Slow Growth for Sparse Grids

John Burkardt, Clayton Webster, Guannan Zhang

....

http://people.sc.fsu.edu/∼jburkardt/presentations/...
slow_growth_2014_savannah.pdf

....

SIAM UQ Conference
31 March - 03 April 2014

References:

- Florian Heiss, Viktor Winschel,
  Likelihood approximation by numerical integration on sparse grids,
  Journal of Econometrics, Volume 144, Number 1, May 2008, pages 62-80.
  Matlab program for sparse grid generation with slow growth
- Erich Novak, Klaus Ritter,
  Simple cubature formulas with high polynomial exactness,
  Constructive Approximation, Volume 15, Number 4, December 1999, pages 499-522.
  Exactness constraint for sparse grids
- Knut Petras,
  Smolyak Cubature of Given Polynomial Degree with Few Nodes for Increasing Dimension, Numerische Mathematik, Volume 93, Number 4, February 2003, pages 729-753.
  C program for sparse grid generation with slow growth
- Miroslav Stoyanov,
  User Manual: TASMANIAN Sparse Grids,
  ORNL Report, Oak Ridge National Laboratory, 2013.
  C++ program for sparse grid generation with slow growth

- **Introduction**
- Clenshaw-Curtis
- Gauss-Legendre
- Gauss-Patterson
- Conclusion

INTRO: Need to Estimate Multidimensional Integrals

My introduction to sparse grids began with the classic example based on the nested points of the 1D exponential Clenshaw-Curtis rule (CCE), using 1, 3, 5, 9, 17, 33, 65, 129, 257, 513, 1025 points.

I could see multidimensional quadrature errors decrease for smooth integrands.

I tested the exactness of the rule and saw that level ℓ could integrate polynomials of total degree $2\ell + 1$ exactly.

Novak & Ritter showed that to get this exactness, it was sufficient that the 1D rules have exactness 1, 3, 5, 7, 9, 11, 13, 15...

The 1D CCE rules are exponential; the exactness requirement is linear.

Mustn't this have some disadvantage?

If so, is there a remedy?

Once we have specified a index list of 1D quadrature rules or "factors", Smolyak allows us to generate a sparse grid in any dimension.

If we set up the Smolyak machinery, and ask it to generate a "sparse grid" in 1D, then we get back the original 1D quadrature rules.

It is common to expect a sparse grid of level ℓ to have an exactness that grows linearly with the level:

$$p = 2\ell + 1 \text{ (Novak \& Ritter)}$$

Now suppose we generate a 1D Clenshaw-Curtis "sparse grid"...

ℓ = level	0	1	2	3	4	5	6	7	8	9	10	...
n = points	1	3	5	9	17	33	65	129	257	513	1025	...
p = exactness	1	3	5	9	17	33	65	129	257	513	1025	...
p(necessary)	1	3	5	7	9	11	13	15	17	19	21	...

In 1D, order and exactness grow exponentially:

$$n = 2^\ell + 1, \quad 1 \le \ell$$

$$p = 2^\ell + 1 = n$$

CC: Nesting

Paradoxically, we use exponential growth in an attempt to **reduce** point counts (in high dimensions).

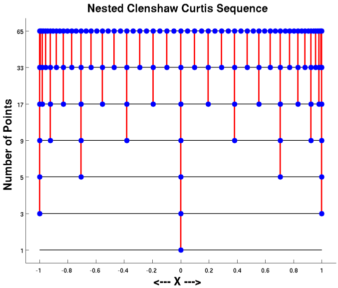
The points of a sparse grid are the logical sum of the points of a collection of product grids that satisfy a constraint on their definition.

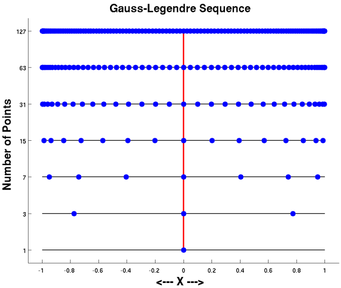
If all these product rules are defined using a 1D **nested** family, then when we gather together the logical sum of the product grids, the total number of points can be greatly reduced.

Compare in 2D the nested CCE versus the non-nested GLE (Gauss-Legendre exponential) sparse grids.

ℓ = level	0	1	2	3	4	5	6	7	8	9	...
n (CCE)	1	5	13	29	65	145	321	705	1537	3329	...
n (GLE)	1	5	22	75	224	613	1578	3887	9268	21561	...

CC: Nested CCE family / Nonnested GLE family



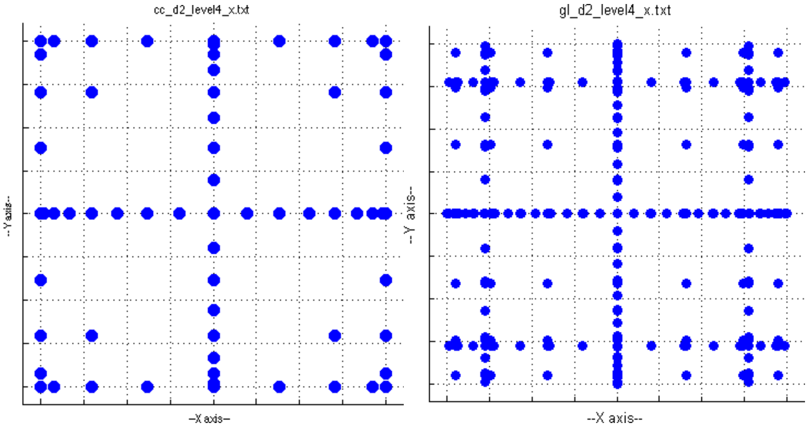


The CCE family is completely nested;

in the GLE family, only the 0.0 value is repeated.

The benefits of nesting become critical as dimension increases.

CC: Nesting in 2D Sparse Grids



Nesting keeps the Clenshaw Curtis sparse grid efficient (65 points). The Gauss-Legendre sparse grid has 224 distinct points.

CC: Keep Nesting, Incomplete Chebyshev?

A nested family of Chebyshev rules seems to require jumping twice as far each time we increase the level. We can explore a modification we might call the Clenshaw Curtis Incomplete (CCI) family, which uses the Chebyshev family as a guide, but only adds two points with each level increase.

level	order	commment
0	1	CCE rule 0
1	3	CCE rule 1
2	5	CCE rule 2
3	7	incomplete CCE rule 3
4	9	CCE rule 3
5	11	incomplete CCE rule 4
6	13	incomplete CCE rule 4
7	15	incomplete CCE rule 4
8	17	CCE rule 4

For incomplete rules, we have to (pre)-compute the weights from basic principles; we need to monitor possible negative weights.

CC: Can We Abandon Nesting?

An alternative (CCL) to the exponentially growing version of the CC rule would be to use a Clenshaw-Curtis family of odd orders and linear growth, $n = 1, 3, 5, 7, 9, \ldots$, which will exactly meet the Novak & Ritter exactness requirement.

This family is not nested. So our tradeoff is that our sparse grids will be combining product rules of lower order, but with more distinct points.

What is the effect in 2D?

ℓ = level	0	1	2	3	4	5	6	7	8	9	...
n (CCE)	1	5	13	29	65	145	321	705	1537	3329	...
n (CCL)	1	5	13	29	57	105	177	281	425	611	...

The CCL rule doesn't show an advantage until the underlying factors begin to differ, after which we see a big reduction.

Does this 2D result carry over to higher dimensions?

CC: Keep Nesting, Slow Exponentiation

Yet another alternative (CCS) retains the exponentially growing factor family, but uses the lowest such rule satisfying the exactness requirement.

In other words, we start with the CCE factor family
$n = 1, 3, 5, 9, 17, 33...$, but repeat rules where possible.

Compare the CCE, CCL and CCS 1D factor families:

ℓ = level	0	1	2	3	4	5	6	7	8	9	...
p (required)	1	3	5	7	9	11	13	15	17	19	...
n (CCE)	1	3	5	9	17	33	65	125	257	513	...
n (CCL)	1	3	5	7	9	11	13	15	17	19	...
n (CCS)	1	3	5	9	9	17	17	17	17	33	...

The CCS factor family grows faster than CCL, and does so in exponential "jumps" but makes those jumps far less often than the CCE family, and inherits the advantages of nestedness.

If we build a 2D sparse grid from the CCS rule, what happens?

Does the 2D sparse grid inherit the “stutter” of the 1D factors?

ℓ = level	0	1	2	3	4	5	6	7	8	9	...
n (CCE)	1	5	13	29	65	145	321	705	1537	3329	...
n (CCL)	1	5	13	29	57	105	177	281	425	611	...
n (CCS)	1	5	13	29	49	81	129	161	225	257	...

and for 6D:

ℓ = level	0	1	2	3	4	5	6	7	8	9	...
n (CCE)	1	13	85	389	1,457	4,865	15,121	44,689	127,105	350,657	...
n (CCL)	1	13	85	389	1,433	4,533	12,961	33,817	82,153	188,039	...
n (CCS)	1	13	85	389	1,409	4,289	11,473	27,697	61,345	126,401	...

And for 10D:

ℓ = level	0	1	2	3	4	5	6	7	...
n (CCE)	1	21	221	1,581	8,801	41,265	171,425	652,065	...
n (CCL)	1	21	221	1,581	8,761	40,425	162,385	584,665	...
n (CCS)	1	21	221	1,581	8,721	39,665	155,105	536,705	...

As *d* increases, the CCL and CCS advantages are delayed and decreased. (In high dimensions, very low order rules predominate.)

- Introduction
- Clenshaw-Curtis
- **Gauss-Legendre**
- Gauss-Patterson
- Conclusion

GL: GLE Factor Family

Try a Gauss-Legendre Exponential family (GLE), orders 1, 3, 7, 15, ...

ℓ = level	0	1	2	3	4	5	6	7	8	9	10	...
n = points	1	3	7	15	31	63	127	255	511	1023	2047	...
p = exactness	1	5	13	29	61	125	253	509	1021	2045	4093	...
p(necessary)	1	3	5	7	9	11	13	15	17	19	21	...

GLE is an open family, CCE is closed.

The GLE order growth is exponential, and double that of CCE.

$$n(\mathrm{GLE})(\ell) = 2^{\ell+1} - 1$$
$$p(\mathrm{GLE})(\ell) = 2 \cdot (2^{\ell+1} - 1) - 1 = 2 \cdot n(\mathrm{GLE})(\ell) - 1$$

Exactness is 4 times that of CCE, fantastically above Novak & Ritter.

GL: GLL Factor Family

The GL family is unsuitable for nesting; exponential growth is misguided.

Linear growth (GLL) rule uses lowest order rule satisfying Novak & Ritter. GLL rules have orders 1, 2, 3, 4, ... because 1D rules are more exact.

Now that we got the growth rate under control, consider a tiny bit of nesting, defining the GLO rule, to uses the lowest odd order rule satisfying Novak & Ritter.

ℓ = level	0	1	2	3	4	5	6	7	8	9	10	...
n(GLE)	1	3	7	15	31	63	127	255	511	1023	2047	...
p(necessary)	1	3	5	7	9	11	13	15	17	19	21	...
n(GLL)	1	2	3	4	5	6	7	8	9	10	11	...
p(GLL)	1	3	5	7	9	11	13	15	17	19	21	...
n(GLO)	1	3	3	5	5	7	7	9	9	11	11	...
p(GLO)	1	3	5	9	9	13	13	17	17	21	21	...

$$n(\text{GLL})(\ell) = 2\ell + 1$$

$$n(\text{GLO})(\ell) = 2 \cdot \lfloor \frac{\ell + 1}{2} \rfloor + 1$$

Will the GLO tradeoff improve the GLL option?

2D:

ℓ = level	0	1	2	3	4	5	6	7	8	9	...	
n (GLE)	1	5	21	73	221	609	1,573	3,881	9,261	21,553	49,205	...
n (GLL)	1	5	13	29	53	89	137	201	281	381	501	...
n (GLO)	1	5	9	17	29	41	65	81	121	141	201	...

10D:

ℓ = level	0	1	2	3	4	5	6	7	...	
n (GLE)	1	21	261	2,441	18,881	126,925	764,365	4,208,385	21,493,065	...
n (GLL)	1	21	221	1,581	8,761	40,405	162,025	581,385	1,904,465	...
n (GLO)	1	21	201	1,201	5,281	19,165	61,285	177,525	474,885	...

15D:

ℓ = level	0	1	2	3	4	5	6	7	...
n (GLE)	1	31	541	6,911	71,621	635,687	4,995,357	35,537,007	...
n (GLL)	1	31	511	5,921	53,921	409,727	2,695,967	15,751,937	...
n (GLO)	1	31	451	4,151	27,671	145,697	644,937	2,506,137	...

GLO outperforms GLL rule, and does do by using bigger rules!

This suggests the powerful benefit of multidimensional nesting.

GP: The Gauss-Patterson Factor Family

Nesting and the doubled exactness of Gaussian rules are two techniques that have a significant influence on the properties of sparse grids.

This suggests looking at a Gauss-Patterson (GP) factor family.

The GP family begins with the 1 and 3 point GL rules. Thereafter, given a rule with n points, the next rule fixes those points, and adds $n + 1$ new points, enforcing nesting. A Gauss procedure squeezes out the best accuracy possible, given the constraint that the old points must not be moved.

The result is a nested family with the same exponential growth as GLE and somewhat reduced exactness,

GP: GPE Factor Family

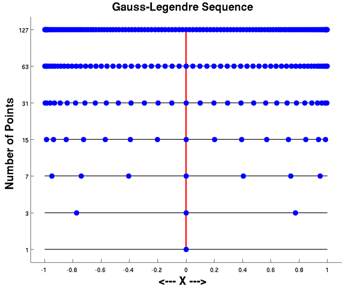
Here is the exactness table for the GPE 1D factor family:

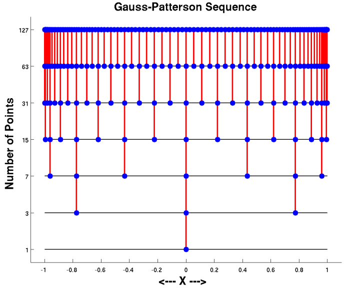
ℓ = level	0	1	2	3	4	5	6	7	8	9	10	...
n = points	1	3	7	15	31	63	127	255	511	1023	2047	...
p = exactness	1	5	11	23	47	95	191	383	767	1535	3071	...
p(necessary)	1	3	5	7	9	11	13	15	17	19	21	...

The number of points is the same as for GLE, while the exactness is reduced:

$$n(\text{GPE})(\ell) = 2^{\ell+1} - 1$$
$$p(\text{GPE})(\ell) = 1.5 \cdot (2^{\ell+1} - 1) + 0.5 = 1.5 \cdot n(\text{GPE})(\ell) + 0.5$$

GP: GLE versus GPE





The GLE family is not nested, but the GPE family is, and retains much of the exactness of Gauss rules.

Here is a quick comparison of GLE and GPE in 2D:

ℓ = level	0	1	2	3	4	5	6	7	8	9	10	...
n (GLE)	1	5	21	73	221	609	1,573	3,881	9,261	21,553	49,205	...
n (GPE)	1	5	17	49	129	321	769	1,793	4,097	9,217	20,481	...

GP: The GSP Version of GPE

Let's go ahead and define a GPS family which only selects the next 1D factor when the Novak & Ritter exactness constraint requires it.

Here are sample point counts comparing GPE and GPS for 2D:

ℓ = level	0	1	2	3	4	5	6	7	8	9	10	...
n (GPE)	1	5	17	49	129	321	769	1,793	4,097	9,217	20,481	...
n (GPS)	1	5	9	17	33	33	65	97	97	161	161	...

and for 6D:

ℓ = level	0	1	2	3	4	5	6	7	8	9	...
n (GPE)	1	13	97	545	2,561	10,625	40,193	141,569	4,710,417	14,960,657	...
n (GPS)	1	13	73	257	737	1,889	4,161	8,481	16,929	30,689	...

and for 10D:

ℓ = level	0	1	2	3	4	5	6	7	8	9	...
n (GPE)	1	21	241	2,001	13,441	77,505	397,825	1,862,145	8,085,505	32,978,945	...
n (GPS)	1	21	201	1,201	5,281	19,105	60,225	169,185	434,145	1,041,185	...

- Introduction
- Clenshaw-Curtis
- Gauss-Legendre
- Gauss-Patterson
- **Conclusion**

CON: Compare the Champions

In summary, we have "improved" versions of CCE, GLE and GPE. How do they stack up against each other?

2D:

ℓ = level	0	1	2	3	4	5	6	7	8	9	10	...
n (CCS)	1	5	13	29	49	81	129	161	225	257	385	...
n (GLO)	1	5	9	17	29	41	65	81	121	141	201	...
n (GPS)	1	5	9	17	33	33	65	97	97	161	161	...

10D:

ℓ = level	0	1	2	3	4	5	6	7	8	9	...
n (CCS)	1	21	221	1,581	8,721	39,665	155,105	536,705	1,677,665	4,810,625	...
n (GLO)	1	21	201	1,201	5,281	19,165	61,285	177,525	474,885	1,192,425	...
n (GPS)	1	21	201	1,201	5,281	19,105	60,225	169,185	434,145	1,041,185	...

CON: Remarks

The classic Clenshaw-Curtis sparse grid achieves nestedness at the cost of exponential growth.

In low dimensions and moderate levels, this results in a noticeable and unneccessary excess number of function evaluations.

Nesting, **Gauss-rules**, and **slow-growth** procedures control point growth, and “buy” you extra levels of sparse grids.

For slow growth procedures on $[-1, +1]$ or $(-\infty, +\infty)$, with a symmetric weight function, the Ritter & Novak exactness constraint is your guide.

The Gauss-Patterson (GPS) sparse grid is one example using all the ideas of nesting, (semi)-Gauss rules, and slow growth.

Software implementations appear in **nwspgr** (Heiss & Winschel), **smolpack** (Petras), and **tasmanian** (Stoyanov).