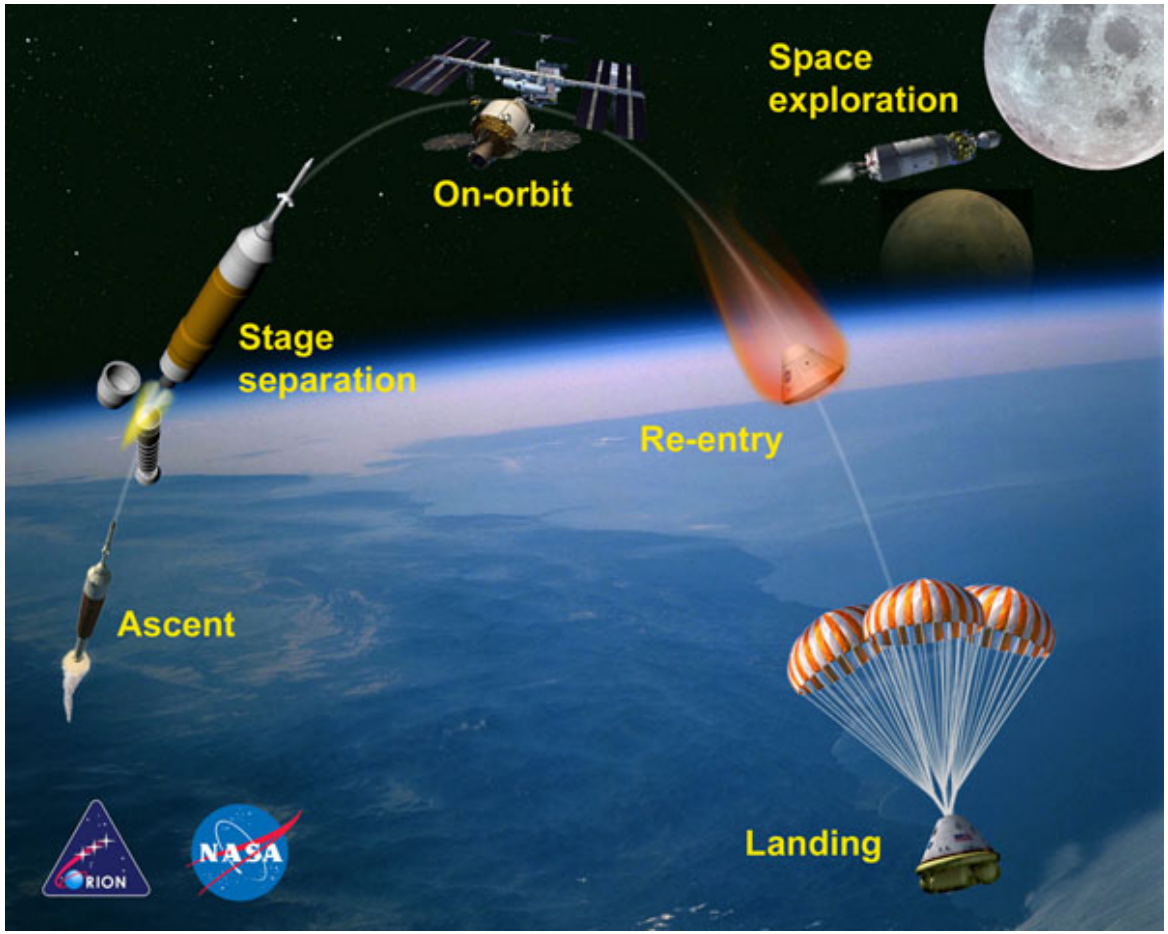
Sparse Grid Collocation for Uncertainty Quantification

John Burkardt
Department of Scientific Computing
Florida State University
http://people.sc.fsu.edu/~jburkardt/presentations/...
...sparse_2012_scala.pdf

21 January 2012

- **Introduction**
- Integrals Arising from Stochastic Problems
- Smolyak's Sparse Grid Definition
- Aspects of Sparse Grids
- Numerical Examples
- Conclusion

INTRO: Orion Space Vehicle Heat Shield



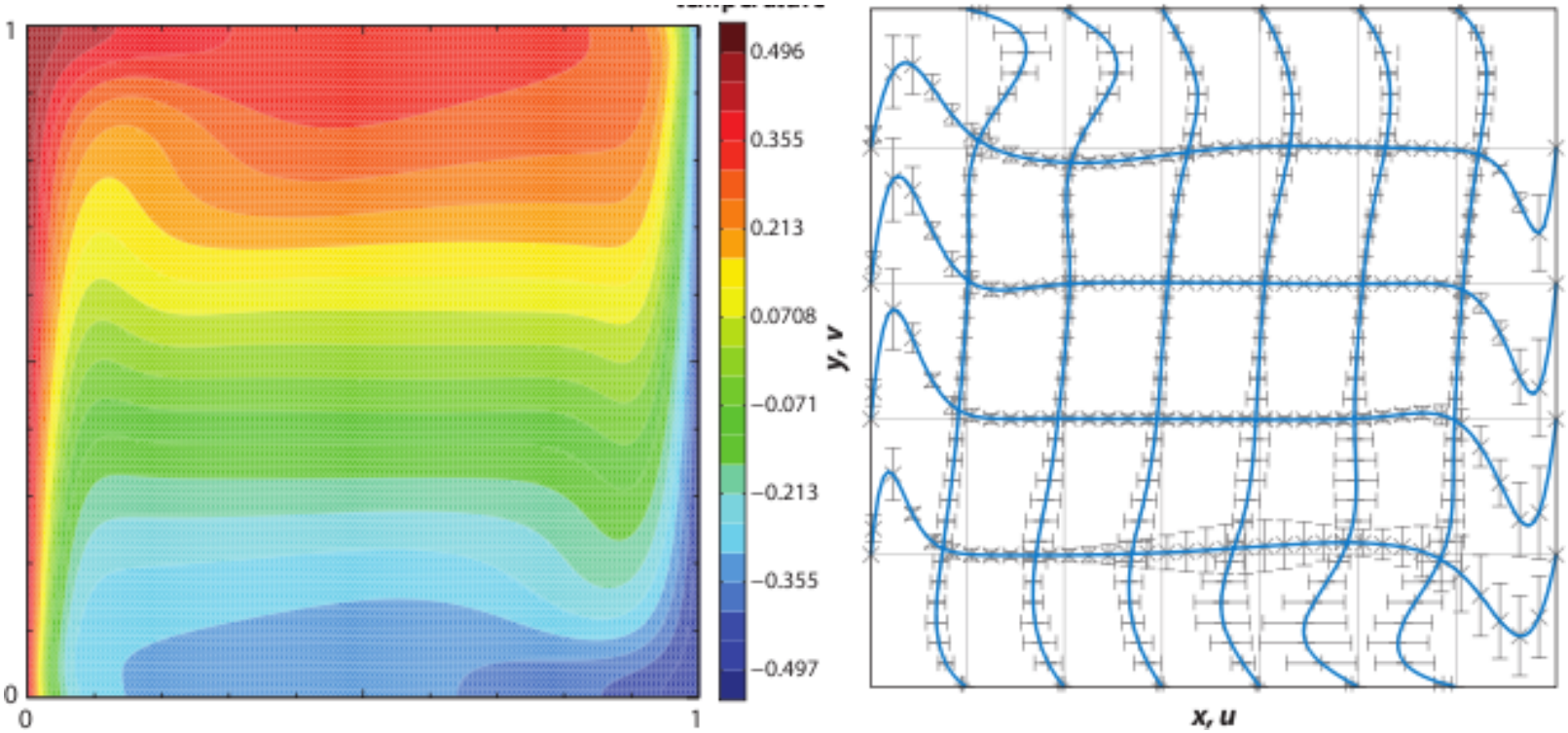
Kleb, Johnston,
Uncertainty Analysis of Air Radiation for Lunar Return Shock Layers, AIAA 2008-6388.

INTRO: Wing Load Test

Babuska, Nobile, Tempone,
Reliability of Computational Science, Numerical Methods for PDE's, Volume 23, Number 4, April 2007.

INTRO: Boussinesq Flow

Boussinesq flow, right hand side temperature is uncertain.



Temperature (left) and velocity (right) with uncertainty bars.

Le Maitre, Reagan, Najm, Ghanem, Knio,
A stochastic projection method for fluid flow II,
Journal of Computational Physics, Volume 181, pages 9-44, 2002.

INTRO: An "Exact Answer" May Be Wrong

The preceding images suggest some of the situations in which a calculation must be carried out in the presence of uncertainty, error, or variation.

One can ignore the uncertainty, assume the data is exact, and compute an "exact answer".

One understands that this "exact answer" will not correspond to the physical situation actually encountered, in part because of the uncertainties we ignored; if we expect the actual behavior to be close to our calculation, that is simply a matter of hope.

However, a variety of techniques are available for making a reasonable model of uncertain parameters, and computing the expected value of the solution, and the magnitude of the likely variations.

INTRO: The Value of an Ace

Since the audience represents a wide range of backgrounds and interests, I'd like to ensure that everyone understands the problem I am considering and the approach I will take.

Suppose I have picked four cards from a deck of 52 cards, and am about to draw my last card. I am hoping to get a pair, three of a kind, full house, four of a kind, or a straight; each of these hands is worth a certain prize, and I'd like to estimate my winnings, even though I am uncertain about the card I will pick.

I can model the uncertainty in my pick by assigning each rank from ace to king a probability based on the number of cards of that rank remaining in the deck, divided by 48. The value of each rank is the value associated with whatever hand I form using that card. The expected value sums each rank's probability times its value.

INTRO: The Value of a Heart

If I take the suit of the cards into account, so that I can get a reward for a flush, for instance, then I now have to consider a larger probability space, that is, one in which I consider all 13 ranks (ace through king) and all 4 suits ♡, ♣, ♢, ♠, making 4*13=52 events I have to account for.

Again, I can assign a value to each event, namely, the money I will win if I choose that card. To estimate the value of the unknown card I am about to pick, I sum the products of every suited rank multiplied by its probability, which in this case will be 1/48 for each card, except for the four I already have.

INTRO: Estimating Values in Real Life

In real life, instead of a short list of card ranks, I consider a quantity such as the size of a particle of "space junk" that might hit the Orion Space Vehicle during reentry, which can have a continuous range of values. By studying records, I can form a probability density function or "pdf" for this quantity.

The expected particle size is computed by multiplying each size by its pdf, and integrating, similar to the sum we did for the cards.

If there is a second factor, such as the relative speed of the particle, I can form a separate density function for that quantity.

To consider collision events in which both size and speed of the particle vary, I consider a two dimensional product space of possibilities, just as 4 suits and 13 ranks became 52 combined possibilities in the card example.

INTRO: Approximating High Dimensional Integrals

Each unknown factor you consider adds a new abstract "dimension". Summarizing all possible events requires integrating over this multidimensional product region.

Thus, analyzing uncertainty often becomes evaluating a high-dimensional integral.

Since we usually can't use formulas in such cases, we apply well known methods for estimating integrals. One method uses random sampling to get a quick estimate; the other checks some carefully chosen values to get a more accurate estimate.

If we go to high dimensions (even 10 dimensions can be "high"), standard methods don't produce accurate answers quickly.

The rest of this talk will consider an improved method of estimating integrals with an uncertain or stochastic influence.

- Introduction
- **Integrals Arising from Stochastic Problems**
- Smolyak's Sparse Grid Definition
- Aspects of Sparse Grids
- Numerical Examples
- Conclusion

STOCHASTIC: Stochastic Problems

Stochastic problems are mathematical models of systems in which some data is uncertain, approximate, or nondeterministic.

In a mathematical model, even uncertainty is described precisely.

In a probabilistic model, we consider an ensemble of possible "realities", and suppose we can assign a probability to each.

Such a model allows us to simulate an uncertain process, and to carry out analysis, such as computing expected values by multiplying every outcome by its probability and integrating.

STOCHASTIC: The Poisson Equation

Let us consider the Poisson diffusion equation:

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

$\mathbf{a}(\vec{x})$ is the diffusivity, $\mathbf{f}(\vec{x})$ a source term.

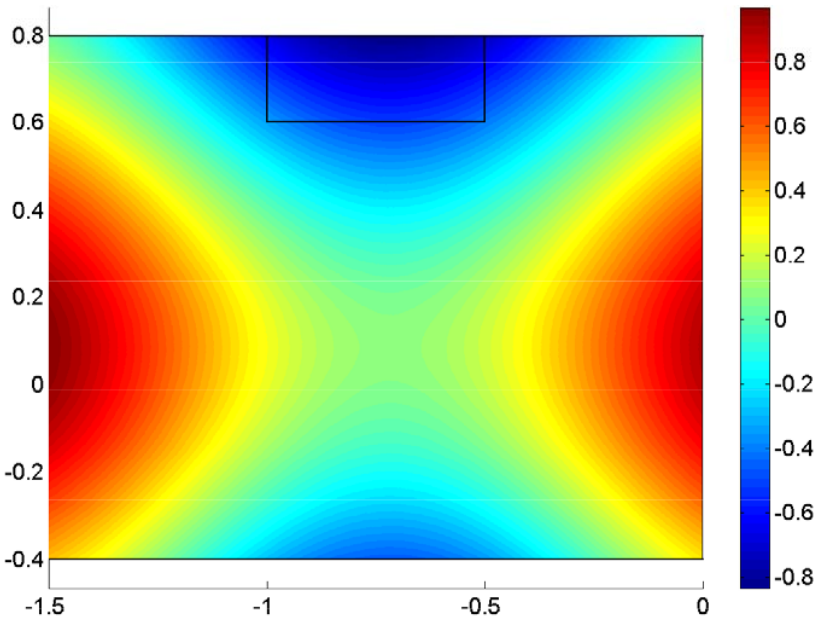
A stochastic version of the Poisson equation might allow uncertainties in the diffusivity:

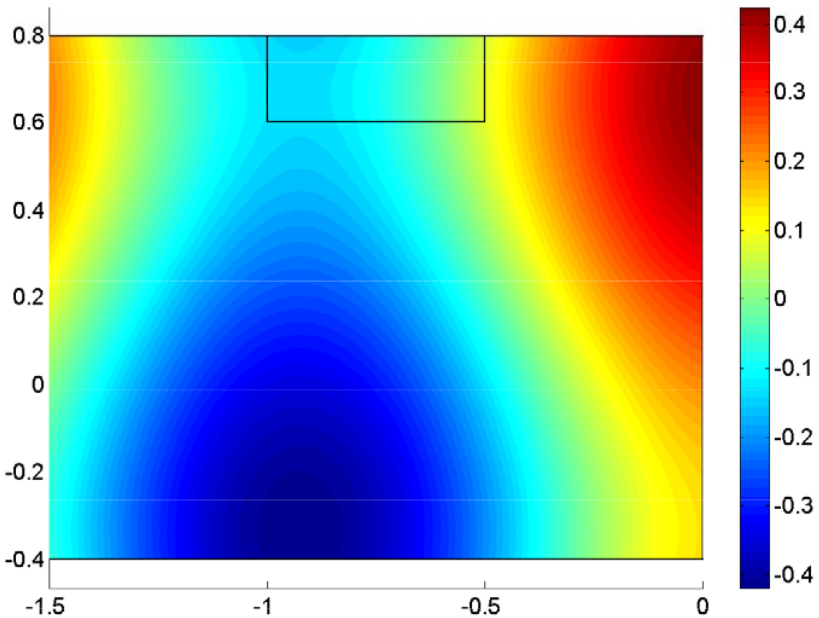
$$-\nabla \cdot (a(\vec{x};\omega)\nabla u(\vec{x};\omega)) = f(\vec{x})$$

Here, ω represents the stochastic influence, and we must now write $\mathbf{u}$ with an implicit dependence on ω as well.

STOCHASTIC: Stochastic Diffusivity

Diffusion $-\nabla \cdot (a \nabla u) = 0$ with stochastic diffusivity $a(x, y; \omega)$.





Log of realizations of the diffusivity $a(x, y; \omega)$ using uniform random variables ω (left) or Gaussian ω (right).

STOCHASTIC: Computing the Expected Value

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.

Moreover, if, with each value of ω we associate a probability density function $\rho(\omega)$, we can reformulate our problem to solve for the finite element coefficients of the expected value function $\bar{\mathbf{u}}(\vec{x})$.

The validity and usefulness of all our results depends on how carefully we choose a probabilistic model to combine with our deterministic system.

STOCHASTIC: The Stochastic Finite Element System

A finite element approach to the deterministic problem integrates the equation against various test functions $\mathbf{v_i}(\vec{x})$:

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

We can still apply the finite element approach to our stochastic problem, but now we integrate over the probability space as well:

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

This is an algebraic system $A * c = f$ for the finite element coefficients of $\bar{\mathbf{u}}(\vec{x})$, but evaluating $\mathbf{A}$ and $\mathbf{f}$ requires approximate integration over $\mathbf{D}$ AND over the probability space $\mathbf{\Omega}$.

STOCHASTIC: Focus on the Integration Problem

Solving the stochastic problem begins by approximating the probability space by some discretized M-dimensional subspace Ω^M. We are then faced with the problem of estimating these integrals.

Clearly, we can do a better job of estimating the influence of the stochastic parameters if we can make M large; on the other hand, the task of estimating integrals in M-dimensional space can quickly become intractable as the dimension increases.

We'll now concentrate on the problem of integral approximation, and thus replace our complicated stochastic diffusion integral by the model task of estimating integrals in an M-dimensional product region:

$$\int_{\Omega^M} f(x)\, dx$$

STOCHASTIC: The Monte Carlo Approach

One choice is the **Monte Carlo** method, which randomly samples the integration parameter, either uniformly or according to a weight function.

The method does not require the integrand function to be smooth, we can choose any number of samples we want, and if we decide to request more samples, these can be combined with the existing ones.

Statistically, the error in the Monte Carlo integral estimates will tend to decrease like $N^{-1/2}$. It's good that this number does not depend on the dimension M. But it is nonetheless a remarkably slow rate. Roughly speaking, each new decimal of accuracy requires 100 times as much work.

Since the error goes down relatively slowly, if our initial estimates are poor, it may be a very long time before we achieve a good estimate.

STOCHASTIC: The Interpolatory Approach

Interpolatory methods sample the integration parameter at selected points, constructing and integrating the interpolating function. If the function is smooth, convergence will be fast, and actually precise if the integrand is a low degree polynomial.

Gauss methods are similar to the interpolatory approach, requiring very specific sampling locations in exchange for squaring the convergence exponent.

For the interpolatory and Gauss rules, the natural approach is to form a **product rule**, using the Cartesian product of the 1D rule taken M times. If we approach this problem by asking for a specific precision, then we will specify some number of points P for the 1D rule. This implies that our product grid will need $N = P^M$ points.

Even for low precision requests P, we can't go far in dimension M before a product rule becomes unaffordable.

Sparse Grid Collocation

- Introduction
- Integrals Arising from Stochastic Problems
- **Smolyak's Sparse Grid Definition**
- Aspects of Sparse Grids
- Numerical Examples
- Conclusion

SMOLYAK: Interpolatory Quadrature Precision and Cost

The guaranteed precision of the product rule comes at a cost which explodes as the dimension M increases.

Each successive entry in a family of quadrature rules integrates precisely more successive monomials of the series *constants*, *linears*, *quadratics*, *cubics*, *quartics*, ...

For a given dimension M, if we increase the precision request P, how many monomials must we be able to precisely integrate?

We assume that a quadrature rule can "buy" a monomial at the cost of a function evaluation.

SMOLYAK: What Monomials Must We Capture?

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

An M-dimensional product of a 1D P-point rule requires P^M function evaluations. This means the M dimensional product rule can integrate precisely far more monomials than it needs to.

Keep in mind that P is likely to stay relatively small, but M may go to 20, 30, 50 or 100.

As M increases, the discrepancy grows. As an extreme example, for $M = 100$, there are just 101 monomials of degree 1 or less, and 5,151 monomials of degree 2 or less, but a product rule of 2 points would request 2^{100} function evaluations.

SMOLYAK: Beat the Product Rule

The Smolyak approach arises as follows. Let Q be a family of 1D quadrature rules, so that $Q(P)$ can integrate polynomials of degree P exactly. (We'll shortly want to assume that the points used by rules $Q(0)$, $Q(1)$, $Q(2)$ are nested.)

Write any 2D product rule $Q(P_1, P2) = Q(P_1) \otimes Q(P_2)$

To approximate an integral in 2D up to and including linear terms, we might use the product rule $Q(1,1) = Q(1) \otimes Q(1)$, using 4 points, and precisely integrating any terms involving 1, x, y, or xy.

Smolyak observed that we can get exactly the constants and linears, using just 3 points, using a rule that looks like this:

$$\mathcal{A}(\mathrm{L{=}1,M{=}2}) = Q(1,0) + Q(0,1) - Q(0,0)$$

(The L indicates the level, and M the spatial dimension.)

SMOLYAK: Rewrite to Form a New Quadrature Rule

We're assuming the rules are nested. So we can represent the rules as follows:

```
Q(0,0) = a * f(0,0)
Q(1,0) = b * f(0,0) + c * f(x,0)
Q(0,1) = d * f(0,0) + e * f(0,y)
```

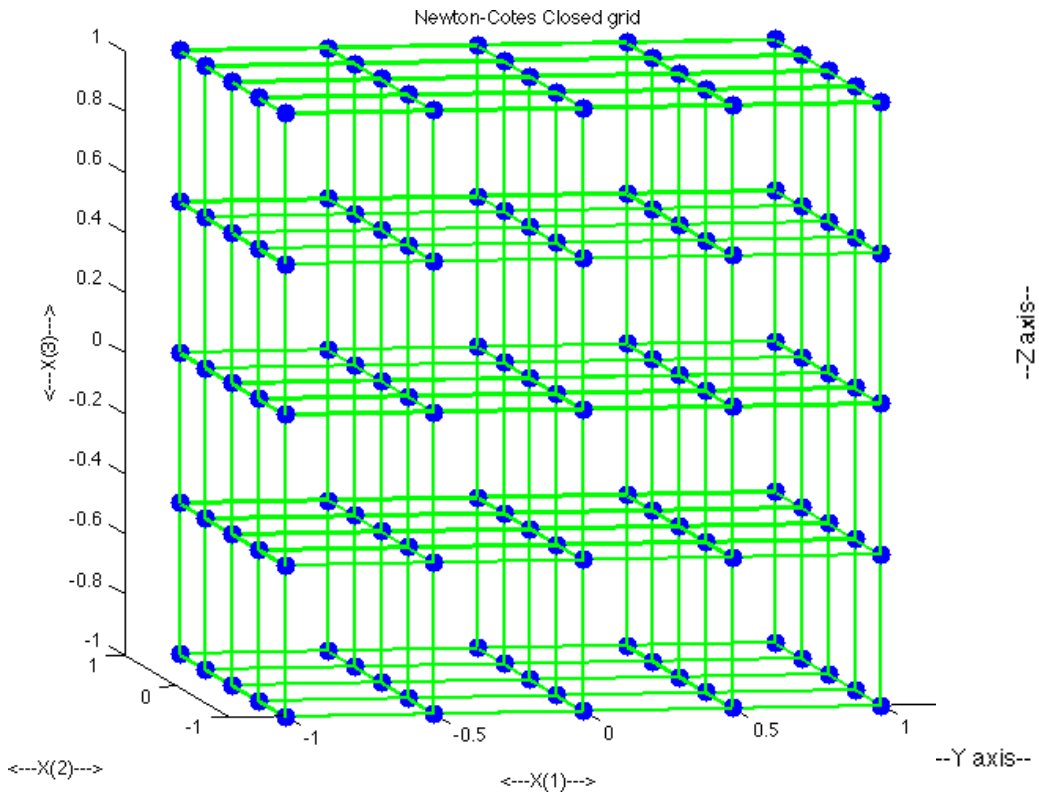
This requires 5 function evaluations before we can combine the three results. Instead, let's combine the coefficients of each function value:

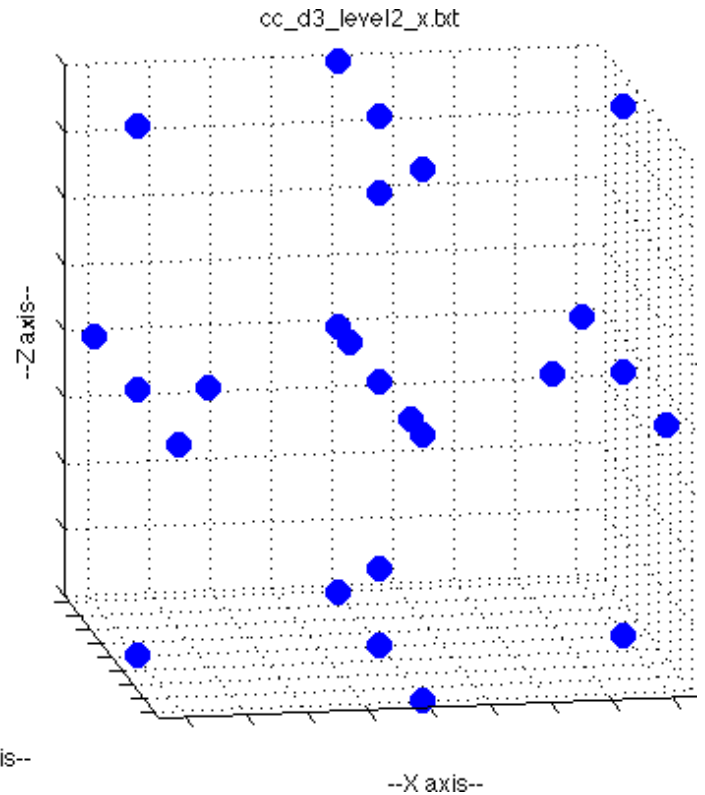
$$\mathcal{A}(\text{L=1,M=2}) = (b + d - a) * f(0,0) + c * f(x,0) + e * f(0,y)$$

Now we've guaranteed we only need 3 function evaluations. And we have also written the Smolyak rule as a quadrature rule, that is, coefficients times function values.

SMOLYAK: 5x5x5 Product Versus Smolyak L=2, M=3

125 points in the product rule, 25 points in the Smolyak rule.





SMOLYAK: Generalizing the Idea

Smolyak determined a formula combining low-order product rules to achieve a desired precision, avoiding the excessive function evaluation of the standard product rule.

The details of precision, order, efficiency and accuracy vary depending on the underlying 1D quadrature rules.

In most cases, Smolyak matches the product rule while avoiding the crushing explosion in function evaluations.

It's useful now to quote the formal definition of the Smolyak procedure, so you can see that it is framed in an abstract way that allows a lot of flexibility in how it is applied.

SMOLYAK: Formal Definition

We have a family of 1D quadrature rules $\mathcal{Q}^\ell$ indexed by ℓ.

We form a sparse grid $\mathcal{A}(L, M)$ for dimension **M** and level **L**.

$\mathcal{A}(L, M)$ is a weighted sum of M-dimensional product rules

$$\mathcal{Q}^{\ell_1} \otimes \cdots \otimes \mathcal{Q}^{\ell_M}$$

The vector $\vec{\ell}$ lists the levels of the component rules used, and $|\vec{\ell}| = \ell_1 + \cdots + \ell_M$ is the sum.

$$\mathcal{A}(L, M) = \sum_{L-M+1 \le |\vec{\ell}| \le L} (-1)^{L-|\vec{\ell}|} \left(\begin{array}{c} M-1 \\ L-|\vec{\ell}| \end{array} \right) (\mathcal{Q}^{\ell_1} \otimes \cdots \otimes \mathcal{Q}^{\ell_M})$$

Thus, the rule $\mathcal{A}(L, M)$ is a weighted sum of M-dimensional product rules whose total level $|\vec{\ell}|$ never exceeds L.

SMOLYAK: Efficiency

We said that 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; we do not see an exponent of M in the point count.

For the extreme case of $M = 100$, a sparse grid based on the Clenshaw Curtis rule can integrate polynomials of total degree 0, 1, 2 or 3 using just 201 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**).

SMOLYAK: Relation with Stochastic Variables

In stochastic problems, the type of random variable used to model uncertainty determines the quadrature rule that should be used to handle that variable. Some common choices include:

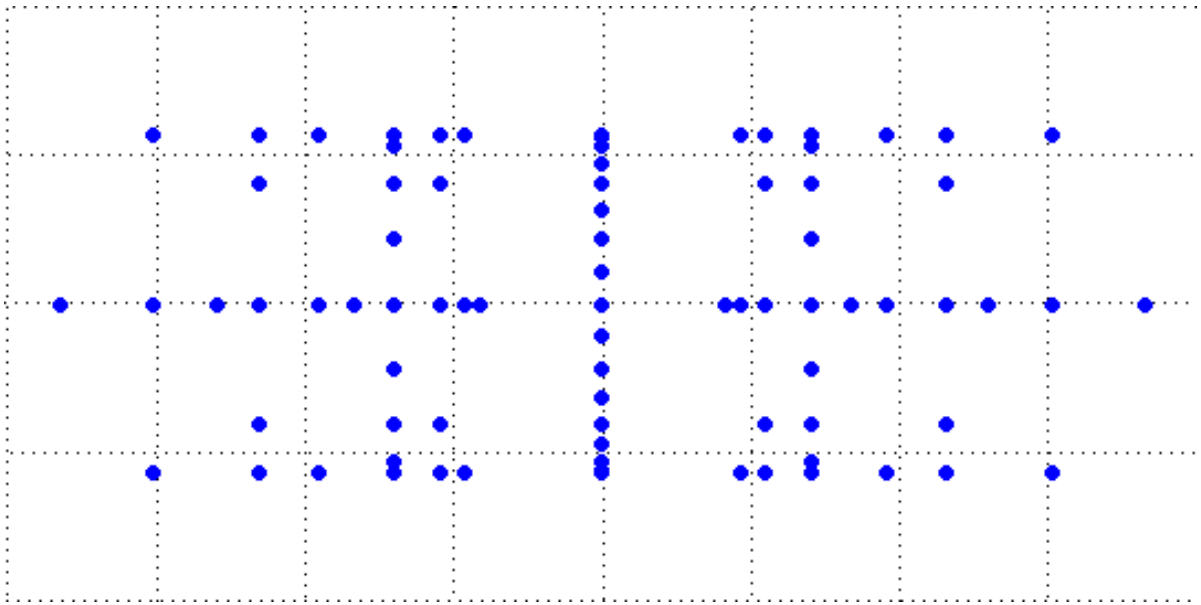
Distribution	Domain	Weight	Quadrature
Uniform	$[-1,+1]$	1	Gauss-Legendre or Clenshaw-Curtis
Gaussian	$(-\infty,+\infty)$	$e^{-(x-\alpha)^2/\beta^2}$	Gauss-Hermite
Gamma	$[0,+\infty)$	$e^{-\alpha x}$	Gauss-Laguerre
Beta	$[-1,+1]$	$(1-x)^{\alpha}(1+x)^{\beta}$	Gauss-Jacobi

Any of these rules can be used in the sparse grid formulation.

SMOLYAK: Mixed Products

The Smolyak procedure allows each dimension to be treated independently in whatever manner is appropriate for that variable.

The sparse grid technique simply combines the given rules, without requiring that they involve the same 1D domain or weight function.



Level 4 Rule, Hermite in X, Clenshaw Curtis in Y

SMOLYAK: Precision, Point Growth, Error Bounds

Refer to **Novak and Ritter*** for details about the construction of rules based on the Clenshaw Curtis rule, error bounds for the approximation of integrals of smooth functions, estimates for the growth in the number of function evaluations with increasing level or dimension, and a proof that the rules are exact for all monomials up to a given degree.

The authors also indicate conditions under which other 1D quadrature rules can be used to construct Smolyak rules of comparable precision.

**High dimensional integration of smooth functions over cubes*,
Numerische Mathematik, volume 75, pages 79-97, 1996.

Sparse Grid Collocation

- Introduction
- Integrals Arising from Stochastic Problems
- Smolyak's Sparse Grid Definition
- **Aspects of Sparse Grids**
- Numerical Examples
- Conclusion

ASPECTS: The Nested Clenshaw-Curtis Sparse Grid

.

The Smolyak definition does not specify the properties of the 1D quadrature rules used to form product rules.

A popular choice uses the 1D Clenshaw Curtis rule (based at the zeros of the Chebyshev polynomials). A nested sequence of rules is selected, of orders 1, 3, 5, 9, 17, 33, ... so the rule of one level includes the points of the previous one. This is the approach in Novak and Ritter.

Nesting keeps our point count low by reusing points; on the other hand, rules of successive levels double in size, which is a potential (but fixable) pitfall.

ASPECTS: Computing X and W for Clenshaw Curtis

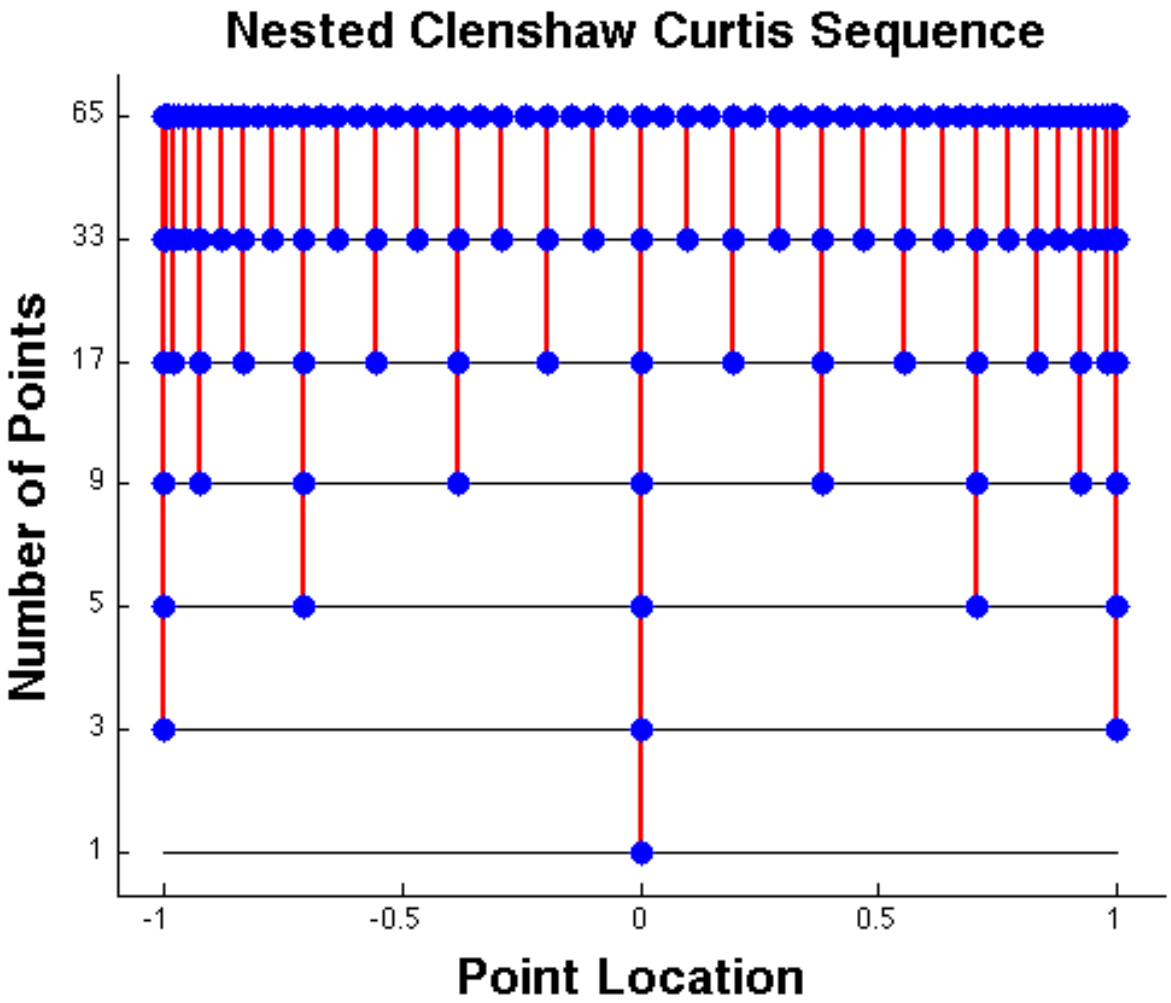
MATLAB code to compute a Clenshaw Curtis rule of order N:

```
theta(1:n) = ( n - 1 : -1 : 0 ) * pi / ( n - 1 );
x(1:n) = cos ( theta(1:n) );
w(1:n) = 1.0;

for i = 1 : n
  for j = 1 : floor ( ( n - 1 ) / 2 )

    if ( 2 * j == ( n - 1 ) )
      b = 1.0;
    else
      b = 2.0;
    end

    w(i) = w(i) - b * cos ( 2 * j * theta(i) ) / ( 4 * j * j - 1 );

  end
end

w(1)     =       w(1)     / ( n - 1 );
w(2:n-1) = 2.0 * w(2:n-1) / ( n - 1 );
w(n)     =       w(n)     / ( n - 1 );
```

http://people.sc.fsu.edu/~jburkardt/m_src/quadrule_fast/quadrule_fast.m

ASPECTS: We Choose a Nested Clenshaw Curtis Sequence

Each grid includes all points from the previous one.
(However, the size of the grid grows more rapidly than required.)



ASPECTS: We Choose a Nested Clenshaw Curtis Sequence

We can index our rules by level **L**, writing **CC(0)** for the the first rule, counting the points in the rule as the order **O**, and denoting the polynomial precision by **P**.

Rule	CC(0)	CC(1)	CC(2)	CC(3)	CC(4)	CC(5)	
L	0	1	2	3	4	5	
O	1	3	5	9	17	33	$2^{L-1}+1$
P	1	3	5	9	17	33	$P = O$

Because our 1D rules have odd order, instead of **P=O-1** we will actually have **P=O**, that is, a 1 point rule gets constants **and** linears, a three point rule picks up cubics, and so on.

ASPECTS: The First Three CC Rules

The CC-based Smolyak rules $\mathcal{A}(L = 0/1/2, M = 2)$ are:

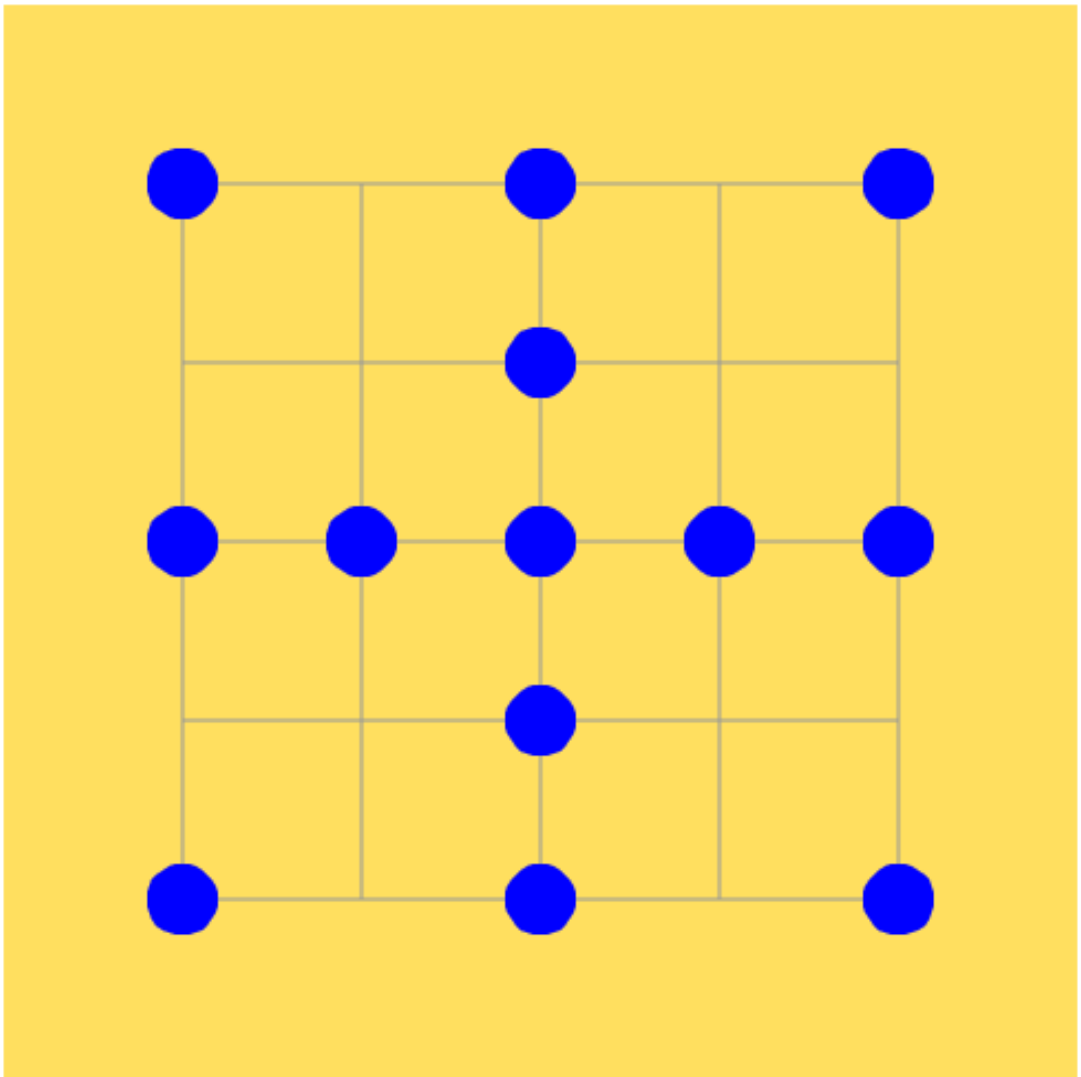
$$\mathcal{A}(0,2) = CC(0) \otimes CC(0)$$

$$\begin{aligned}\mathcal{A}(1,2) = & CC(1) \otimes CC(0) \\ & + CC(0) \otimes CC(1) \\ & - CC(0) \otimes CC(0)\end{aligned}$$

$$\begin{aligned}\mathcal{A}(2,2) = & CC(2) \otimes CC(0) \\ & + CC(1) \otimes CC(1) \\ & + CC(0) \otimes CC(2) \\ & - CC(1) \otimes CC(0) \\ & - CC(0) \otimes CC(1)\end{aligned}$$

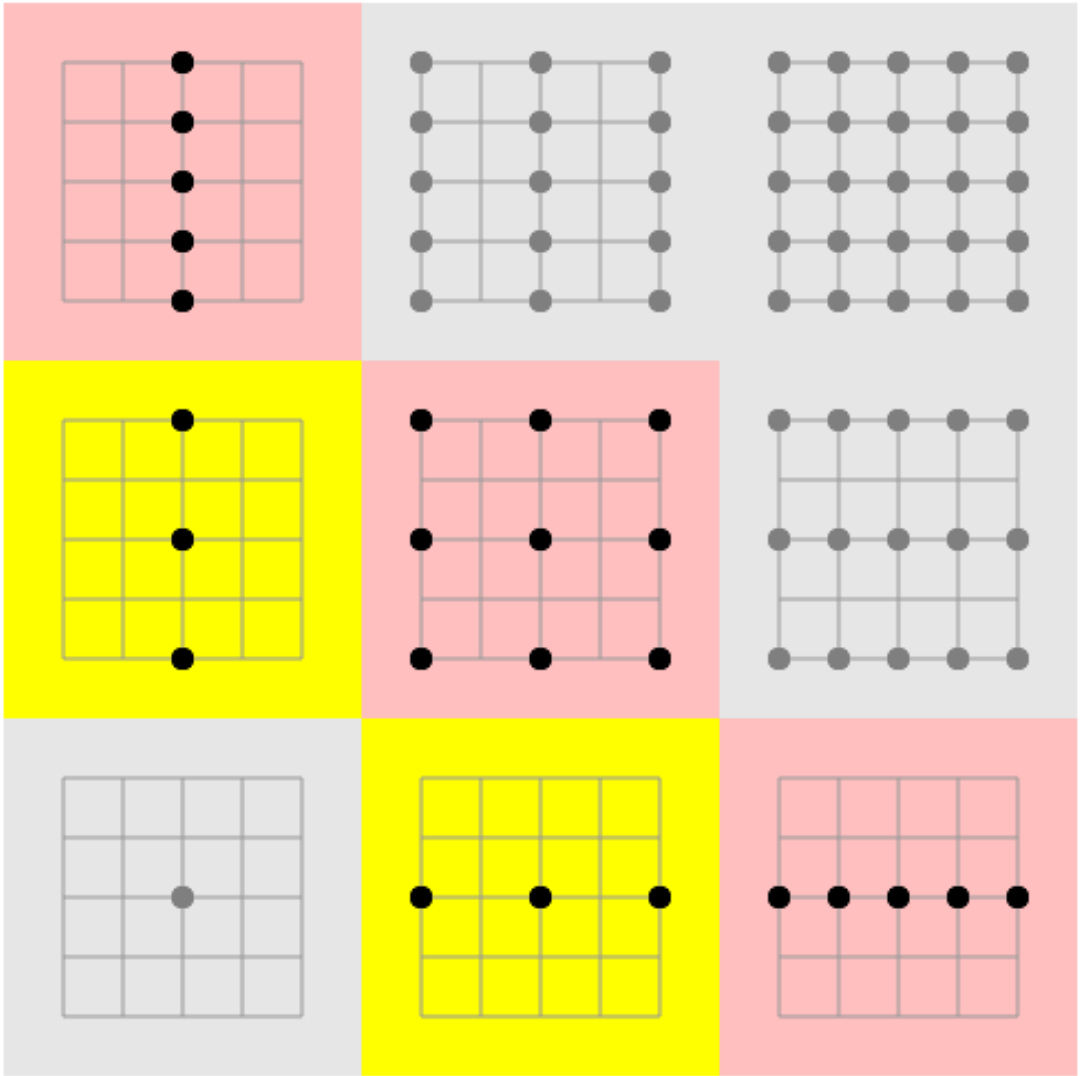
(For higher dimensions, we don't just have +1 and -1 as coefficients, but they will still be combinatorial coefficients.)

ASPECTS: The Quadrature Points of $\mathcal{A}(2,2)$



ASPECTS: $\mathcal{A}(2,2) = 2x0 + 1x1 + 0x2 - 1x0 - 0x1$

It helps to see the underlying product rules that were combined:



ASPECTS: $\mathcal{A}(2,2)$ as good as $C(5) \otimes C(5)$

The picture correctly suggests that the Smolyak combination of the 5 lower order grids is (essentially) as precise as the 5x5 product grid in the upper right corner.

But the Smolyak grid $\mathcal{A}(2,2)$ uses 13 points, the product grid $C(5) \otimes C(5)$ uses 25.

The Smolyak definition chooses a collection of lower order product grids that capture the information necessary to approximate all the monomials of interest, and then combines this information correctly to produce a good integral estimate.

(Because of the nesting choice we made, the precision results are not so neat for higher levels...)

ASPECTS: Precision of a General CC Sparse Grid

Novak and Ritter show that for sparse grids based on the 1D Clenshaw Curtis rule, the precision is related to the level by:

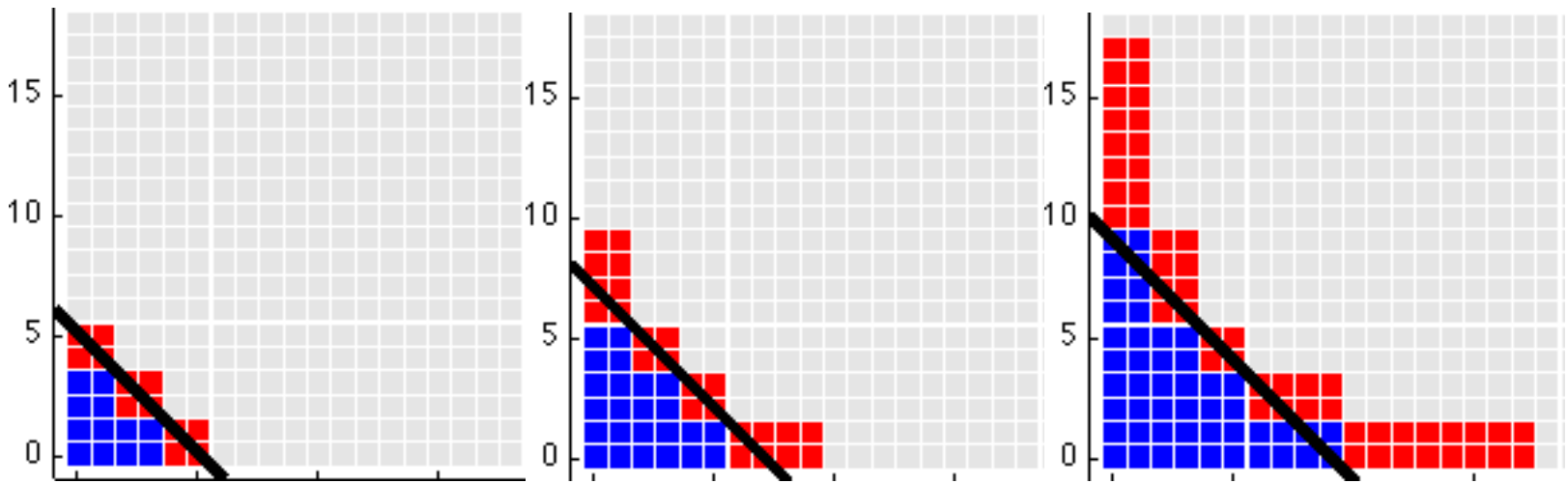
$$P = 2 * L + 1$$

The first sparse grid (L=0) picks up constants and linears, the second adds quadratics and cubics, and so on. This means that our family of sparse grids has a predictable precision behavior based on the level, regardless of dimension.

If our CC-based sparse grids were perfectly efficient, the level L rule would pick up exactly the monomials up to precision P and no more. A precision graph will show that we do better than a product rule, but still have some inefficiency.

ASPECTS: What Monomials are Precisely Integrated?

Here are the precisions of $\mathcal{A}(2,2), \mathcal{A}(3,2), \mathcal{A}(4,2)$. The diagonal black line "fences off" the monomials of degree 2*L+1, which the sparse grid must integrate precisely. With increasing level, the rules spill over the fence, suggesting some inefficiency.



(The red squares are monomials just added on this level.)

ASPECTS: Reducing the Nesting Overhead

Our sparse grid of level L will have the precision $P = 2 * L + 1$ as long as our family of 1D rules has **at least** that same precision sequence:

L	0	1	2	3	4	5	6	7	8	9	10...
P minimal:	1	3	5	7	9	11	13	15	17	19	21...

But our 1D rules double the number of points with each level, in order to effect nesting...which we want to control the point count.

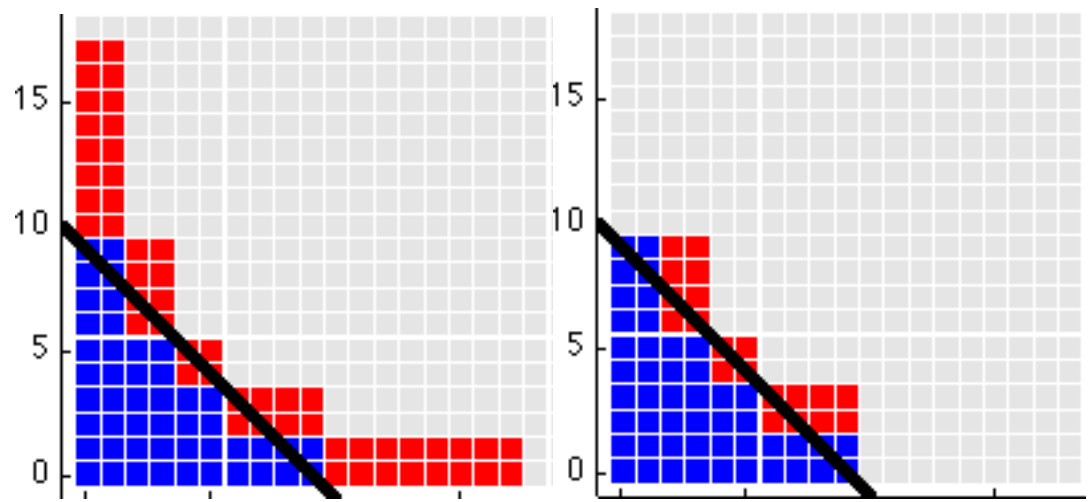
P supplied:	1	3	5	9	17	33	65	129	255	513	1025...

We could keep nesting but reduce the cost of doubling, by reusing a 1D rule if it satisfies the precision requirement.

P suggested:	1	3	5	9	9	17	17	17	17	33	33...

ASPECTS: Standard and Slow Growth versions of $\mathcal{A}(4, 2)$

Using the slower growth strategy preserves our overall precision requirement while cutting down on the order of the finest grid, and hence the number of function evaluations.



Both rules achieve precision $P = 2 * L + 1 = 9$;
the standard rule uses 65 points, the slow rule 49.

- Introduction
- Integrals Arising from Stochastic Problems
- Smolyak's Sparse Grid Definition
- Aspects of Sparse Grids
- **Numerical Examples**
- Conclusion

NUMERICAL: Function Evaluations Versus Accuracy

.

We estimate the integral of $f(x)$ over an M-dimensional hypercube $[a, b]^M$ using N points or function evaluations.

The "cost" of the estimate is N. If an estimate is not satisfactory, it's important to know how fast the error is likely to go down if we increase N and try again.

For the Monte Carlo method, the rate of error decay is known to be $O(\frac{1}{\sqrt{N}})$. The rate is independent of spatial dimension M, and essentially independent of the smoothness of $f(x)$.

Compare Monte Carlo and sparse grid values of N and accuracy.

NUMERICAL: Sparse Grids Require Smoothness!

Let $f(x)$ be the characteristic function of the unit ball in 6D:

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

Can you see why negative estimates are possible for the sparse grid, even though the integrand is never negative?

Sparse grids need smooth integrands; and because sparse grids use extrapolation, they are liable to unpleasant errors otherwise.

NUMERICAL: MC Quadrature Can be Slow

Monte Carlo doesn't diverge, but look how hard we have to work to get three places of accuracy for the characteristic function of the unit ball in 6D.

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

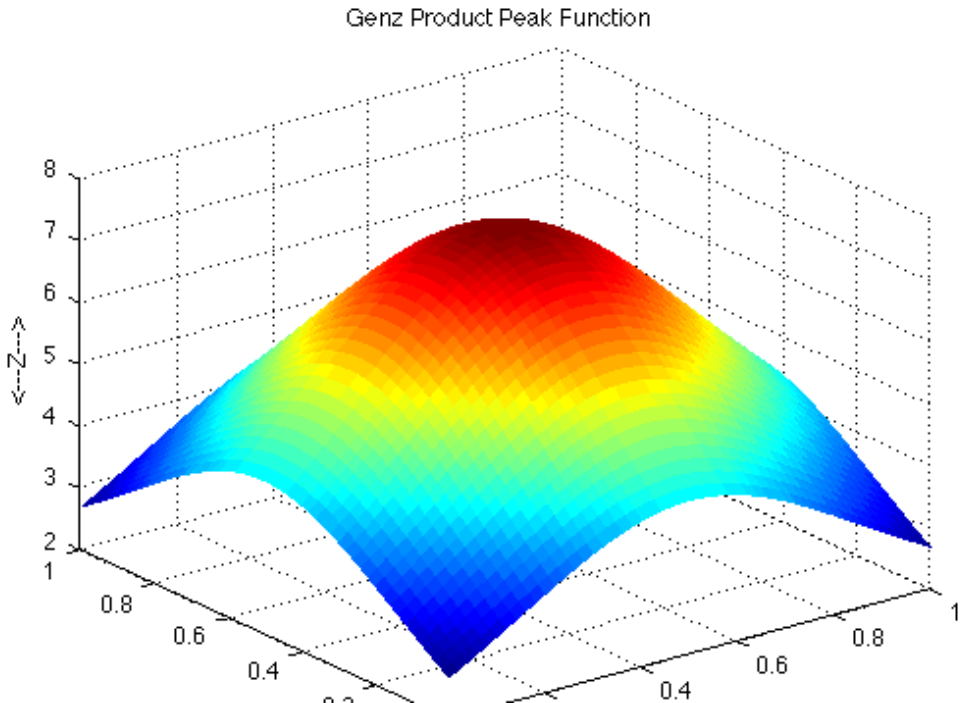
Should we want one more digit of accuracy, we can expect to need **100 times as many points** $\approx$ 3.3 billion points.

NUMERICAL: Genz Product Peak Test in 6D

Alan Genz provided six high dimensional test integrals to catch various strengths and failings of a given procedure.

The **product peak function** is defined on the unit hypercube, with given C and Z vectors, and is smooth:

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

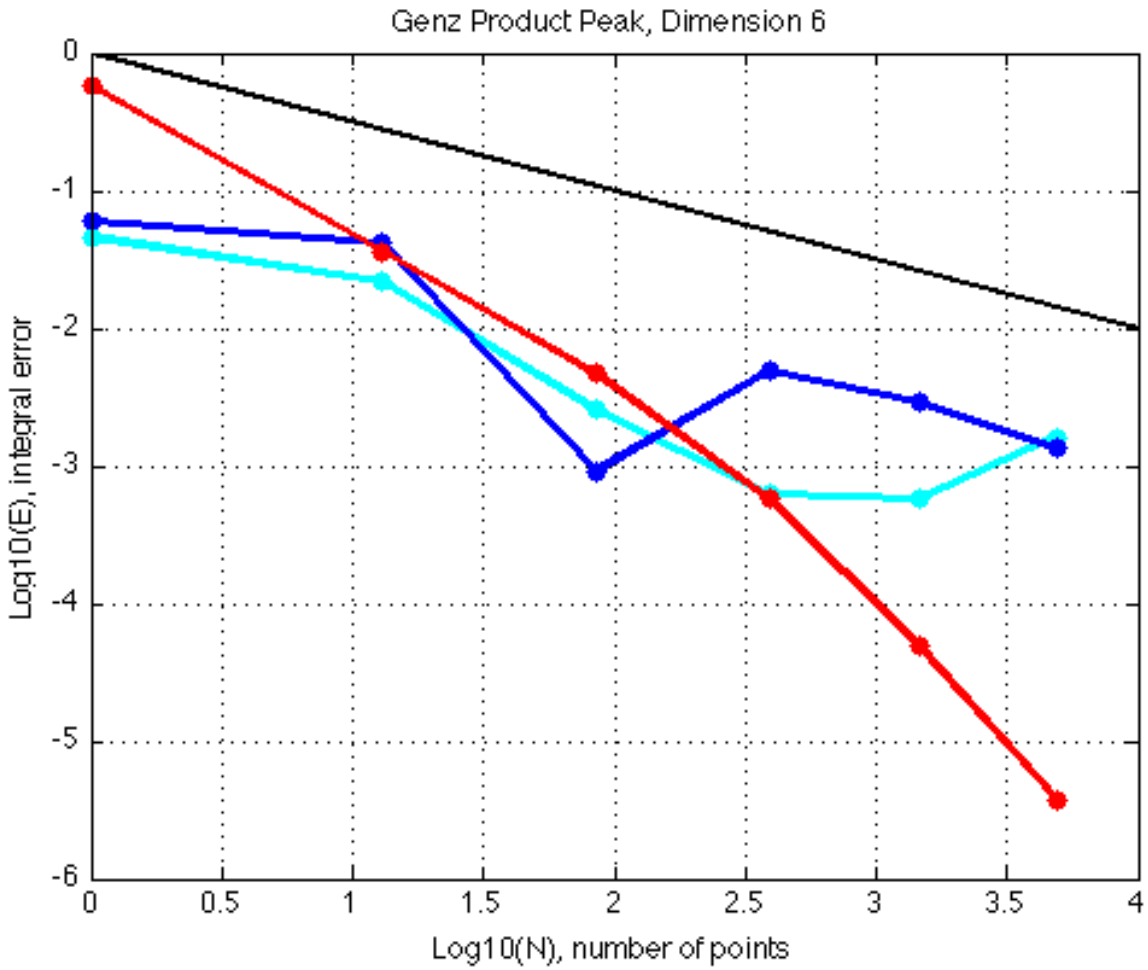


NUMERICAL: Genz Product Peak Test in 6D

Red: Sparse grid estimate

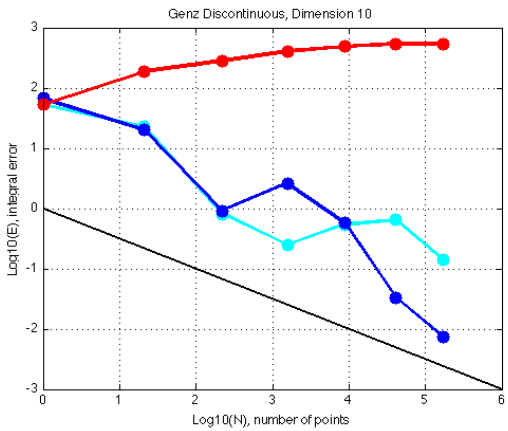
Blue & Cyan: MC estimates

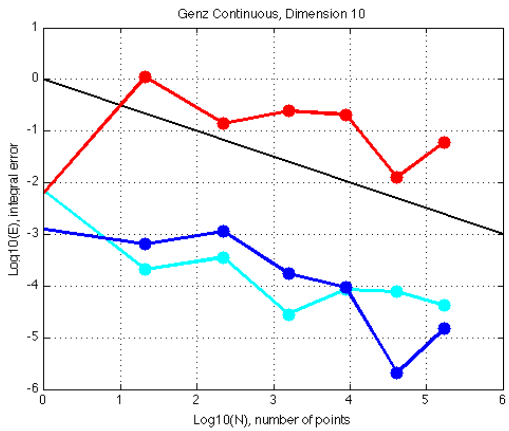
Black: Expected MC Rate of Decrease

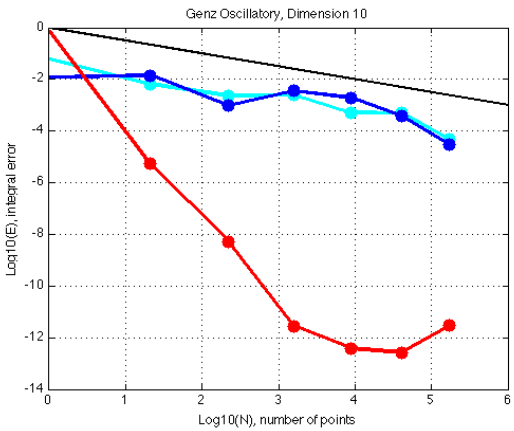


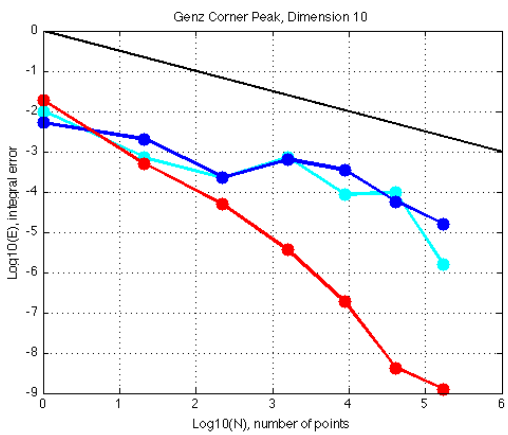
NUMERICAL: Genz Tests in 10D

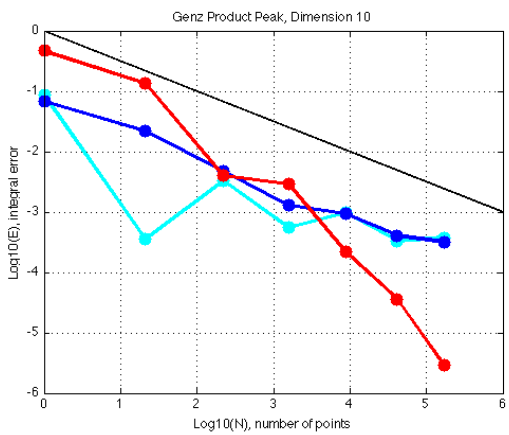
Discontinuous, Continuous, Oscillatory
Corner Peak, Product Peak, Gaussian
(sparse grid estimate in red)

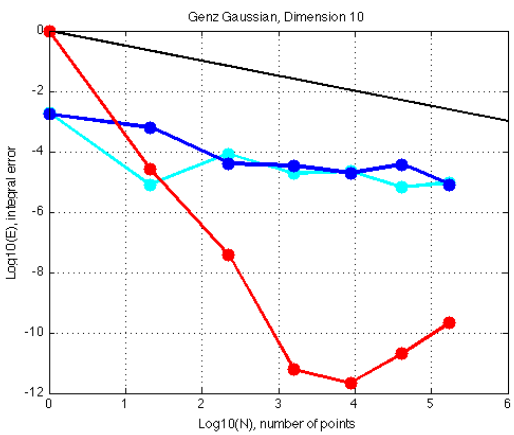












NUMERICAL: The Poisson Equation

Let's return to the stochastic Poisson equation considered earlier:

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

Our integration problem seeks the expected value of $u(x, y; \omega)$, assuming we have a probabilistic model for the stochastic influence.

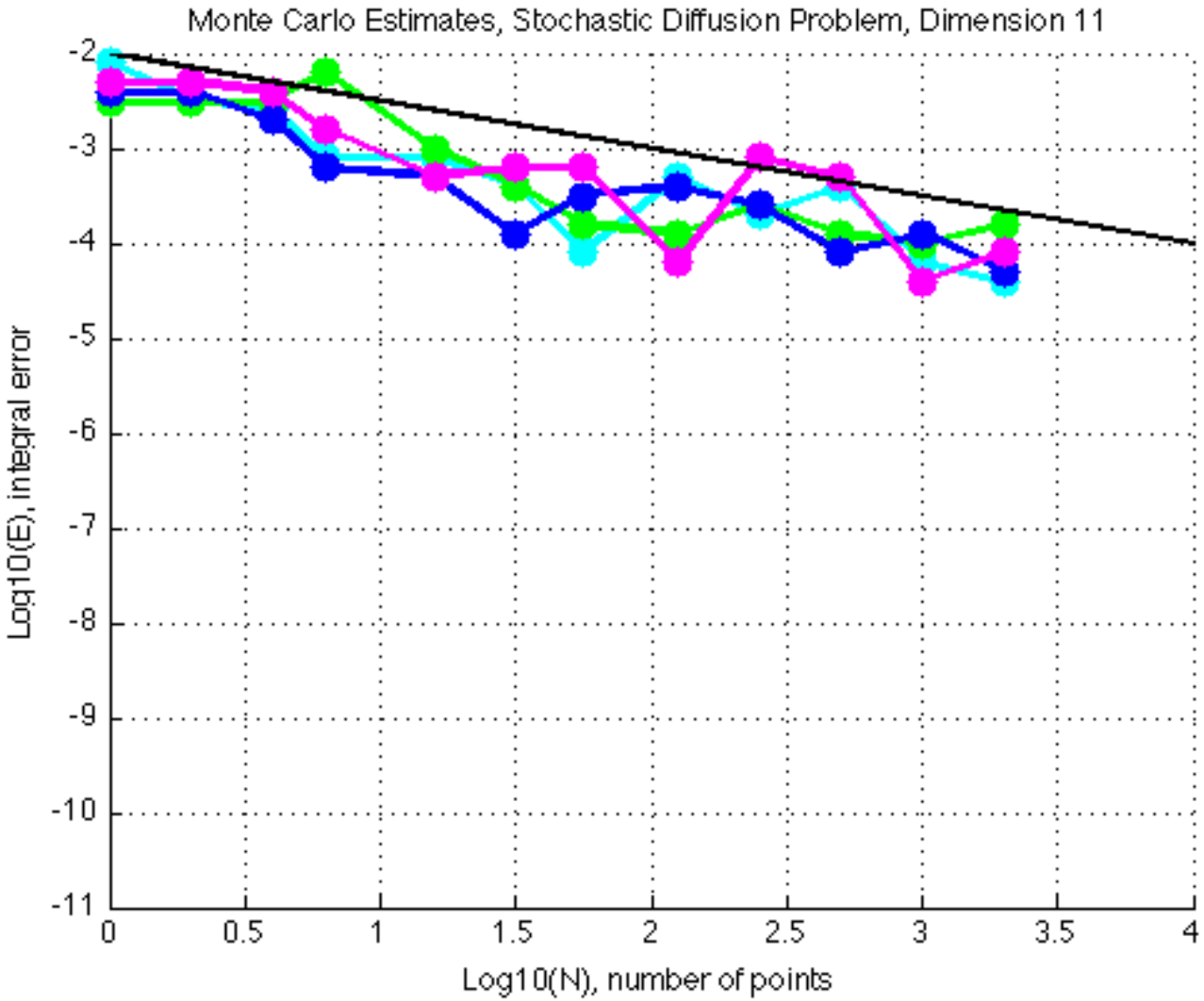
Monte Carlo: select a random set of parameters ω according to $pr(\omega)$, solve the Poisson equation for u, and average.

Sparse grid: choose a level, defining a grid of ω values, solve the Poisson equation for u, multiply by the probability, and take a weighted average.

Clayton Webster, *Sparse grid stochastic collocation techniques for the numerical solution of partial differential equations with random input data*, PhD Thesis, Florida State University, 2007.

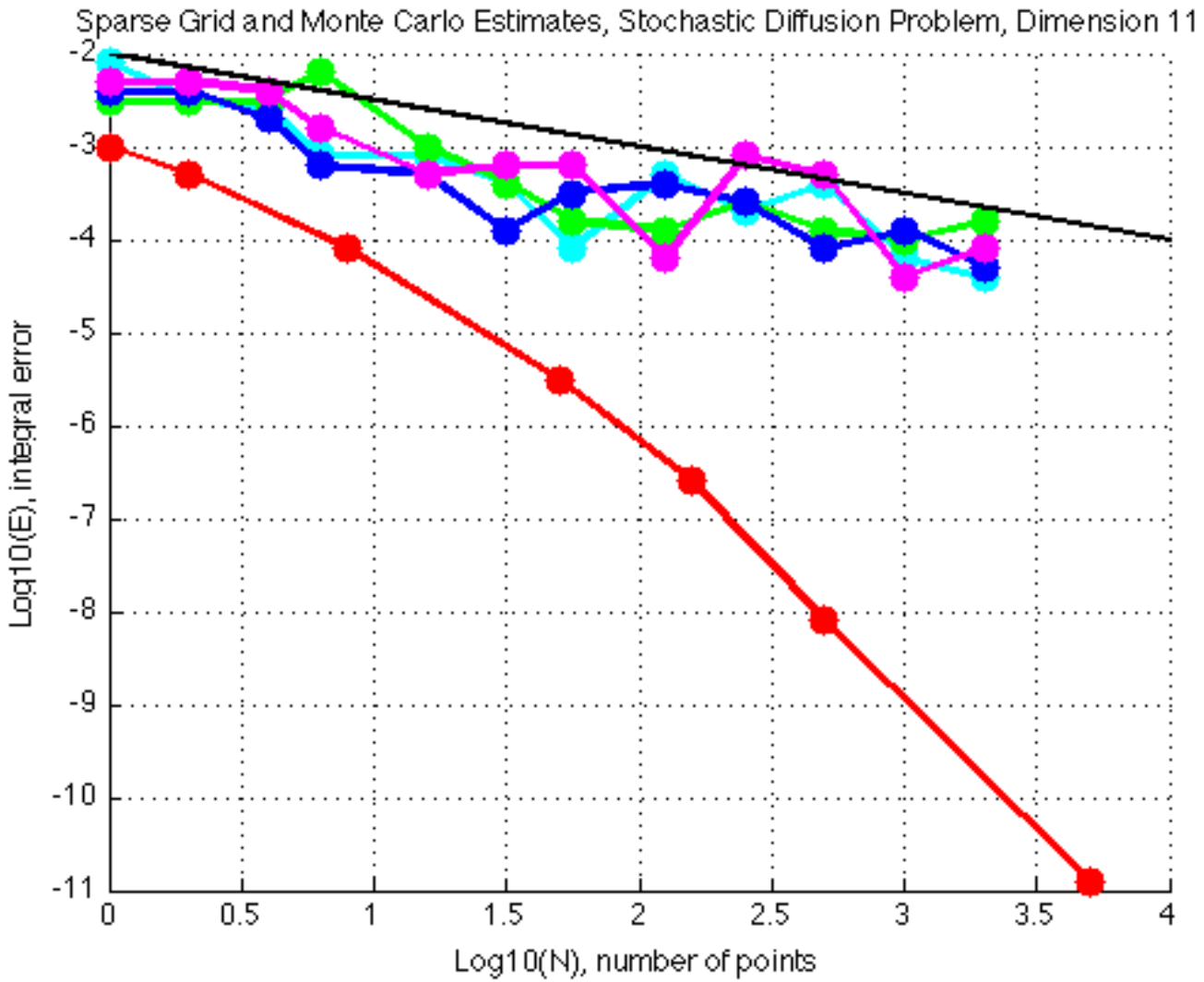
NUMERICAL: Four Monte Carlo Estimates

The black line is the Monte Carlo trend.



NUMERICAL: Sparse Grid Versus Monte Carlo

The sparse grid estimates converge rapidly.



NUMERICAL: Remarks

For the stochastic diffusion problem, $u(x, y; \omega)$ has a very smooth dependence on the perturbations in ω.

For this reason, a sparse grid can sample the solution for a small set of perturbations ω and accurately estimate the expected value.

If we had a code to solve the original Poisson equation for a given conductivity field, the sparse grid procedure simply needs to call that unmodified code with different conductivities.

This is why sparse grids are called a nonintrusive method. Other procedures for dealing with uncertain or stochastic influences may require extensive changes, new variables, and a larger coupled system to solve.

Sparse Grid Collocation

- Introduction
- Integrals Arising from Stochastic Problems
- Smolyak's Sparse Grid Definition
- Aspects of Sparse Grids
- Numerical Examples
- **Conclusion**

CONCLUSION: High Dimensional Integration is Intractable

In general, estimating the integral of an arbitrary function in a high dimensional space is an intractable problem.

Nonetheless, we are able to estimate the high dimensional integrals associated with stochastic problems and uncertainty quantification when the mathematical modelling of uncertainty results in integrands that are smooth, or localized, or can be well approximated by certain easily integrated density functions.

Sparse grids are a powerful tool for treating such special integrands, whose estimated integrals can be pulled out of an otherwise impenetrable high dimensional jungle!

CONCLUSION: Software

SPINTERP, a MATLAB program, by Andreas Klimke, is a great way to explore the power of sparse grids.

```
x = spgrid ( l, m )
```

returns the points of a sparse grid of level **L** in dimension **M**.

To estimate an integral:

```
z = spvals ( @fun, m )
q = spquad ( z )
```

SPINTERP can also interpolate and optimize using sparse grids.

---

http://www.ians.uni-stuttgart.de/spinterp/

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.

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