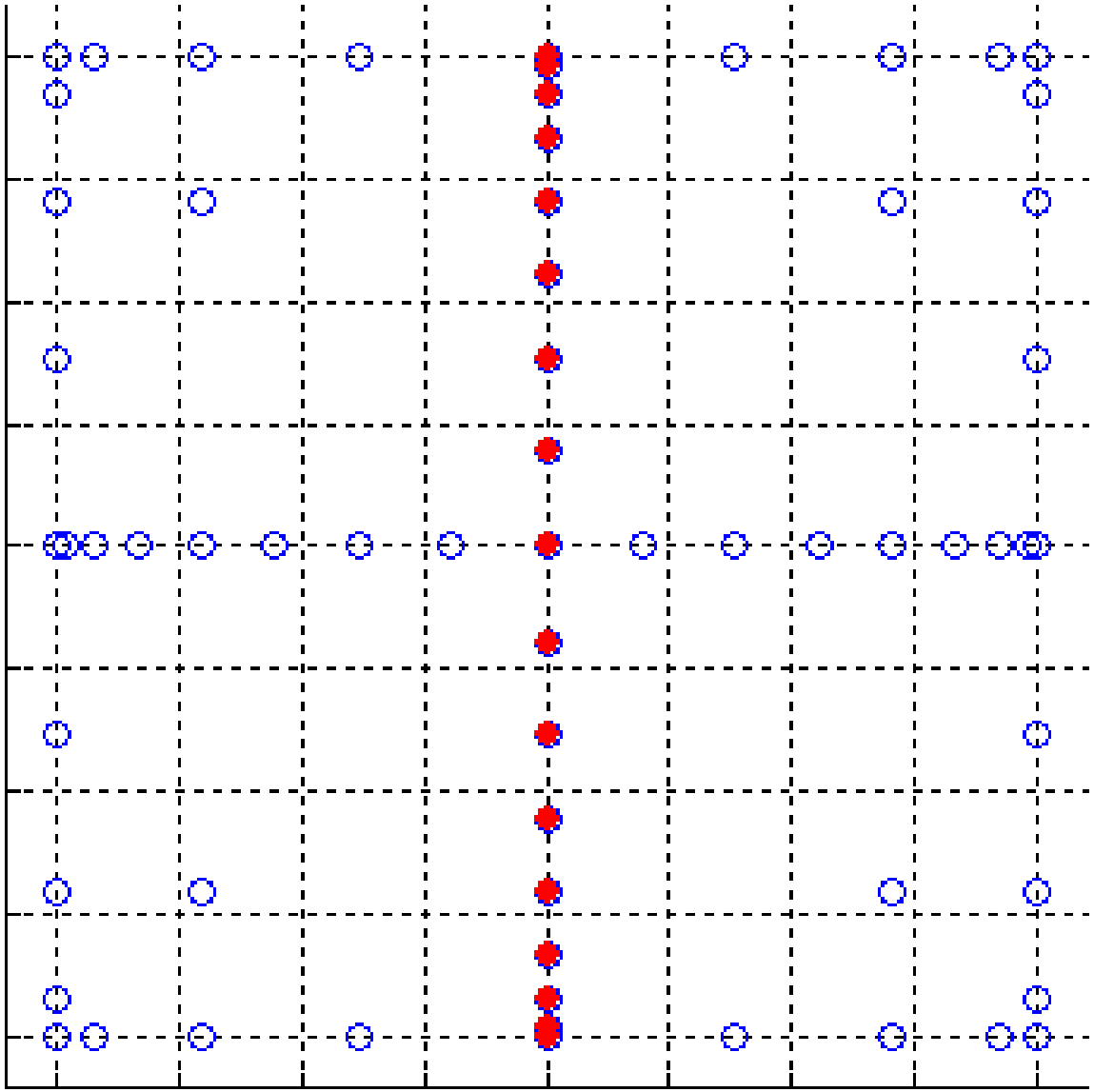
Smolyak Quadrature: 2D Level4 5x5 component



Smolyak Quadrature: 2D Level4 3x9 component

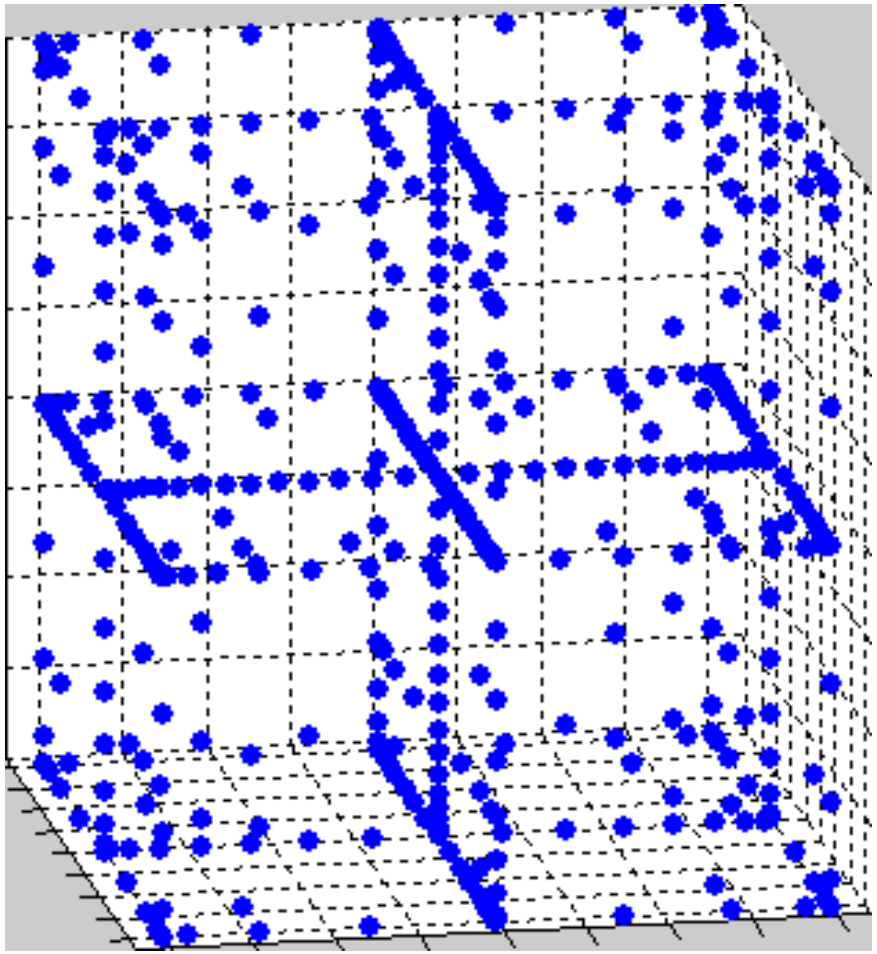


Smolyak Quadrature: 2D Level4 1x17 component

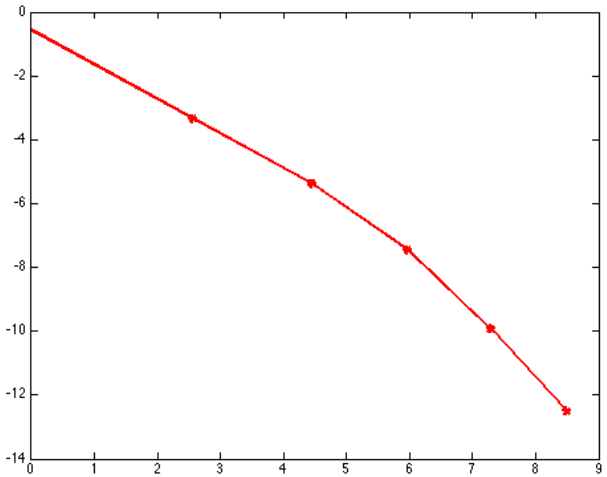


Smolyak Quadrature: 3D Level5 Smolyak Grid

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

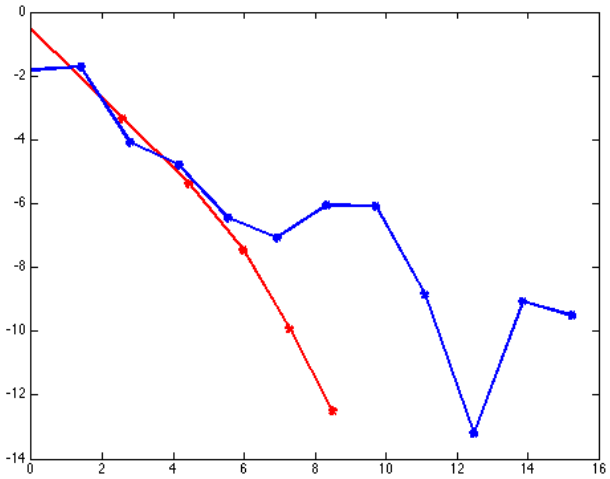


Smolyak Quadrature: 6D Sparse Grid Quadrature Error



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.

Routines for DAKOTA: Begin

1. Integration
2. Quadrature
3. Interpolatory Quadrature
4. Product Rules
5. Smolyak Quadrature
6. **SPARSE GRID SOFTWARE**
7. Conclusion

Sparse Grid Software

I have been working on several software libraries for sparse grid integration. The libraries are available in C++, FORTRAN90 and MATLAB. They include functions which:

- given dimension D and level L, determine the number of points N.
- allocate space for abscissas X and weights W.
- determine the abscissas and weights.
- approximate the integral as a weighted sum of function values.
- provide test integrand functions, to estimate the accuracy of the rules.

Step 1 requires the function "sparse_grid_cc_size" and step 3 is carried out by "sparse_grid_cc".

Sparse Grid Software: Rule Generation

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

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

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

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

Sparse Grid Software: Accuracy Check

If the rule of level **L** is implemented correctly, it should exactly integrate every monomial up to total degree $2 * L + 1$.

The routine "monomial_quadrature" can check this.

```
int E = { 3, 1, 2 };

error = monomial_quadrature ( D, E, N, W, X );
```

Sparse Grid Software: Accuracy Result

Error	Total Degree	Monomial Exponents
0.000000	0	0, 0
0.000000	1	1, 0
0.000000	1	0, 1
0.250000	2	2, 0
0.000000	2	1, 1
0.250000	2	0, 2
0.500000	3	3, 0
0.250000	3	2, 1
0.250000	3	1, 2
0.500000	3	0, 3

Conclusion: Begin

1. Integration
2. Quadrature
3. Interpolatory Quadrature
4. Product Rules
5. Smolyak Quadrature
6. Sparse Grid Software
7. **CONCLUSION**

Conclusion: Future Work

- Precompute quadrature rules, for parallel application
- Composite version for decomposition of a few dimensions.
- Allow some dimensions to be approximated more carefully.
- Detect anisotropy in the data.
- Estimate quadrature error.
- Investigate rules for $[0, \infty)$ and $(-\infty, \infty)$.

Conclusion: The End

- High dimensional integration is a feature of modern algorithms
- Accurate Monte Carlo results take a long time
- Product rules quickly become useless
- "Smooth" data can be well integrated by Smolyak grids
- Abstract probability spaces may generate suitably smooth data

Conclusion: Software

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**.

SPARSE_GRID_CC a directory on my website containing examples of sparse grids generated from Clenshaw Curtis rules.

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.

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.