Gaussian Kissing

Peter Jaeckel

OTC Analytics

Autumn 2023

Outline

1. The curse of dimensionality
2. Basis function approaches
3. Radial Basis Functions
4. Meshless induction
5. Cluster generation
6. Cluster analysis
7. Gaussian Kissing
8. Appendix

The curse of dimensionality

- There are many applications for the calculation of forward-conditional expectation that, in one dimension, we would likely solve by the aid of backward induction solvers:
  - American / Bermudan options
  - CVA, DVA, FVA, and other XVA calculations require an expectation conditional on being positive (or negative), e.g.:

$$\mathsf{E}\big[(v(t,\boldsymbol{x}))_+ \cdot f(x)\big] = \mathsf{E}\big[v(t,\boldsymbol{x}) \cdot \mathbf{1}_{\{v(t,\boldsymbol{x})>0\}} \cdot f(x)\big] \tag{1.1}$$

    for some function $f(x) \geq 0$.

- In low dimensionalities, this can be computed directly with conventional, i.e., lattice-based finite differencing methods.
- The computational effort of lattice-based methods grows typically like

$$n^d$$

  where n is the number of lattice nodes in each dimension and d is the dimensionality.
- This n^d growth is also known as *the curse of dimensionality*.

- In moderate or higher dimensions, the required expectations can, in principle, be computed with Monte-Carlo simulations[1].

- When regression methods are used in (moderately) high dimensions for the interim conditional valuation criteria[2], typically, only linear regression functions are employed.

- This is because the computational effort of the required regression calculations grows dramatically when bi-linear or even higher order regression polynomials (or other basis functions) are used[3].

[1] For a survey of available American/Bermudan Monte Carlo methods, see, e.g., [JA10].
[2] as is in practice, nowadays, seemingly, always the case
[3] Your mileage may, of course, vary.

- The burden here is the calculation of the higher order cross-moments of the regression variables. There are

$$\begin{pmatrix} d+k \\ k \end{pmatrix} = \frac{(d+k)!}{k!}$$

to be computed[4]!

- Arguably, this is another variation of a *curse of dimensionality*.

[4] with k being the maximum total power of the multinomial

Basis function approaches

There are a variety of methods based on a decomposition

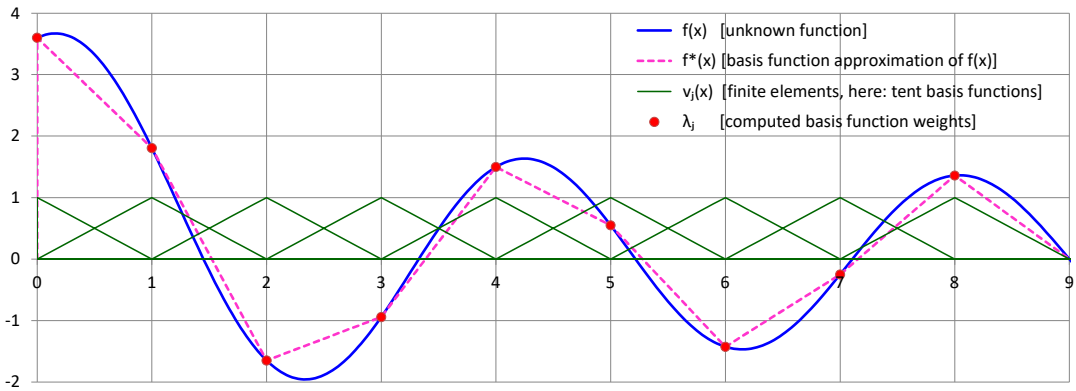
$$f(x) \simeq \sum_j \lambda_j v_j(x) \tag{2.1}$$

to solve PDEs like

$$[\partial_t f +]\ \mathcal{L} \cdot f = g(x[;t]) \tag{2.2}$$

where $\mathcal{L}$ is a generic advection-diffusion operator.

- Finite elements.
  - Spatially limited basis functions.
  - Usually piecewise linear, i.e., only C^0.
  - Second derivatives of $v_j(x)$ are removed from the PDE by means of integration by parts (variational formulation).
  - In one dimension, the finite elements are usually *tent functions*[5].



  - In one dimension, finite elements give us the same linear systems as the standard nearest-neighbour-stencil finite difference method.

[5] aka *triangular functions*

- Higher order polynomials, e.g., hp-FEM.
- Spectral bases (➡ Chebyshev polynomials)
- Wavelets
- **Radial basis functions**
- etc. etc.

- In all basis function approaches, ultimately, we need to compute the basis function weight vector $\boldsymbol{\lambda}$, e.g., by matching the function value $f(\boldsymbol{x})$ in a set of interpolation nodes $\{\boldsymbol{x}_i\}$:

$$V \cdot \boldsymbol{\lambda} \;=\; \boldsymbol{f} \tag{2.3}$$

with

$$(V)_{ij} \;=\; v_j(\boldsymbol{x}_i) \tag{2.4}$$

- In one dimension, the *interpolation matrix* (aka *mass matrix* in the context of finite elements) V is guaranteed to be non-singular if
  - the basis functions $v_j(\cdot)$ are linearly independent

  and
  - the interpolation nodes $\{\boldsymbol{x}_i\}$ are distinct.

This brings us to...

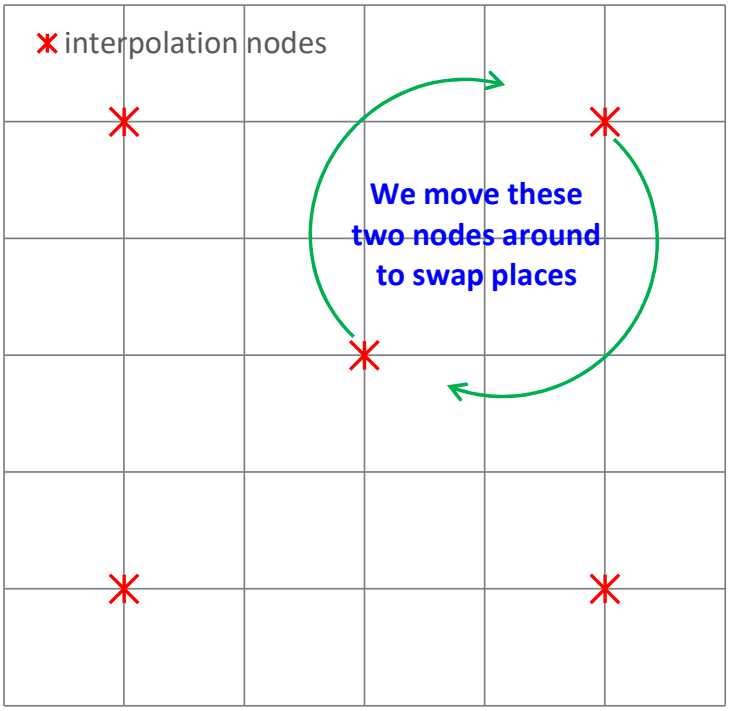
... the Mairhuber-Curtis theorem [Mai56, Cur58]:

*In two or more dimensions, a generic interpolant **leads to a singular system** for infinitely many configurations of interpolation nodes [FF15b].*

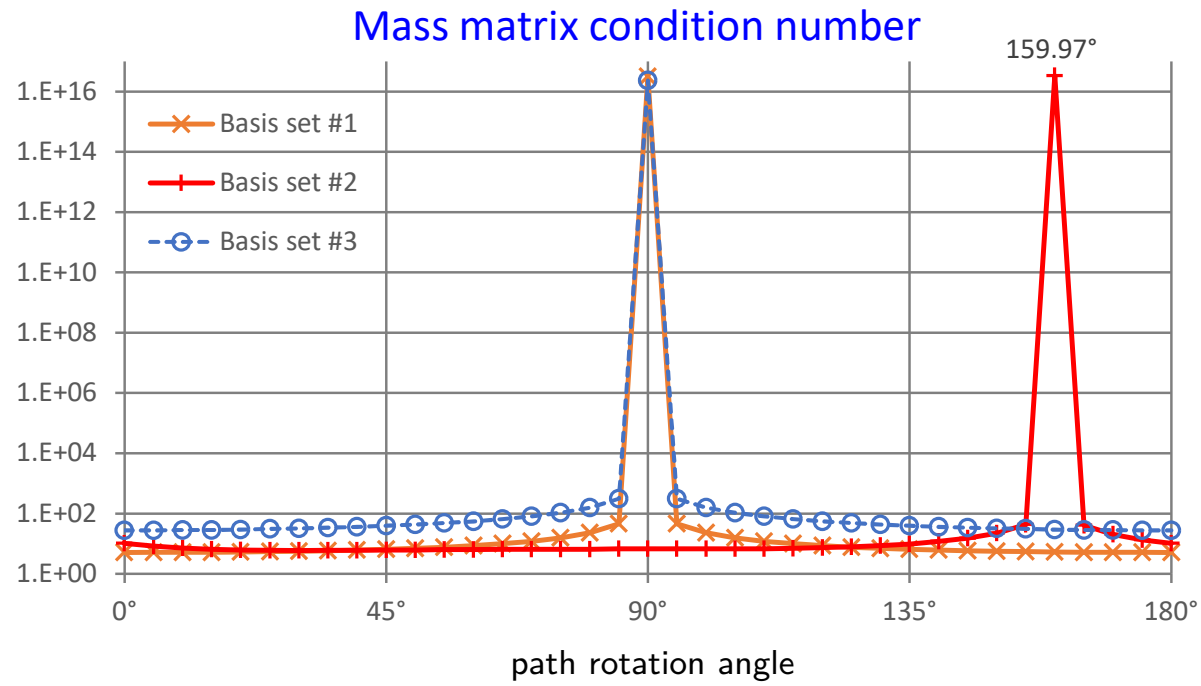
- For any basis function set, the i-th row of the interpolation matrix V is uniquely determined by the interpolation node $\boldsymbol{x}_i$.
- Exchanging rows of a matrix switches the sign of its determinant.
- In one dimension, in a gradual continuous move of adjacent interpolation nodes, we can only exchange them via an intermediate configuration when they coincide.
- The collision point is also the point when the interpolation matrix has two identical rows, and thus a zero determinant (and thus is singular), but not before, and not after.

Here's the kicker...

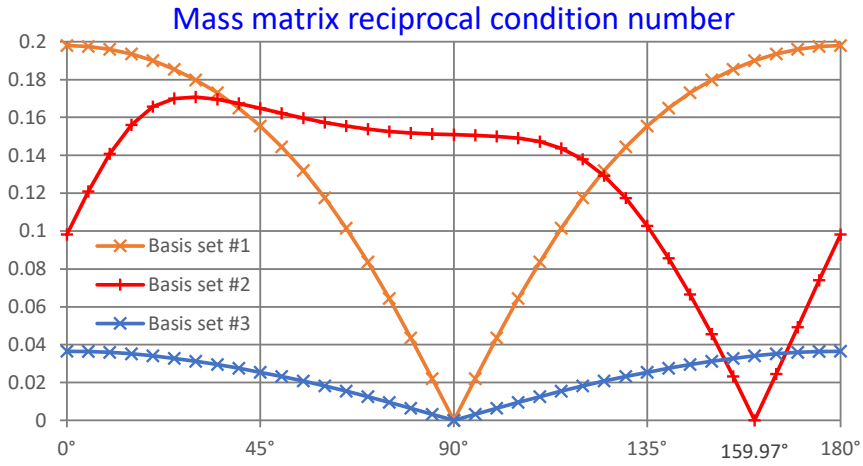
- In two or more dimensions, we can exchange interpolation nodes via paths ***that do not intersect***.
- Example in 2D:



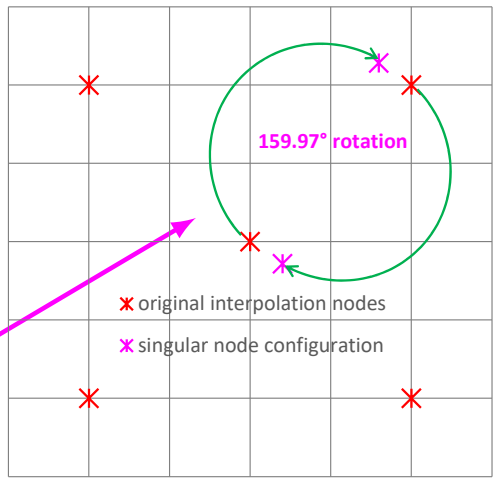
Taking 3 basis function sets, we compute the matrix condition number as a function of the rotation of the designated two *interpolation* nodes around their common midpoint:



Often it is more instructive to view the *reciprocal* condition number:



Basis set		singularity
#1	$1, x, y, x \cdot y, x^2 + y^2$	obvious at 90° by symmetry
#2	$1, x, y, x^2, y^2$	unexpected at 159.97°
#3	$\sqrt{1 + \lvert \boldsymbol{x}_j - \boldsymbol{r}_j \rvert^2}$	obvious at 90° by symmetry



$\{\boldsymbol{r}_j\}$ are the original (unrotated) interpolation nodes

However, one can design interpolation systems (in high dimensions) that *are always regular*.

Crucially, *the basis functions must depend on the interpolation data!*

$$v^\star(t, \boldsymbol{x}) = \sum_{s=1}^{m} c_s \cdot \psi_s(\boldsymbol{x}) \tag{2.5}$$

REPEAT:

Not just the interpolation coefficients, but also the basis must depend on the data!

- In 1979, R. Franke produced a 380(!) page technical report comparing 29(!) different surface interpolation methods [Fra79].

  *The most impressive method in these tests is the **multiquadric method** of Hardy [Har71]. It is consistently best or near best in terms of accuracy, and always results in visually pleasant surfaces.*

- ***Multiquadrics*** are hyperboloid radial basis functions of the form

$$\phi_j(\boldsymbol{x}) = \sqrt{1 + \epsilon^2 |\boldsymbol{x} - \boldsymbol{r}_j|^2} \tag{2.6}$$

[ϵ is the *shape parameter*; $\boldsymbol{r}_j$ is the *centre* or *basis node*]

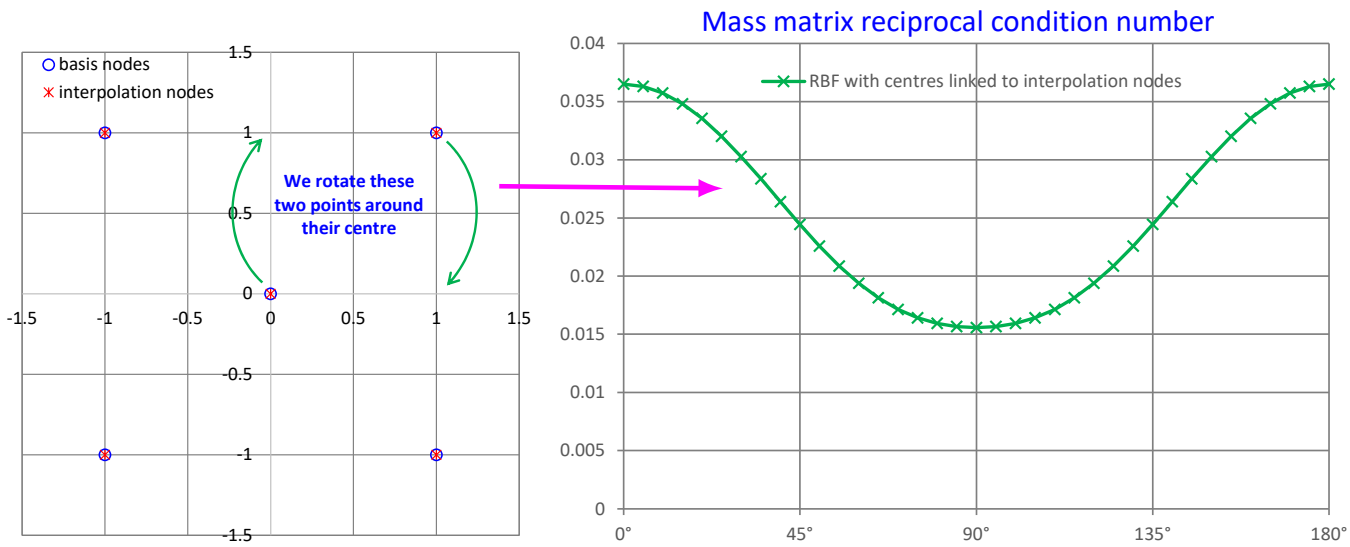
- In 1984, Micchelli [Mic84] proved that the linear system to compute the interpolation coefficients for the "MultiQuadric Surface" (MQS) method [as it was then known] is always regular (in full arithmetic precision).

- The radial nature of the multiquadric hyperboloids subsequently led to a flurry of developments of

Radial Basis Functions.

- The beauty of all this is that the input data can be arbitrarily scattered!

Indeed, if we repeat the continuous-path-interpolation-node-exchange experiment in 2D:



No singularities!

Radial Basis Functions

A radial basis function decomposition for a function $f(\boldsymbol{x}) : \mathbb{R}^d \to \mathbb{R}$ is typically defined by

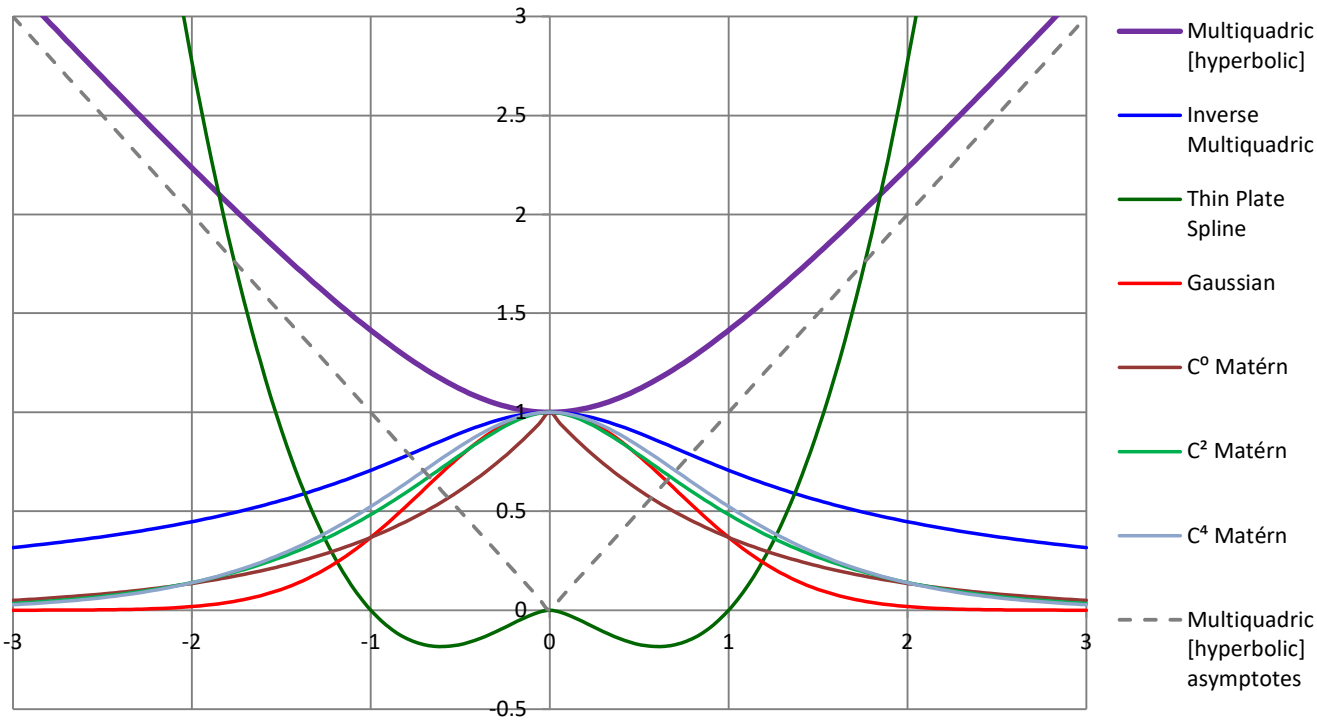
$$f^\star(\boldsymbol{x}) = \sum_{j=1}^{m} \lambda_j \cdot \phi(|\boldsymbol{x} - \boldsymbol{x}_j|; \epsilon_j) \tag{3.1}$$

with $\phi(r; \epsilon)$ being one of:-

Multiquadric [Har71]	$\sqrt{1 + (\epsilon r)^2}$ [aka *hyperboloid*]
Inverse Multiquadric [Har71]	$1/\sqrt{1 + (\epsilon r)^2}$
Gaussian	$\mathrm{e}^{-(\epsilon r)^2}$
Thin Plate Spline	$(\epsilon r)^2 \ln(\lvert\epsilon r\rvert)$
C^0 Matérn	$\mathrm{e}^{-\lvert\epsilon r\rvert}$
C^2 Matérn	$\mathrm{e}^{-\sqrt{3}\lvert\epsilon r\rvert} \cdot (1 + \sqrt{3}\lvert\epsilon r\rvert)$
C^4 Matérn	$\mathrm{e}^{-\sqrt{5}\lvert\epsilon r\rvert} \cdot (1 + \sqrt{5}\lvert\epsilon r\rvert + 5(\epsilon r)^2/3)$

In stark contrast to *finite elements*, radial basis functions are not necessarily locally concentrated functions.

This is particularly true for the so praised *multiquadric* basis:



One of the amazing properties of multiquadrics (and inverse multiquadrics) is that, under certain technical conditions on $f(x)$, in one dimension, RBF-interpolation on a regular grid with node distance h converges to $f(x)$

exponentially.

That's

$$|f(x) - f^*(x)| \sim \mathcal{O}\left(\mathrm{e}^{-c/h}\right) \tag{3.2}$$

which is obviously much faster than any power convergence where

$$|f(x) - f^*(x)| \sim \mathcal{O}\left(h^p\right) \tag{3.3}$$

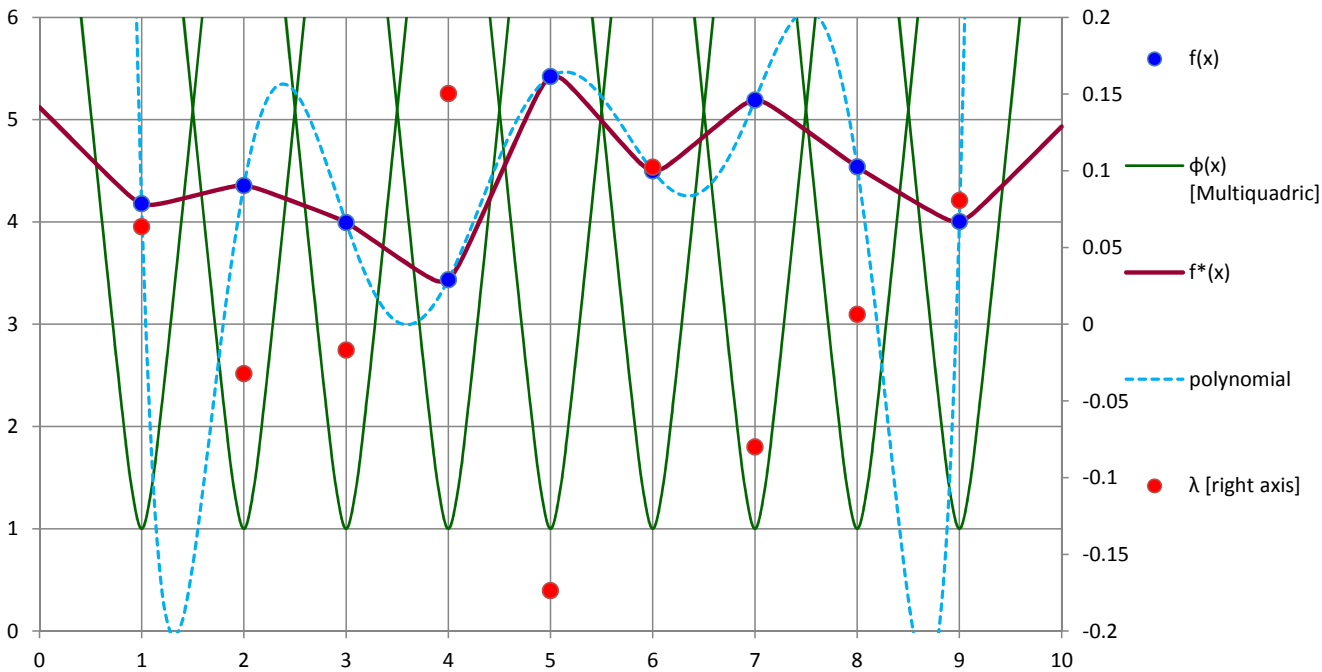
for any p.

This result was derived more than twenty years after Hardy picked them out as the best practical choice for scattered data interpolation!

What's more, "*Madych and Nelson showed that for the space of conditionally positive definite functions to which MQ belongs, a semi-norm exists and is minimized by such functions*" [Kan90b].

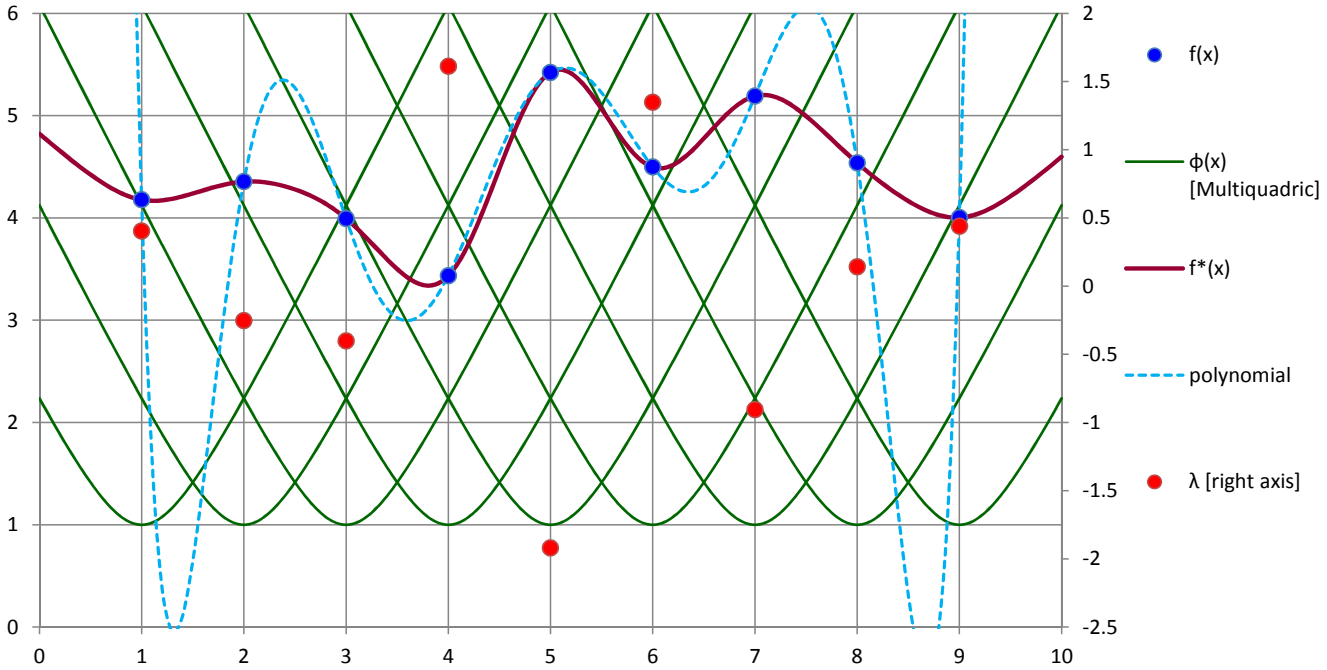
RBF-interpolation converges to *polynomial interpolation* as $\epsilon \to 0$.

$\epsilon = 10$:



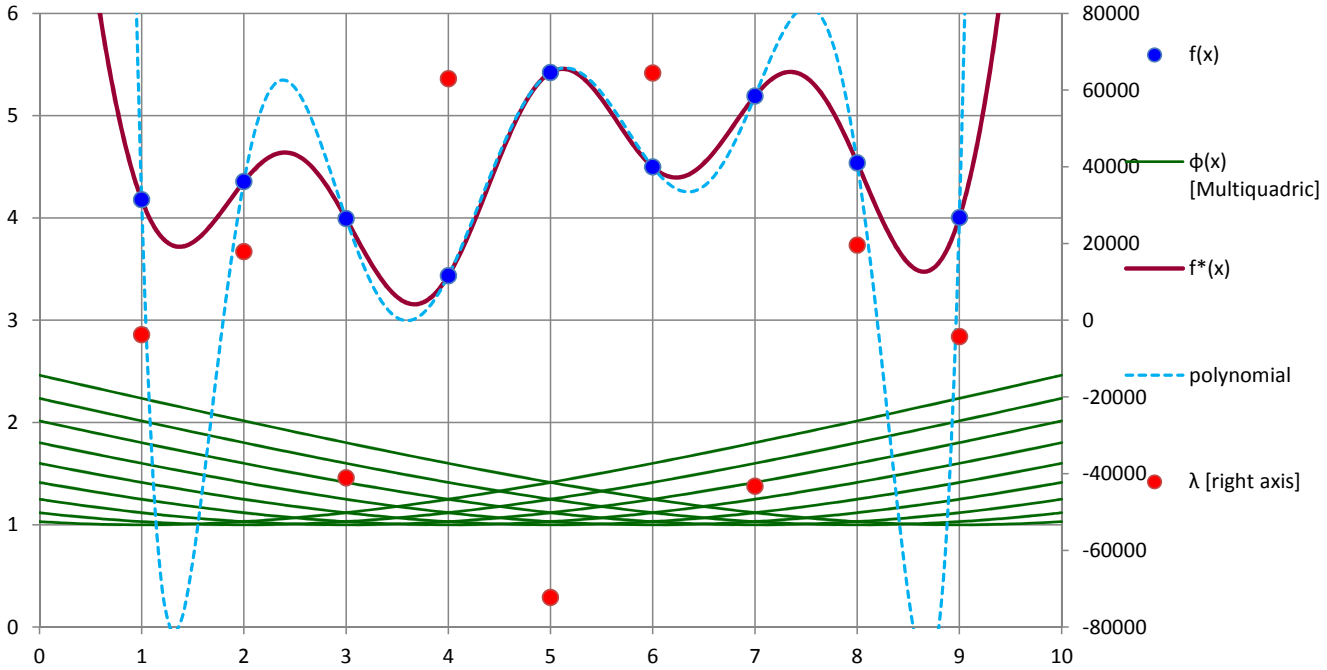
Reciprocal condition number $\kappa(\Phi)^{-1} = 0.0152$.

$\epsilon = 2$:



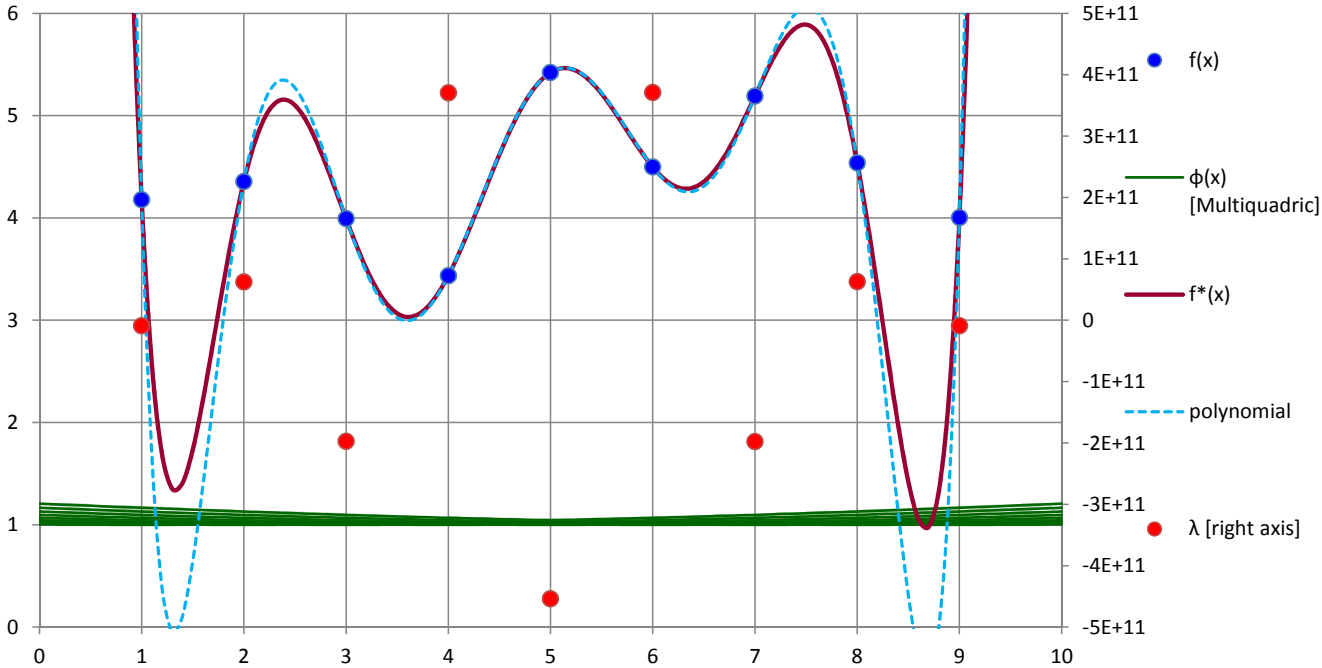
Reciprocal condition number $\kappa(\Phi)^{-1} = 0.0061$.

$\epsilon = 0.25$:



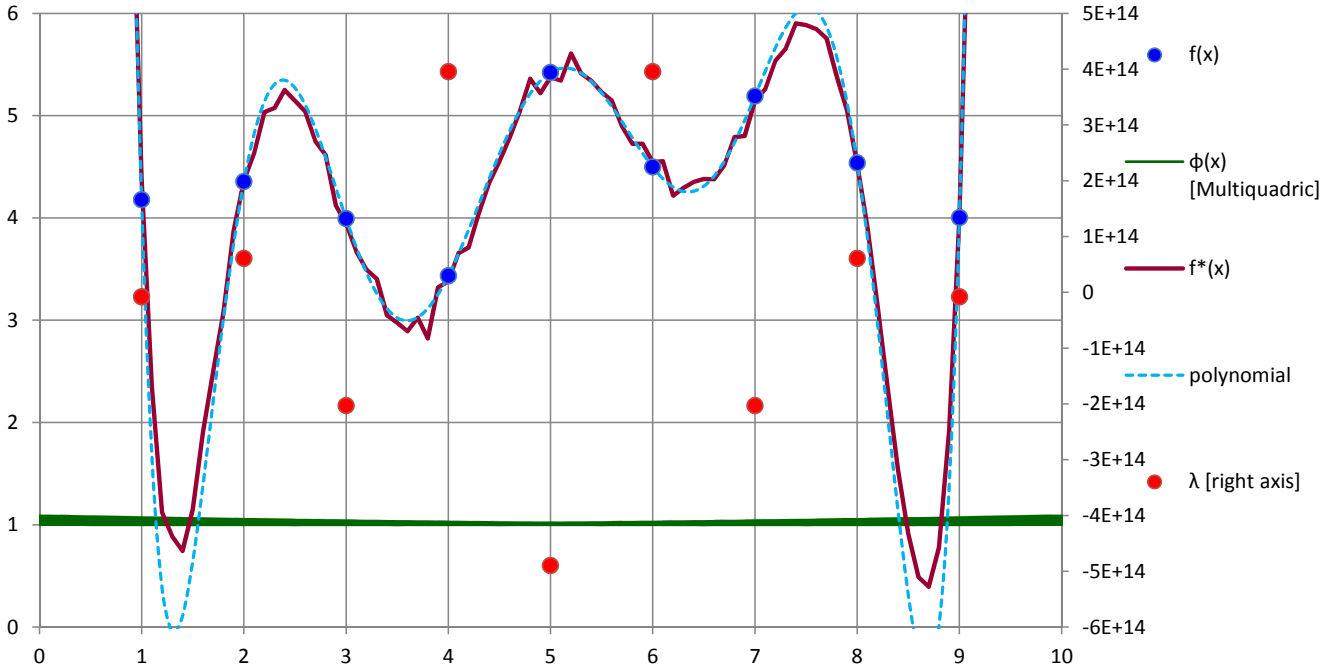
Reciprocal condition number $\kappa(\Phi)^{-1} = 6.65\text{E-}07$.

$\epsilon = 0.075$:



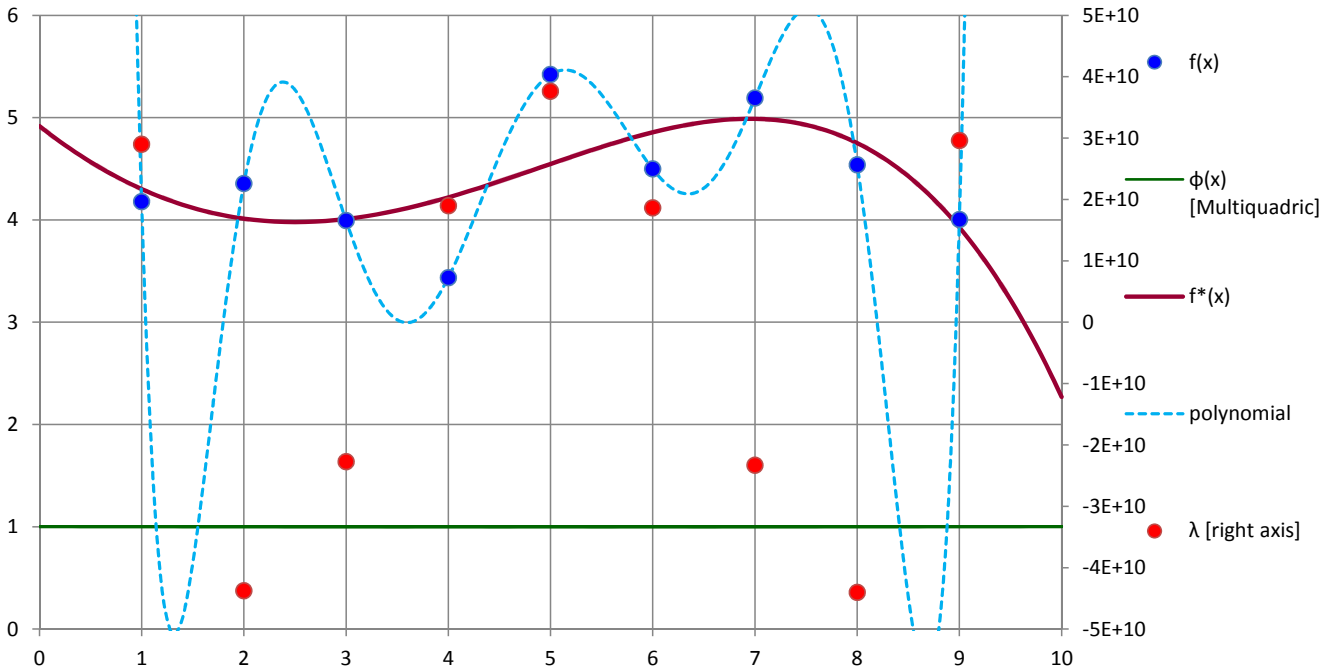
Reciprocal condition number $\kappa(\Phi)^{-1} = 1.61\text{E-}13$.

$\epsilon = 0.0475$:



Reciprocal condition number $\kappa(\Phi)^{-1} = 1.55\text{E-}16$.

$\epsilon = 0.001$:



Reciprocal condition number $\kappa(\Phi)^{-1} = 1.46\text{E-}18$. SVD cut-off at 2.23E-16.

- The limit of the *shape parameter* $\epsilon \to 0$ is known as the ***flat limit***. It is often desirable since it (typically) approximates the "smoothest" solution similarly to how it converges to the polynomial solution in the 1D case.
- In the flat limit ($\epsilon \to 0$), all terms like $\sqrt{1+\epsilon^2 r^2}$ or $\mathrm{e}^{-\epsilon^2 r^2}$ go to 1, rendering the interpolation matrix as a "flat" array of 1s.
- We saw how the interpolation becomes ragged as we approach this limit.
- Once $\kappa(\Phi)^{-1}$ approaches or drops below the machine's floating point resolution, the solution incurs more and more truncation-induced noise.
- Once again, ***Subtractive Cancellation*** raises its ugly head... 
- One should (at least) use a Moore-Penrose (SVD) solution[6].

[6] e.g., via Lapack's "Divide-and-Conquer" algorithm (DGELSD)

- Fornberg and Flyer [FF15b] gave some concrete analytical examples where the matrix condition number dependence on ϵ is as bad as $\mathcal{O}(\epsilon^{-84})$.
- Truncation (via SVD) aside, actual remedial approaches include:-
  - "*The most straightforward approach for calculating in the small ϵ regime is to use extended precision arithmetic.*" [FF15b].
    Alas, due to how the condition number scales as a power of ϵ, this quickly leads to the need for hundreds of digits. Not *directly* practical[7].
  - The "Padé-Contour" method [FWL04]. Complex (literally, by taking ϵ into the complex plane), involving many evaluations of the linear system solution+interpolation, (semi-)analytical expansions etc.
  - RBF-QR [FP07, FF15b]: depends on the geometry, only up to 3D (afaik). Think *Spherical Harmonics* decomposition ...
- There is plenty more to say on RBFs — *please also have a look at my 2017/18 presentation on Cluster Induction* [Jäc17].

[7] An alternative approach is to represent all numbers like $1+\nu$ with *tiny* ν [and those resulting from their arithmetic] as $(\frac{m+p}{n+q})$ ["*FloatFractions*"] where $m, n \in \mathbb{Z}$ and $p, q \in \mathbb{R}$ via operator overloading.

Meshless induction

Kansa [Kan90a] demonstrated that, apart from **serving well to interpolate a value surface** over scattered data, radial basis function decompositions are **also useful to approximate partial derivatives** of the value surface.

Next [Kan90b], given an advection-diffusion PDE

$$\partial_t f + \mathcal{L} \cdot f = 0 \tag{4.1}$$

substitute the decomposition

$$f(x) = \boldsymbol{\phi}(x)^\top \cdot \boldsymbol{\lambda} \tag{4.2}$$

for f, where $\boldsymbol{\phi}(x)$ is a vector of radial basis functions

centered in an arbitrarily scattered cluster of vertices.

This gives us

$$\boldsymbol{\phi}(\boldsymbol{x})^\top \cdot \dot{\boldsymbol{\lambda}}(t) + (\mathcal{L} \cdot \boldsymbol{\phi}(\boldsymbol{x}))^\top \cdot \boldsymbol{\lambda}(t) \;=\; 0\,. \tag{4.3}$$

with time-dependent weights $\boldsymbol{\lambda}(t)$.

Note that $\mathcal{L}$ applies ***analytically*** to the individual basis functions $\phi_i(x)$!

Denoting the result of the analytical evaluation of $\mathcal{L}$ applied to $\phi(\boldsymbol{x})$ as

$$\mathcal{L}\phi(\boldsymbol{x}) \;:=\; \mathcal{L} \cdot \phi(\boldsymbol{x}) \tag{4.4}$$

we now have

$$\boldsymbol{\phi}(\boldsymbol{x})^\top \cdot \dot{\boldsymbol{\lambda}}(t) + \mathcal{L}\phi(\boldsymbol{x})^\top \cdot \boldsymbol{\lambda}(t) \;=\; 0\,, \tag{4.5}$$

i.e., a[8]

"clustered" ordinary differential equation in $\boldsymbol{\lambda}(t)$.

Note that $\boldsymbol{x}$ takes on the role of a parameter vector!

[8] sometimes also referred to as *meshless method of lines* ["MOL"]

In order to meet Micchelli's non-singularity criterion, we ask for the cluster ODE to hold for $\boldsymbol{x}$ to be in any of the cluster vertices $\{\boldsymbol{x}_i\}\,\forall\, i = 1, \ldots, n$ where the radial basis functions are centered.

Defining the matrices Φ and $\mathcal{L}\Phi$ whose elements are

$$(\Phi)_{ij} \;=\; \phi(|\boldsymbol{x}_i - \boldsymbol{x}_j|; \epsilon_j) \tag{4.6}$$

$$(\mathcal{L}\Phi)_{ij} \;=\; \mathcal{L} \cdot \phi(|\boldsymbol{x} - \boldsymbol{x}_j|; \epsilon_j)\big|_{\boldsymbol{x}=\boldsymbol{x}_i}\,, \tag{4.7}$$

we obtain the cluster ODE system

$$\Phi \cdot \dot{\boldsymbol{\lambda}}(t) + \mathcal{L}\Phi \cdot \boldsymbol{\lambda}(t) \;=\; 0\,. \tag{4.8}$$

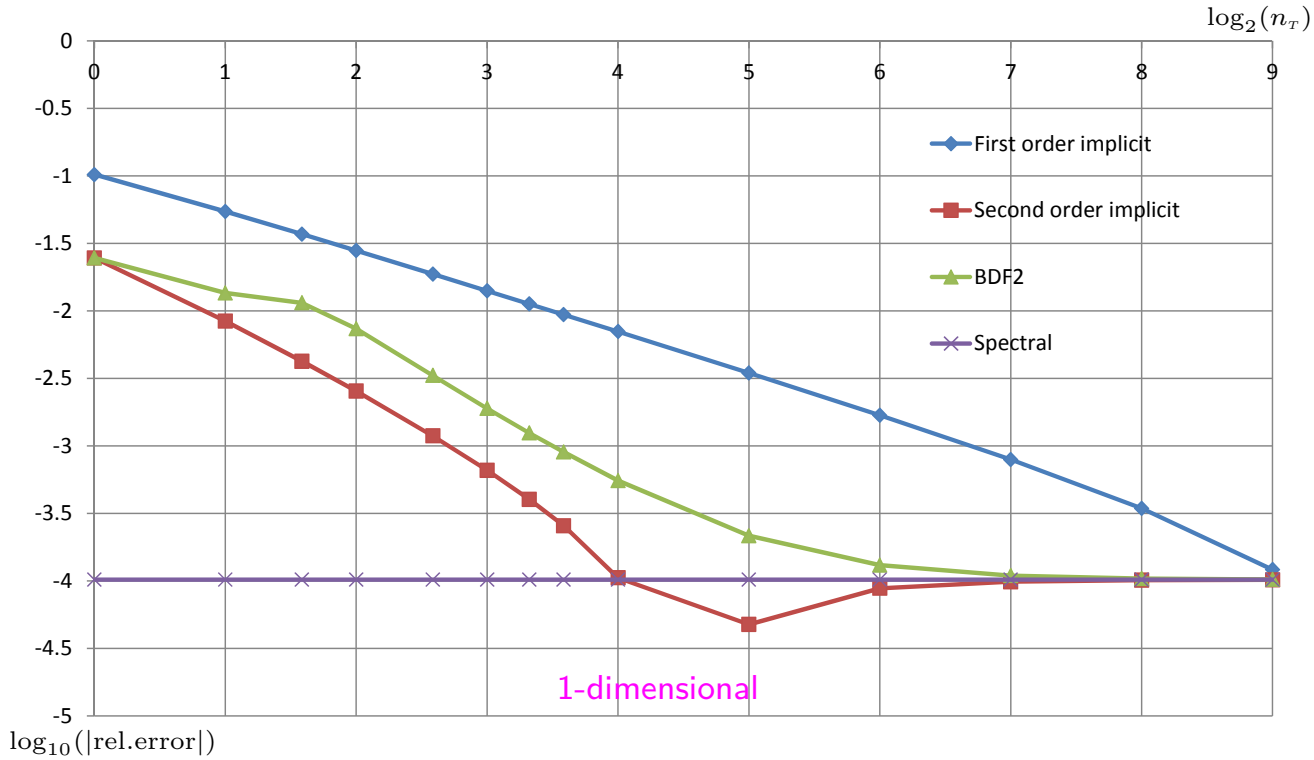
REMARK. It is in principle possible to have more or fewer basis functions than locations at which we ask for the original cluster ODE (4.5) to hold, or to choose locations different from the basis centres. Either way, we will then demand for the system (4.8) to be best matched in a least squares sense, naturally leading to the use of Singular Value Decomposition for its solution.

At this point, I will skip a significant amount of technical details such as:-

- analytical expressions for the matrix elements of $\mathcal{L}\Phi$ for some RBF types, for, say, the usual advection-diffusion generators of affine models;
- what numerical integration schemes are suitable for backward (or forward) induction in the time direction and their stability features / the spectrum of the resulting discrete-time-step propagator matrices;
- useful spatial coordinate transformations that compensate for the typical $\sim \sqrt{t}$ growth of our domain of interest in financial applications;
- other useful features such as the addition of a constant and linear basis functions (taking care of essentially linear instruments);
- the impact of heterogenous ϵ_j [FZ07] and how to choose it (I typically set it as the reciprocal of the nearest cluster neighbour distance !);

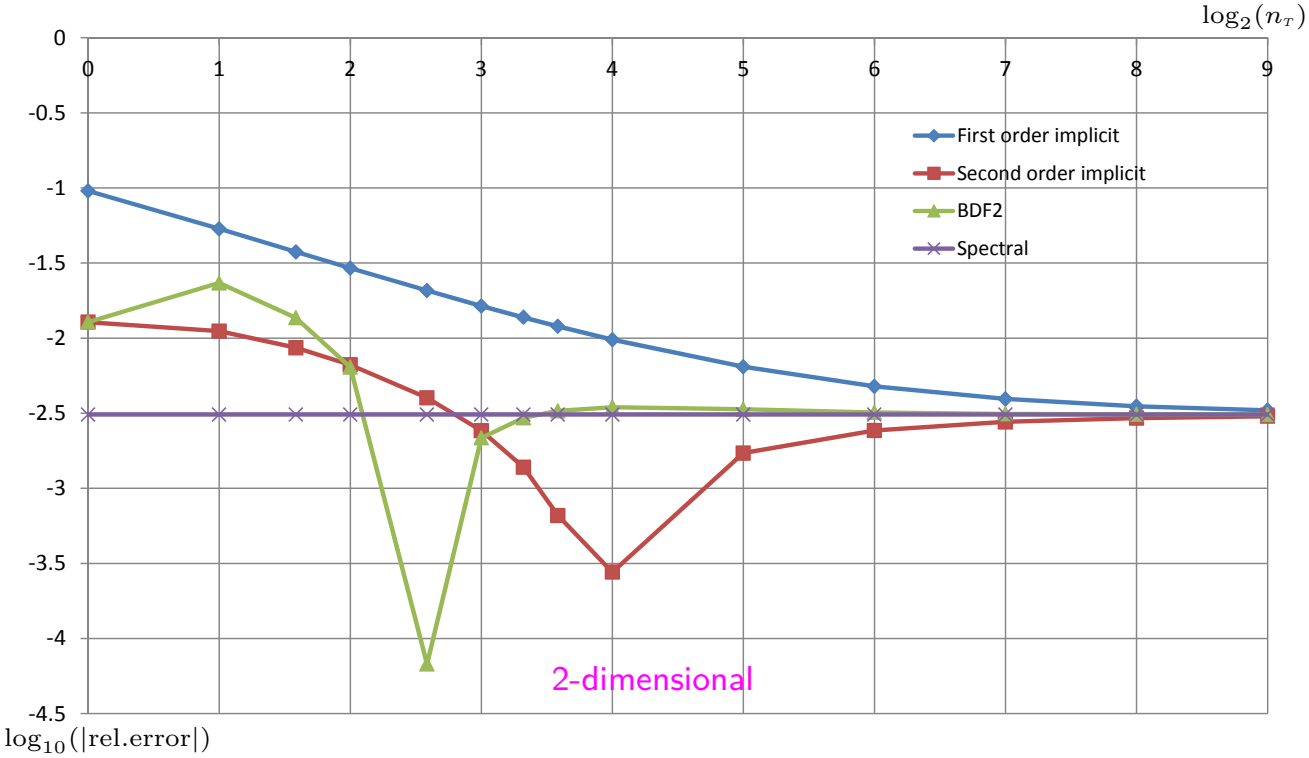
Please see my 2017/18 presentation on Cluster Induction [Jäc17] for more information on these points.

Vanilla 1Y ATM FX option with Black-Scholes and $n_{\text{cluster}} = 2^7 - 1 = 127$.

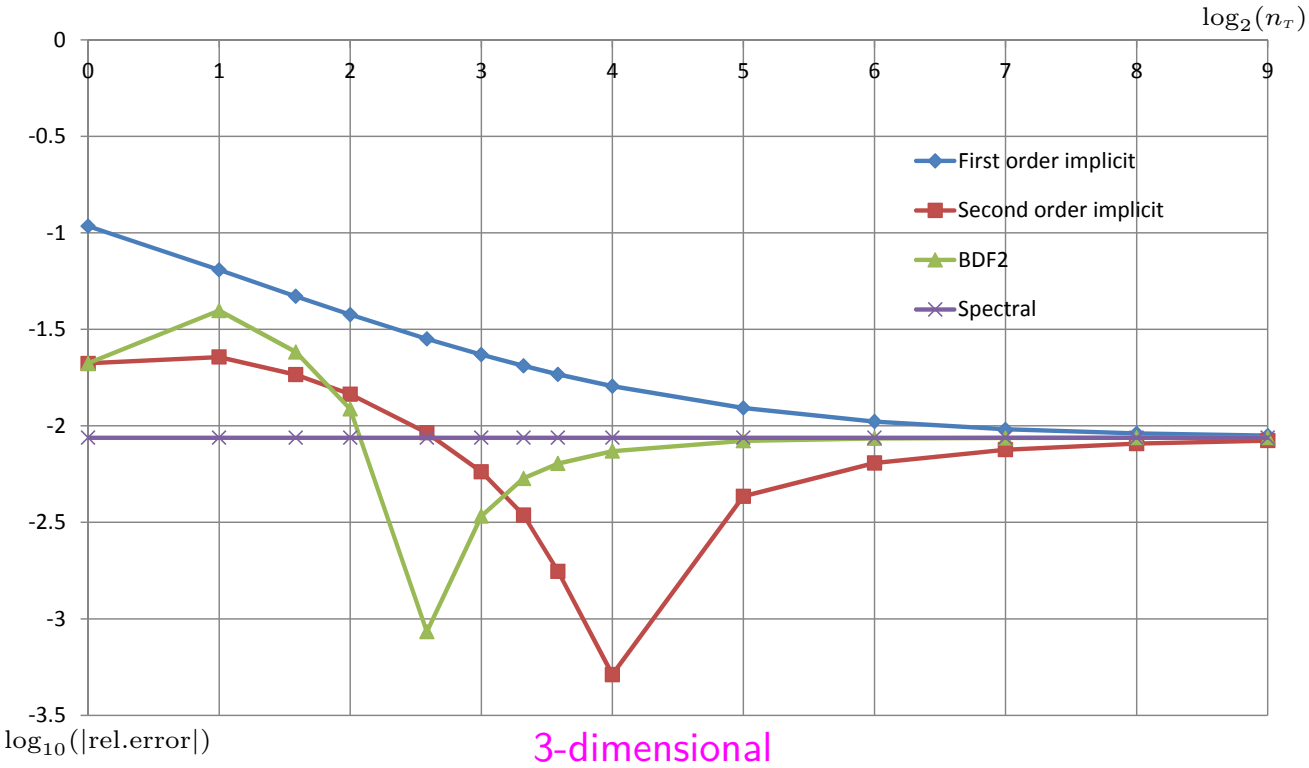


- See, e.g., [Jäc13], for the 2nd order implicit scheme. See [Jäc17] for all other parameters etc.
- The "Spectral" method is an approximation for the matrix exponential $\mathrm{e}^{t \cdot G}$.

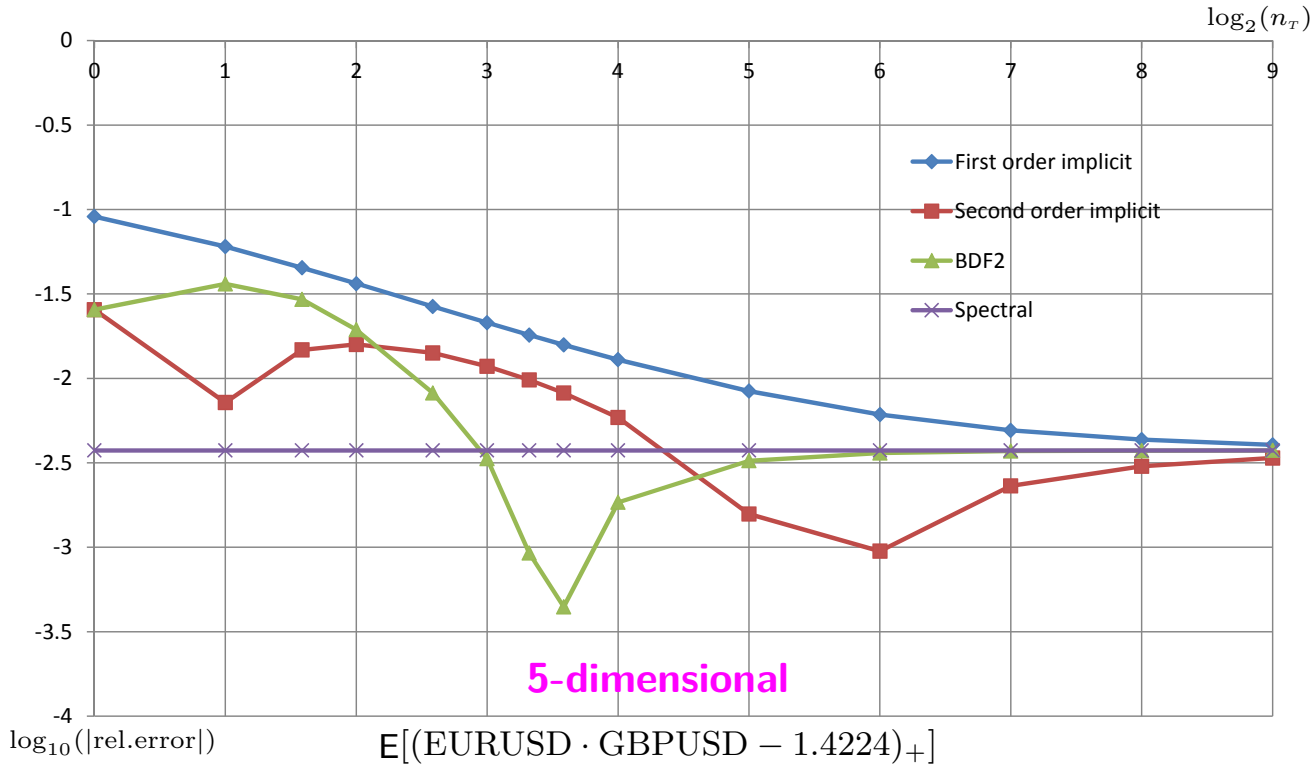
1Y ATM composite option with Black-Scholes and $n_{\text{cluster}} = 2^8 - 1 = 255$.



1Y ATM FX option with Linear Gaussian Markov model and $n_{\text{cluster}} = 255$.



1Y composite option with Linear Gaussian Markov model and $n_{\text{cluster}} = 511$.



How do we generate the cluster?

- Sadly, most research is done in just 1 or 2 dimensions! (!!!)
- There is little in the literature for more than 2 dimensions, up to which researchers tend to use fairly regular point distributions.
- A transfer of ideas from finite element methods is appealing.
- Only that little is out there for more than 3 dimensions, and effective (adaptive) tesselations for FEM tend to be specialised for 2 dimensions.

Quote [FF15a]:

In 2-D: Quick to go from quasi-uniform nodes to well-balanced Delaunay triangularization.

In 3-D: Finding good tetrahedral sets can even become a dominant cost.

To cut a long line of experimental results short, this seems to work best:

- In d dimensions, draw Sobol' vectors and transform them from uniform to Gaussian coordinates via the inverse cumulative normal function.

➡ ***Gaussian Sobol' vectors***

- We use these as our *normalised coordinate* cluster nodes[9].
- However, we are not finished yet. Theory and experiments suggest that the final result depends a lot on the "*quality*" of the cluster.
- So, let's optimize the cluster *quality*.
- What *quality*?

[9] For details as to how these are/can be mapped to calculation coordinates, see [Jäc17].

Consider that each point $\boldsymbol{x} \in \mathbb{R}^d$ is represented by its nearest cluster node

$$\boldsymbol{x} \;\to\; \boldsymbol{\xi}_{\hat{k}(\boldsymbol{x};\mathcal{C})} \tag{5.1}$$

with

$$\hat{k}(\boldsymbol{x};\mathcal{C}) \;:=\; \arg\min_{k\in\{1,..,N\}} |\boldsymbol{x} - \boldsymbol{\xi}_k| \tag{5.2}$$

where N is the number of nodes in the given cluster $\mathcal{C} = \{\boldsymbol{\xi}_1, ..., \boldsymbol{\xi}_N\}$.

This is the classical definition of a ***quantization***.

Then, we call the expectation[10]

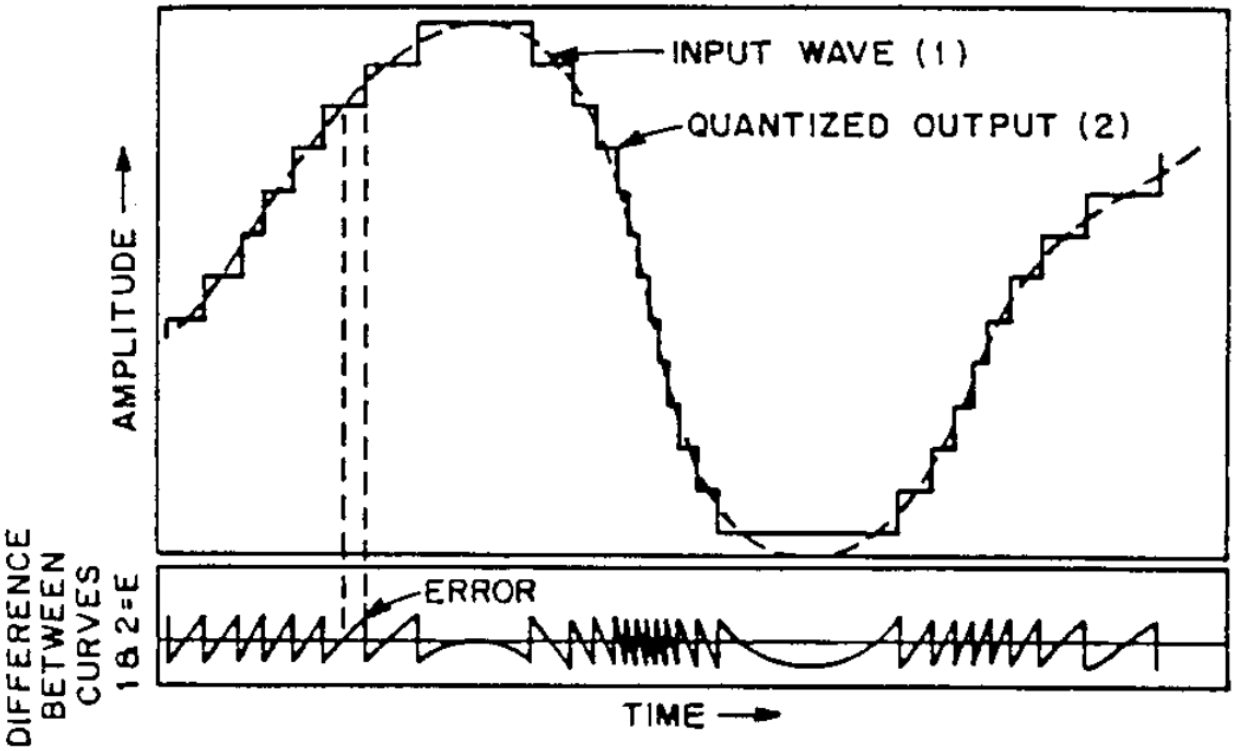
$$D(\mathcal{C}) \;:=\; \mathsf{E}\left[\left|\boldsymbol{x} - \boldsymbol{\xi}_{\hat{k}(\boldsymbol{x};\mathcal{C})}\right|^2\right] \tag{5.3}$$

the *distortion* [PD51] of $\mathcal{C}$'s representation of $\mathbb{R}^d$.

Since we operate with $\boldsymbol{x}$ to represent independent financial risk factors, we take the expectation under a multi-variate standard normal distribution for $\boldsymbol{x}$ (with zero correlation).

[10] This is the *mean square distortion* of Panter and Dite [PD51]. In other contexts, we may also find its square root defined as the *root mean square (RMS) distortion*.

An image says more than a thousand words to explain the name *"distortion"*:



- A classical result in statistics is that the distortion $D(\mathcal{C})$ is minimal when

$$\boldsymbol{\xi}_k \cdot \mathrm{E}\left[\mathbf{1}_{\{\hat{k}(\boldsymbol{x};\mathcal{C})=k\}}\right] = \mathrm{E}\left[\boldsymbol{x} \cdot \mathbf{1}_{\{\hat{k}(\boldsymbol{x};\mathcal{C})=k\}}\right] \tag{5.4}$$

  for all $k \in \{1, .., N\}$.

- This is to say that $\boldsymbol{\xi}_k$ must be equal to the first moment of $\boldsymbol{x}$ conditioned on the domain of all points in $\mathbb{R}^d$ whose nearest cluster node is $\boldsymbol{\xi}_k$.

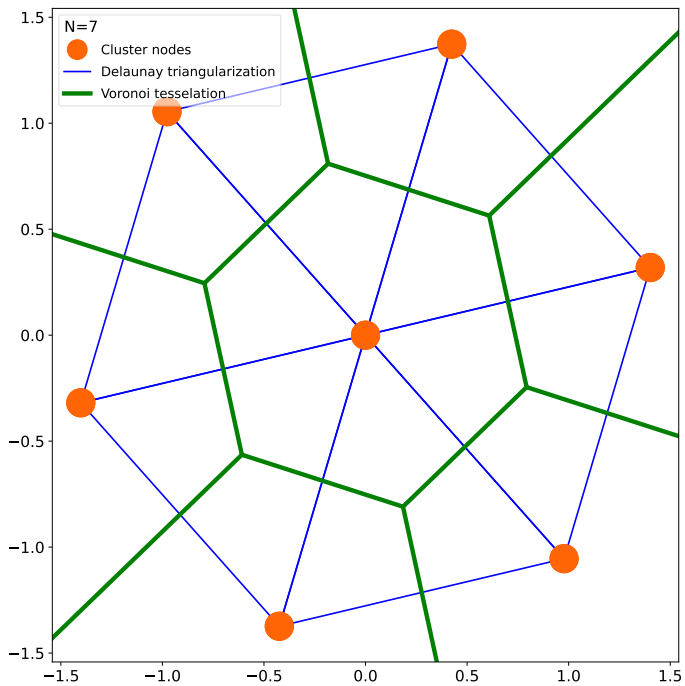
- We call the cluster $\mathcal{C}_d^*(N)$ that satisfies (5.4) the

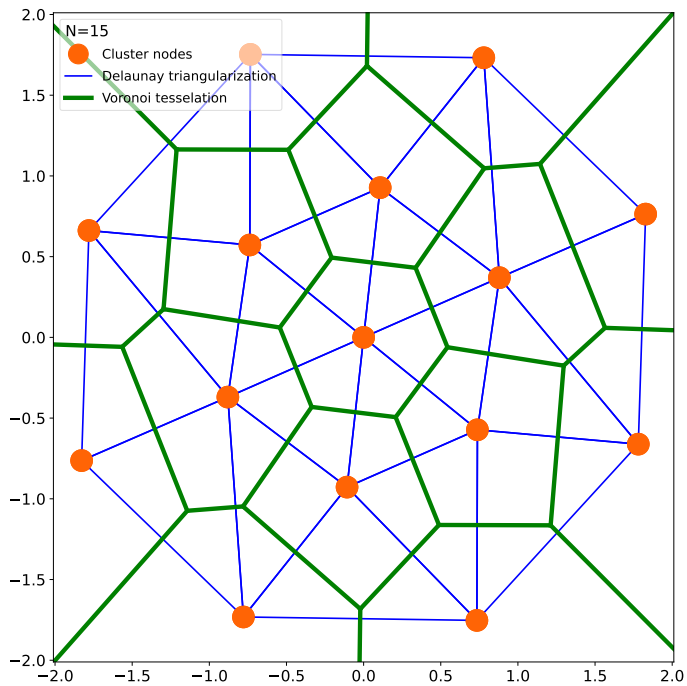
  Minimum Distortion Cluster

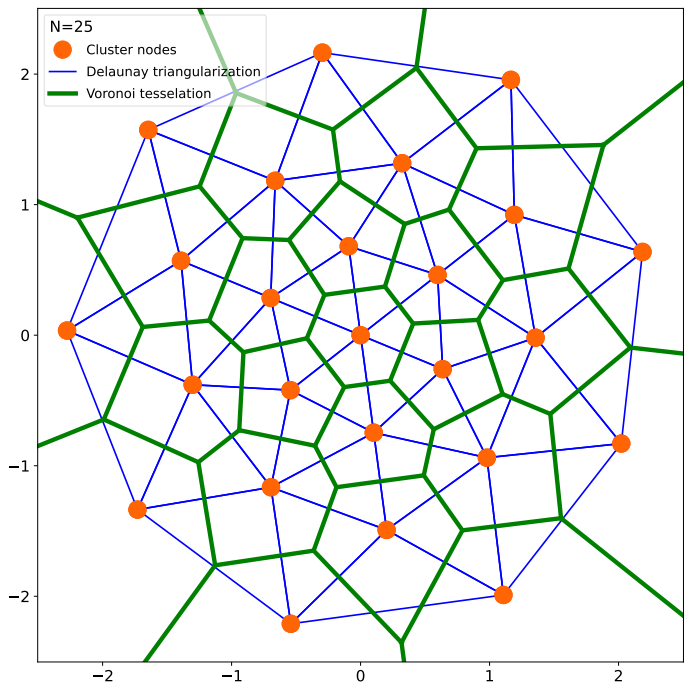
  of N nodes in d dimensions, though it is better known[11] as the

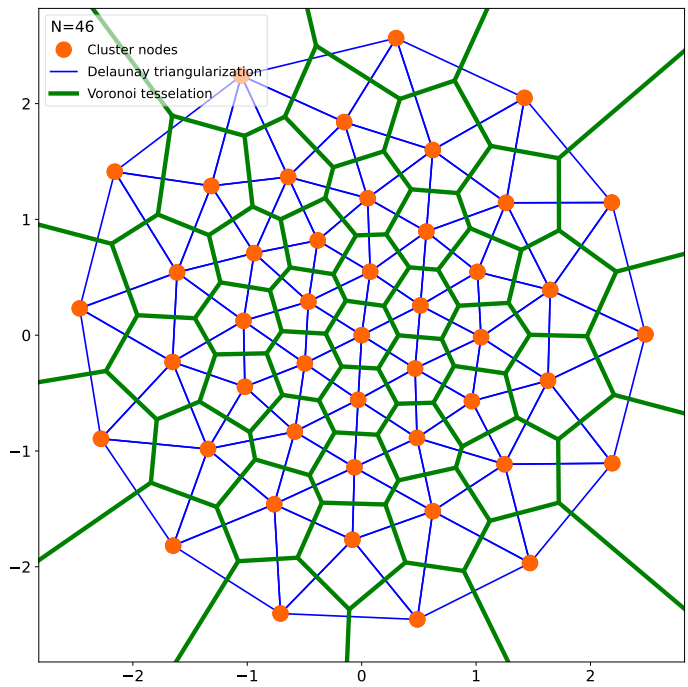
  Centroidal Voronoi Tesselation.

[11] Some authors in mathematical finance have unfortunately referred to such a choice of nodes simply as a *"quantization"* which is not in line with the rest of mathematics, physics, and engineering, where a *quantization* only means the choice of *any* representation of a continuum by a subset of discrete values, without any statement about the rule that led to the choice of nodes.


N=7
Cluster nodes
Delaunay triangularization
Voronoi tesselation


N=15
Cluster nodes
Delaunay triangularization
Voronoi tesselation


N=25
Cluster nodes
Delaunay triangularization
Voronoi tesselation


N=46
Cluster nodes
Delaunay triangularization
Voronoi tesselation

- Due to considerable geometrical complications, there is no competitive "analytical" method to compute $\mathcal{C}_d^*(N)$ when d is more than two(-ish).
- A simple algorithm attributed to Lloyd [Llo57] is to iterate

$$\boldsymbol{\xi}_k{}^{(j+1)} := \frac{\mathsf{E}\left[\boldsymbol{x}\cdot\mathbf{1}_{\left\{\hat{k}(\boldsymbol{x};\mathcal{C}^{(j)})=k\right\}}\right]}{\mathsf{E}\left[\mathbf{1}_{\left\{\hat{k}(\boldsymbol{x};\mathcal{C}^{(j)})=k\right\}}\right]} = \mathsf{E}\left[\boldsymbol{x}\,\middle|\,\hat{k}(\boldsymbol{x};\mathcal{C}^{(j)})=k\right]. \tag{5.5}$$

  This algorithm is also known as *Voronoi iteration* or *Voronoi relaxation*.

- Again, in more than two (or so) dimensions, it is not practical to compute the conditional expectations in (5.5) (semi-)analytically whence we resort to a **good old-fashioned Sobol'-Monte-Carlo evaluation**[12].

[12] Here, too, unfortunately, some authors have referred to the numerical evaluation of those conditional expectations by means of a sampling method as *"stochastic" gradient descent methods* despite the fact that none of the above has anything to do with *stochasticity* or *randomness* or the concept of any *[stochastic] process in time*.

The **Voronoi-Sobol' Iteration** algorithm in a nutshell:

- Precompute a d-dimensional sampling set of Gaussian Sobol'-Monte-Carlo draws, say M =32767 points, $\boldsymbol{x}_i$ for $i = 1, .., M$.
- Start with a d-dimensional initial cluster $\mathcal{C}^{(0)} = \left\{\boldsymbol{\xi}_k{}^{(0)}\right\}$ comprised by Gaussian Sobol'-Monte-Carlo draws of size $k = 1, .., N$.

Then, begin the *Outer Iteration*:-

- Zero-out a workspace of N vectors $\boldsymbol{\Xi}_k \in \mathbb{R}^d$ and counter variables m_k for $k = 1, .., N$.
- Do the *Inner Loop* over all of the M sampling points $\boldsymbol{x}_i$, for each to find the nearest (previous) cluster node $\boldsymbol{\xi}_{\hat{k}}{}^{(j)}$ with index $\hat{k} = \hat{k}(\boldsymbol{x}_i, \mathcal{C}^{(j)})$, and set

$$\boldsymbol{\Xi}_{\hat{k}} \mathrel{+}= \boldsymbol{x}_i \qquad \text{and} \qquad m_{\hat{k}} \mathrel{+}= 1\,.$$

- Upon completing the *Inner Loop* over the sampling points ($i = 1, .., M$), set

$$\boldsymbol{\xi}_k{}^{(j+1)} := \boldsymbol{\Xi}_k / m_k \qquad \text{for} \quad k = 1, .., N.$$

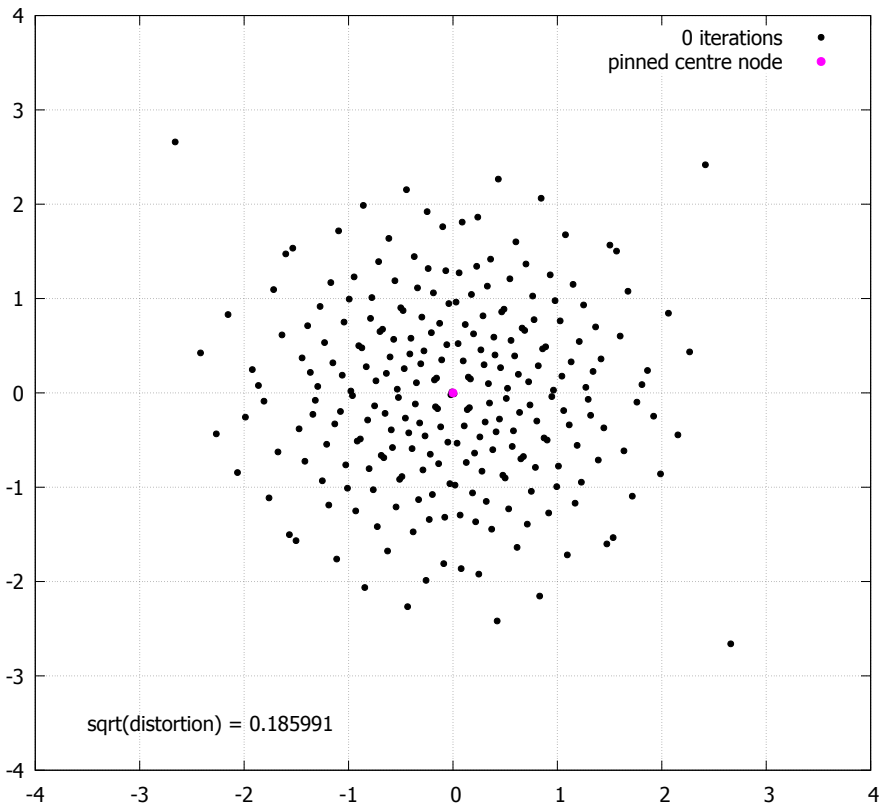
- Terminate the *Outer Iteration* if $\boldsymbol{\xi}_k{}^{(j+1)} \equiv \boldsymbol{\xi}_k{}^{(j)}\ \forall\, k = 1, .., N$,
  or if a certain number of maximum iterations has been reached (e.g., 127).

Notes:-

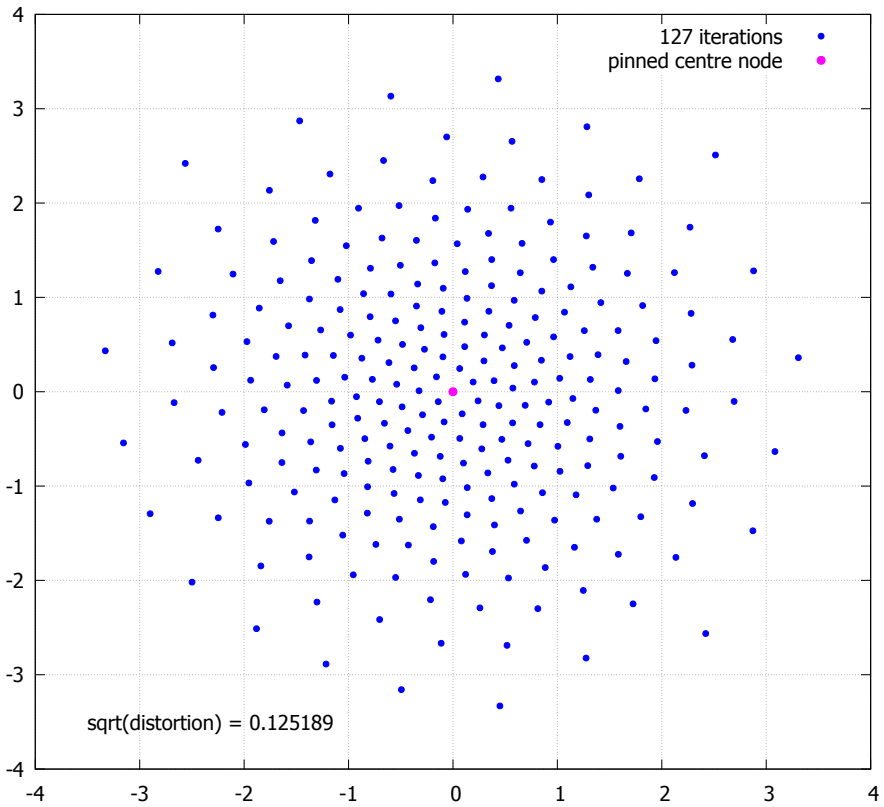
- The *Outer Iteration* (5.5) *may not converge (!)*. This is due to the discrete underlying sampling set used to evaluate the conditional expectation.
- **The above mentioned non-convergence is nothing to worry about!** It simply means that the algorithm eventually just cycles over a *discrete* set of equally good estimates for $\mathcal{C}^*$ for the given size N, dimensionality d, and Gaussian-Sobol' sampling set.
- Cache all computed clusters in memory for this run-time session since more computations are likely to want the same! Caching it mitigates the possibly considerable time it can take to create this cluster.
- A good small cluster ($N \leq 255$) wins over a large bad cluster.

- Grow the cluster N size gently with increasing dimensionality d.
  More on that later.
- In higher dimensions, such optimum-representation cluster computation techniques are part of a range of ***machine learning algorithms***, e.g., *"k-Means Clustering"*.
- Caveat emptor: you may find the oft-praised "Anderson acceleration" technique of limited use. Think *Richardson extrapolation*, c.f., [Bac22].
- Instead, I found the **Hamerly algorithm** [Ham10] of great benefit!
  It can also be multi-threaded.
- See Wikipedia [Llo18] for more acceleration techniques and for links how these methods are also used in finite element calculations.

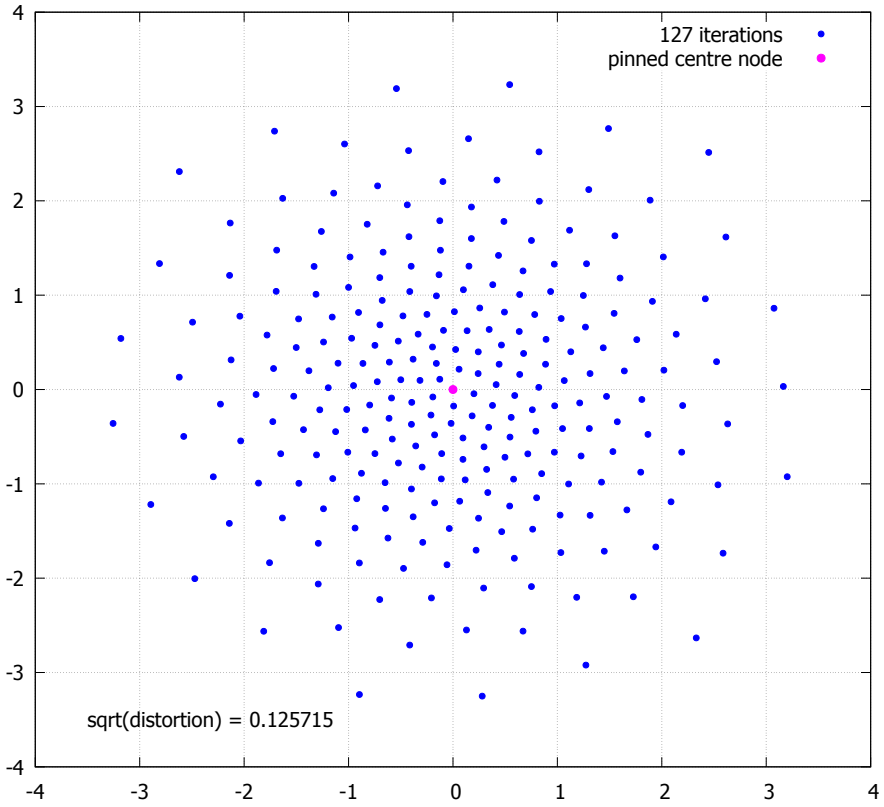
For $n_{\text{cluster}} = 255, n_{\text{sampling}} = 131071, n_{\text{dim}} = 2$, starting with:



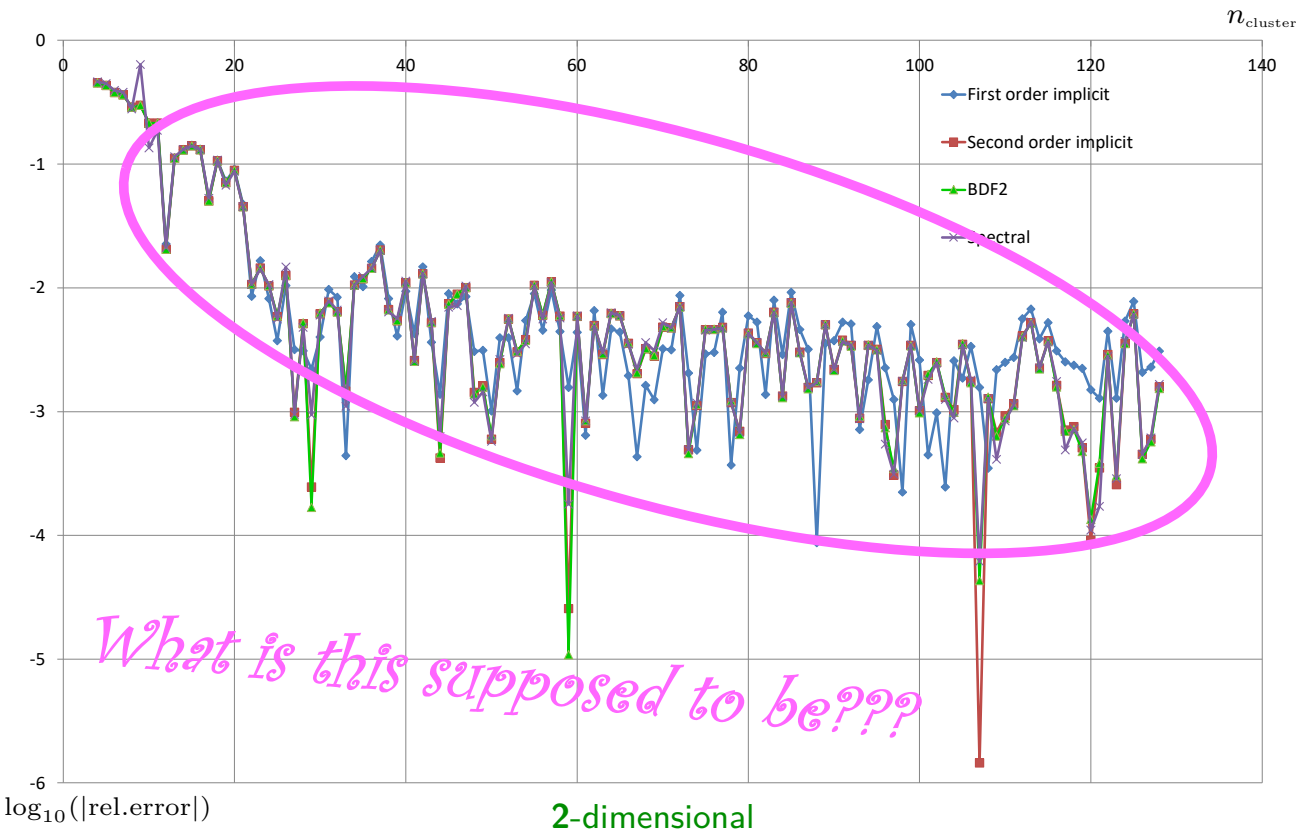
After 127 iterations, we arrive at:



Here with a different algorithm based on initial sub-sampling ($\sim 3\times$ faster):

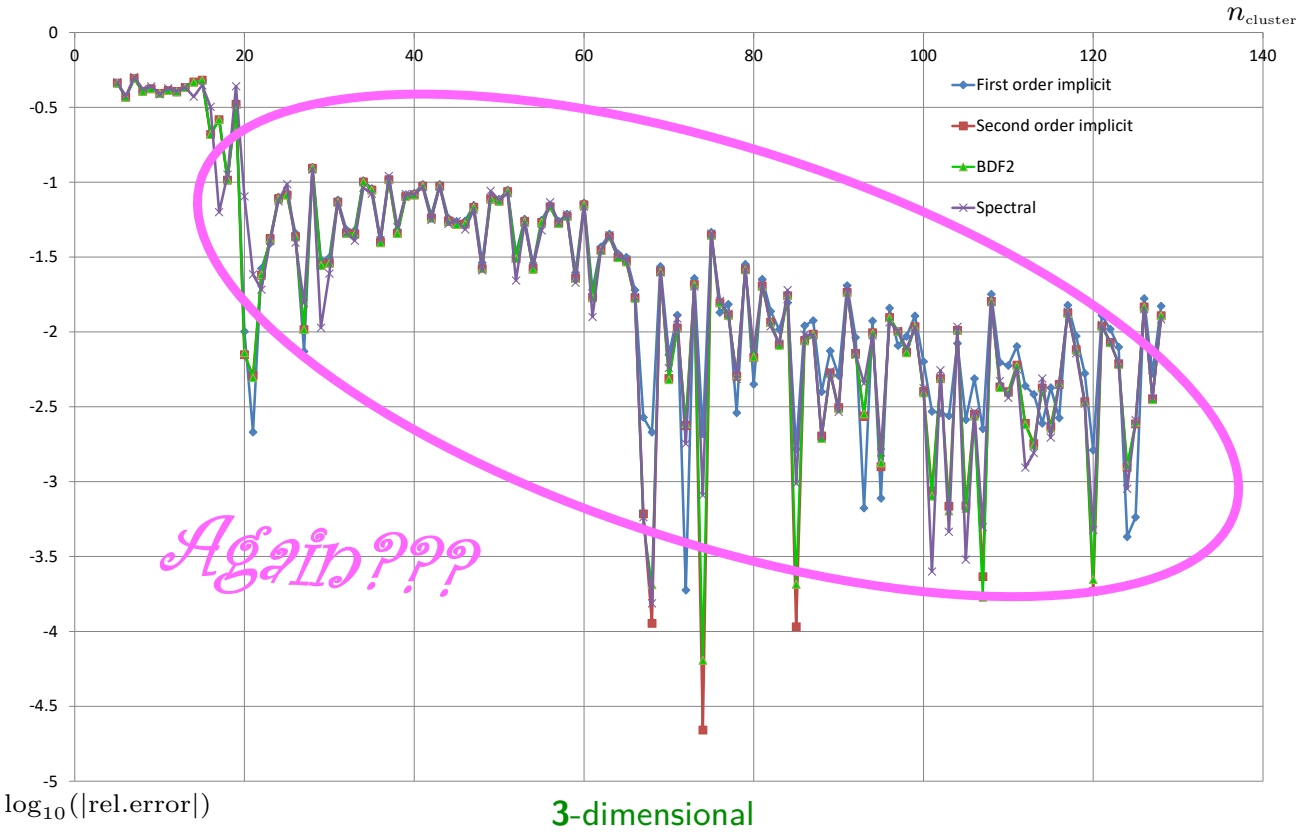


2D: 1Y ATM composite option with Black-Scholes and $n_T = 64$.

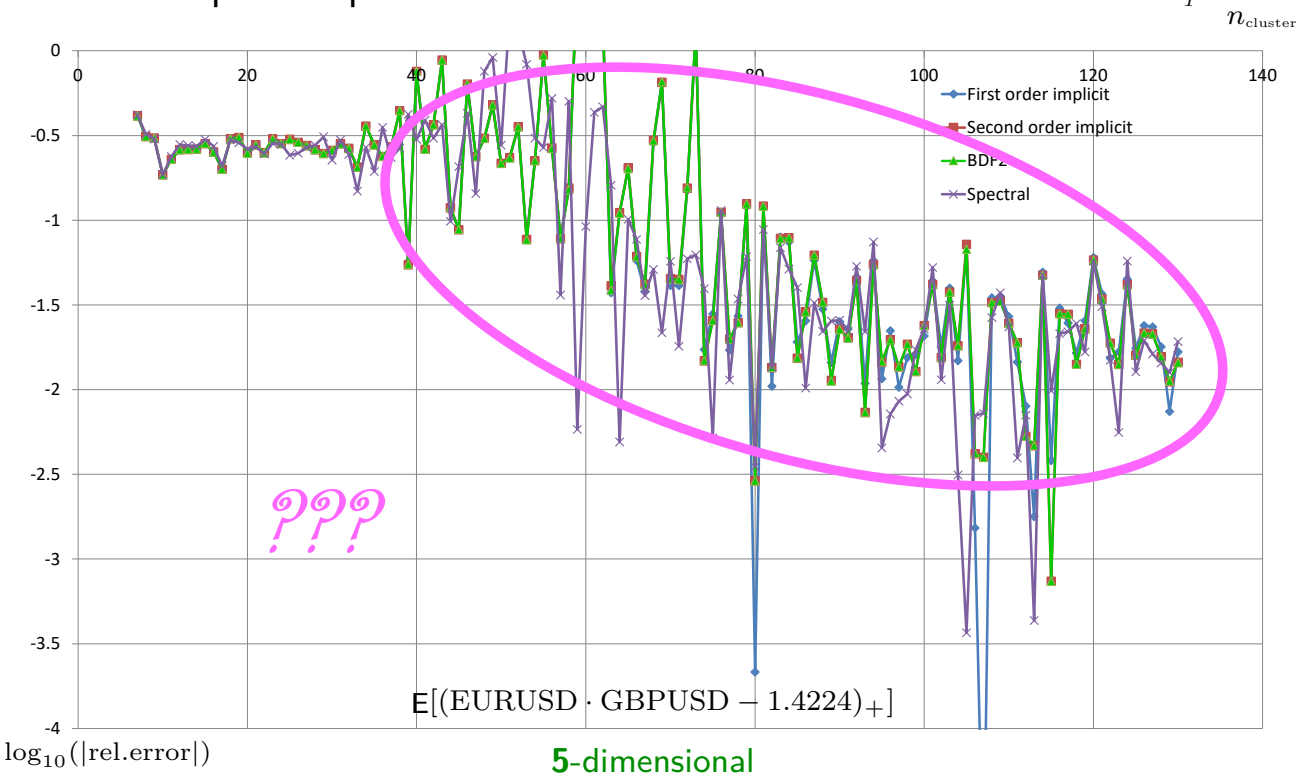


- All clusters used here and in the following are optimised via Voronoi iteration as described.

3D: 1Y ATM FX option with Linear Gaussian Markov model and $n_T = 64$.

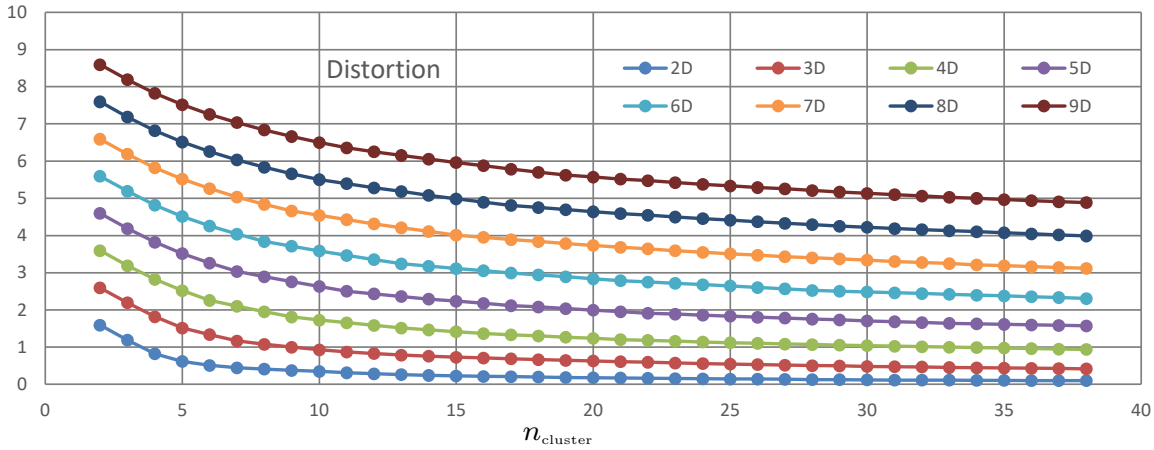


5D: 1Y composite option with Linear Gaussian Markov model and $n_T = 64$.



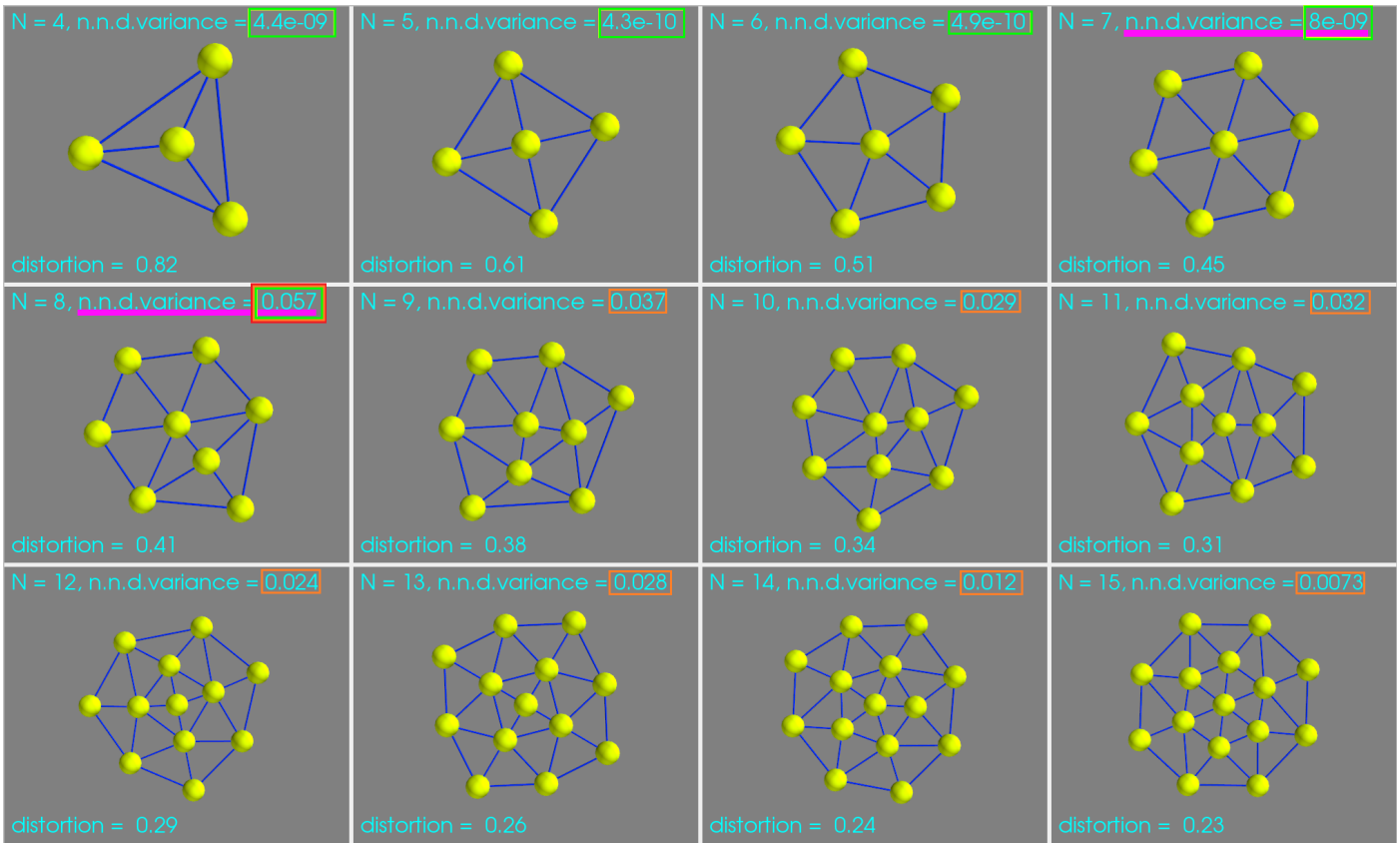
- This warrants a closer look.

- We notice that the onset of the "*irregular*" behaviour depends on the dimensionality.
- Distortion doesn't give us any signal as to what's going on:

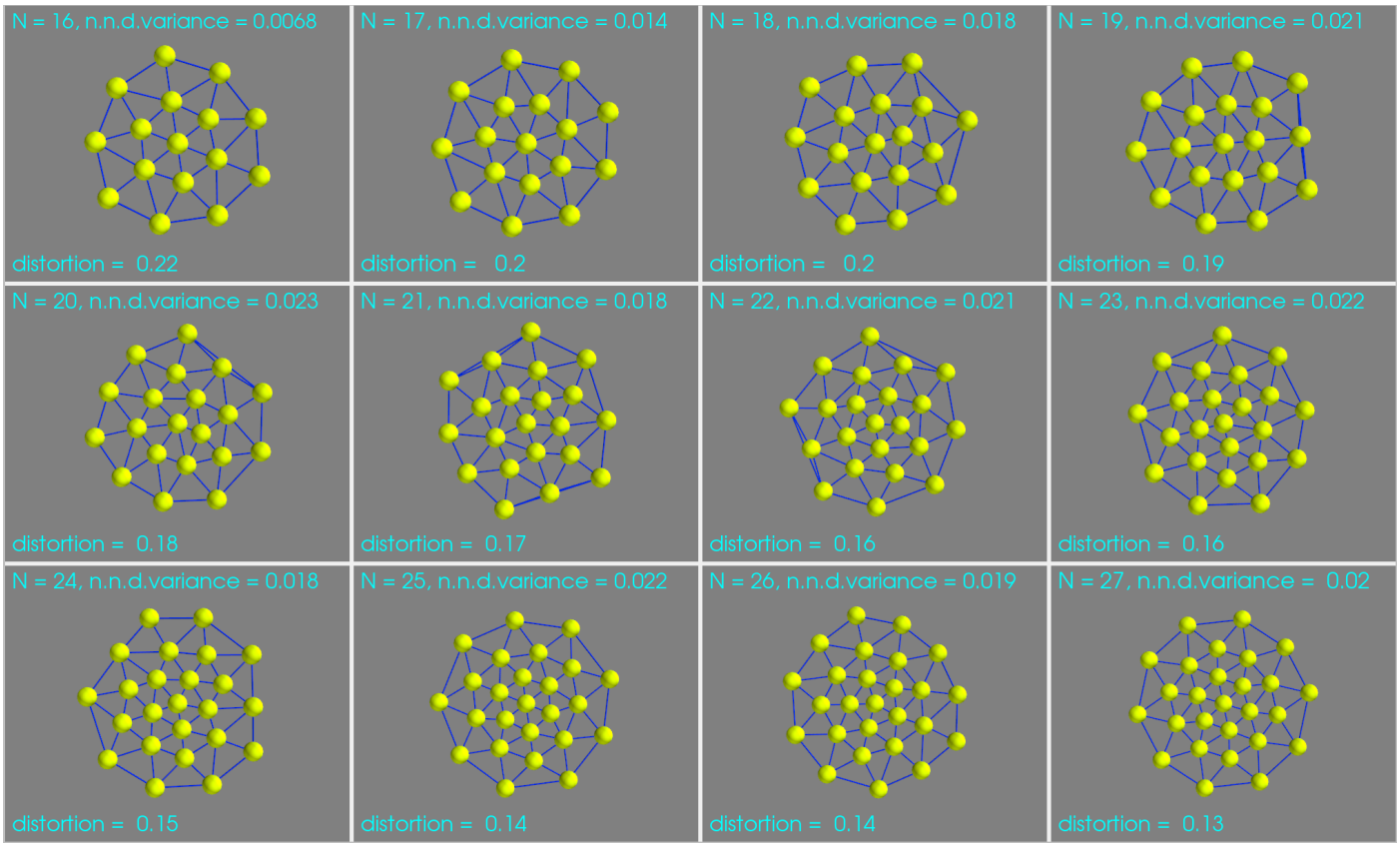


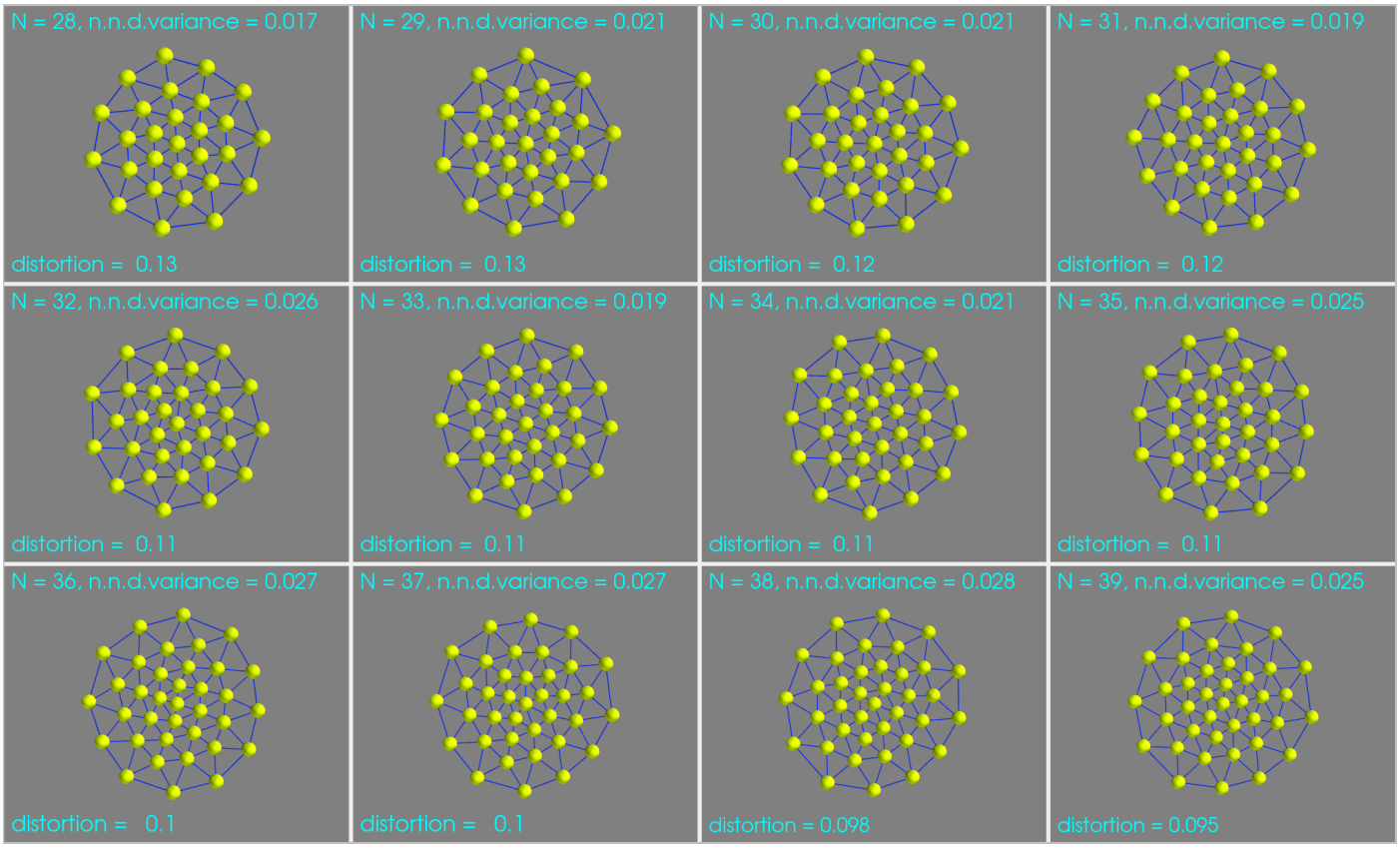
- Short of better ideas, we resorted to visual inspection.
- We also computed the *nearest neighbour distance variance* (n.n.d.variance) for each cluster $\mathcal{C}$ defined as

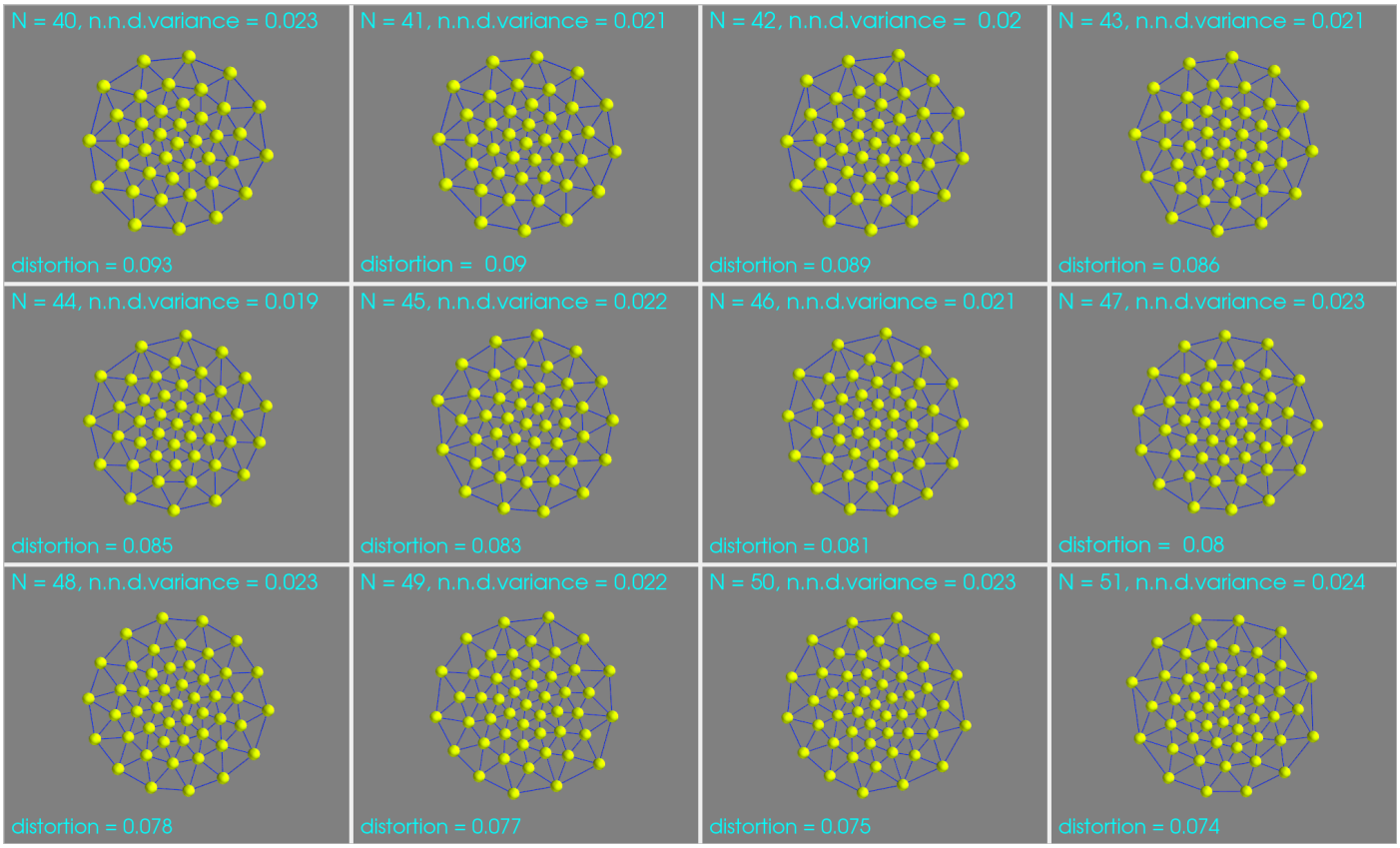
$$\mathrm{V}_{\xi\in\mathcal{C}}\Big[\big|\xi - \underbrace{\xi_{\hat{k}(\xi;\mathcal{C})}}_{\text{"nearest neighbour of } \xi \text{ in cluster } \mathcal{C}\text{"}}\big|\Big] \ . \tag{6.1}$$

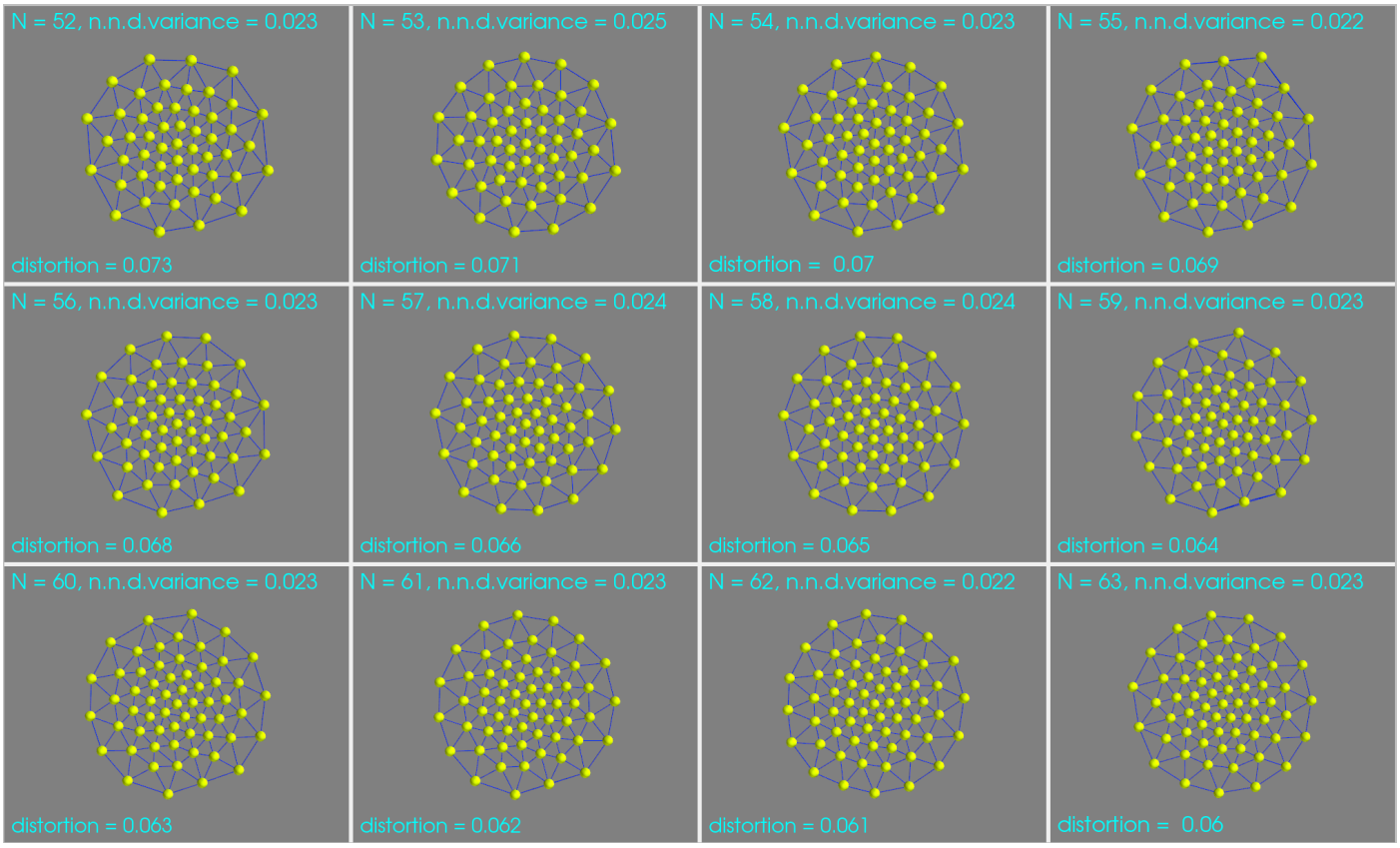


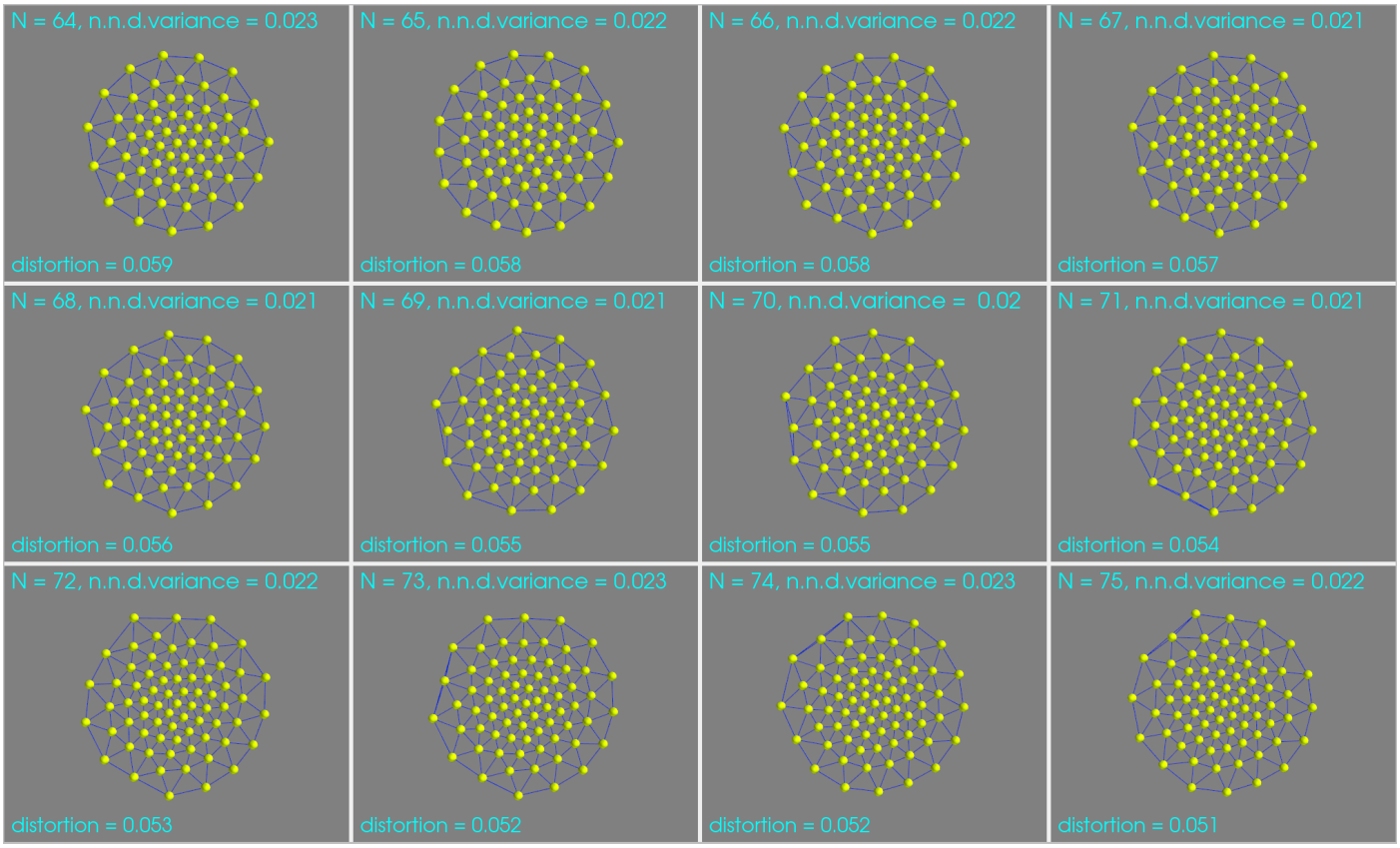
Note: blue edges represent the Delaunay triangularization and are not part of the cluster.

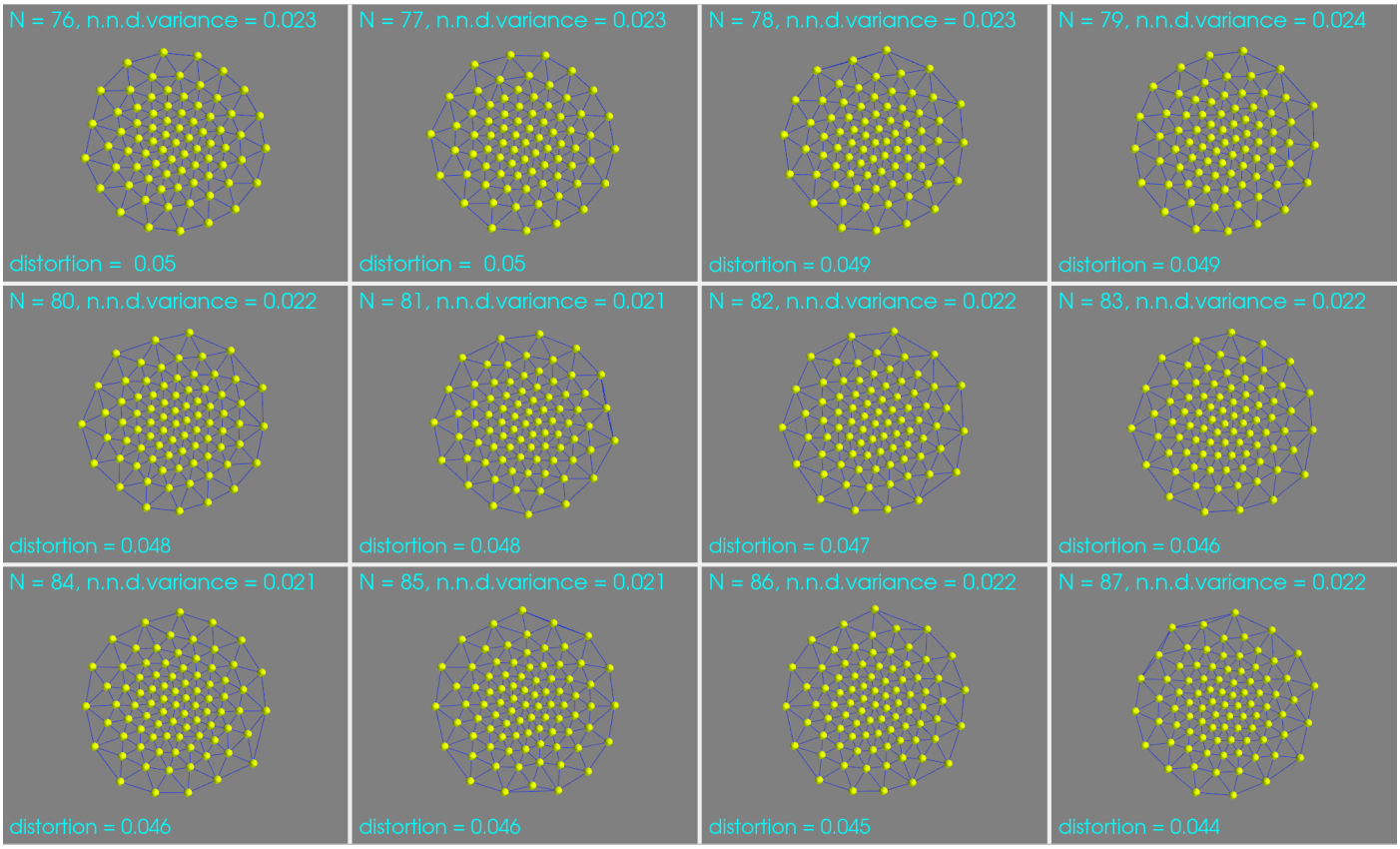

N = 16, n.n.d.variance = 0.0068
distortion = 0.22
N = 17, n.n.d.variance = 0.014
distortion = 0.2
N = 18, n.n.d.variance = 0.018
distortion = 0.2
N = 19, n.n.d.variance = 0.021
distortion = 0.19
N = 20, n.n.d.variance = 0.023
distortion = 0.18
N = 21, n.n.d.variance = 0.018
distortion = 0.17
N = 22, n.n.d.variance = 0.021
distortion = 0.16
N = 23, n.n.d.variance = 0.022
distortion = 0.16
N = 24, n.n.d.variance = 0.018
distortion = 0.15
N = 25, n.n.d.variance = 0.022
distortion = 0.14
N = 26, n.n.d.variance = 0.019
distortion = 0.14
N = 27, n.n.d.variance = 0.02
distortion = 0.13


N = 28, n.n.d.variance = 0.017
distortion = 0.13
N = 29, n.n.d.variance = 0.021
distortion = 0.13
N = 30, n.n.d.variance = 0.021
distortion = 0.12
N = 31, n.n.d.variance = 0.019
distortion = 0.12
N = 32, n.n.d.variance = 0.026
distortion = 0.11
N = 33, n.n.d.variance = 0.019
distortion = 0.11
N = 34, n.n.d.variance = 0.021
distortion = 0.11
N = 35, n.n.d.variance = 0.025
distortion = 0.11
N = 36, n.n.d.variance = 0.027
distortion = 0.1
N = 37, n.n.d.variance = 0.027
distortion = 0.1
N = 38, n.n.d.variance = 0.028
distortion = 0.098
N = 39, n.n.d.variance = 0.025
distortion = 0.095

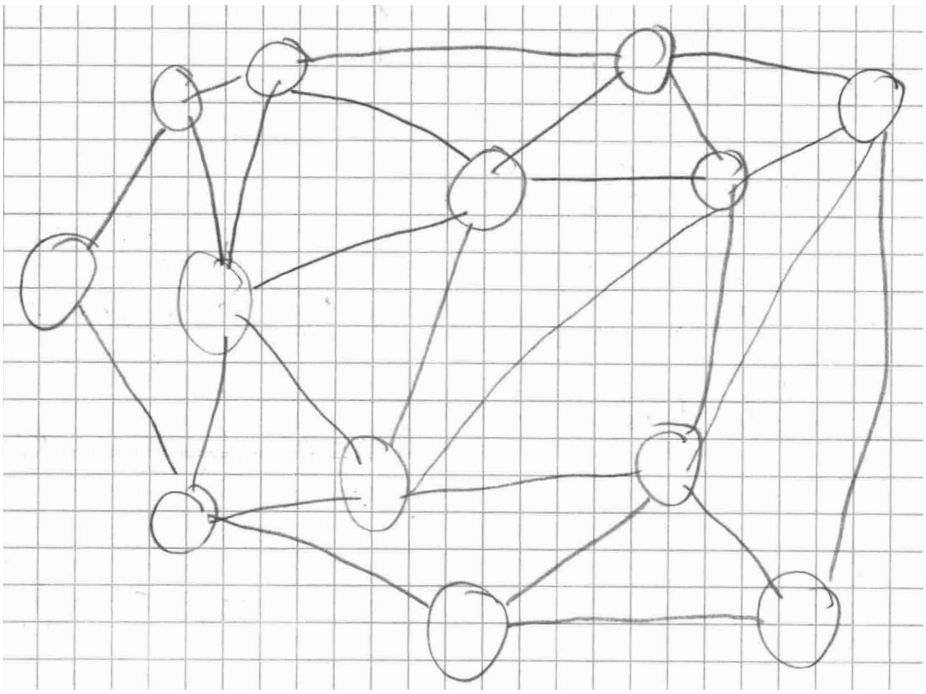

N = 40, n.n.d.variance = 0.023
distortion = 0.093
N = 41, n.n.d.variance = 0.021
distortion = 0.09
N = 42, n.n.d.variance = 0.02
distortion = 0.089
N = 43, n.n.d.variance = 0.021
distortion = 0.086
N = 44, n.n.d.variance = 0.019
distortion = 0.085
N = 45, n.n.d.variance = 0.022
distortion = 0.083
N = 46, n.n.d.variance = 0.021
distortion = 0.081
N = 47, n.n.d.variance = 0.023
distortion = 0.08
N = 48, n.n.d.variance = 0.023
distortion = 0.078
N = 49, n.n.d.variance = 0.022
distortion = 0.077
N = 50, n.n.d.variance = 0.023
distortion = 0.075
N = 51, n.n.d.variance = 0.024
distortion = 0.074


N = 52, n.n.d.variance = 0.023
distortion = 0.073
N = 53, n.n.d.variance = 0.025
distortion = 0.071
N = 54, n.n.d.variance = 0.023
distortion = 0.07
N = 55, n.n.d.variance = 0.022
distortion = 0.069
N = 56, n.n.d.variance = 0.023
distortion = 0.068
N = 57, n.n.d.variance = 0.024
distortion = 0.066
N = 58, n.n.d.variance = 0.024
distortion = 0.065
N = 59, n.n.d.variance = 0.023
distortion = 0.064
N = 60, n.n.d.variance = 0.023
distortion = 0.063
N = 61, n.n.d.variance = 0.023
distortion = 0.062
N = 62, n.n.d.variance = 0.022
distortion = 0.061
N = 63, n.n.d.variance = 0.023
distortion = 0.06


N = 64, n.n.d.variance = 0.023
distortion = 0.059
N = 65, n.n.d.variance = 0.022
distortion = 0.058
N = 66, n.n.d.variance = 0.022
distortion = 0.058
N = 67, n.n.d.variance = 0.021
distortion = 0.057
N = 68, n.n.d.variance = 0.021
distortion = 0.056
N = 69, n.n.d.variance = 0.021
distortion = 0.055
N = 70, n.n.d.variance = 0.02
distortion = 0.055
N = 71, n.n.d.variance = 0.021
distortion = 0.054
N = 72, n.n.d.variance = 0.022
distortion = 0.053
N = 73, n.n.d.variance = 0.023
distortion = 0.052
N = 74, n.n.d.variance = 0.023
distortion = 0.052
N = 75, n.n.d.variance = 0.022
distortion = 0.051

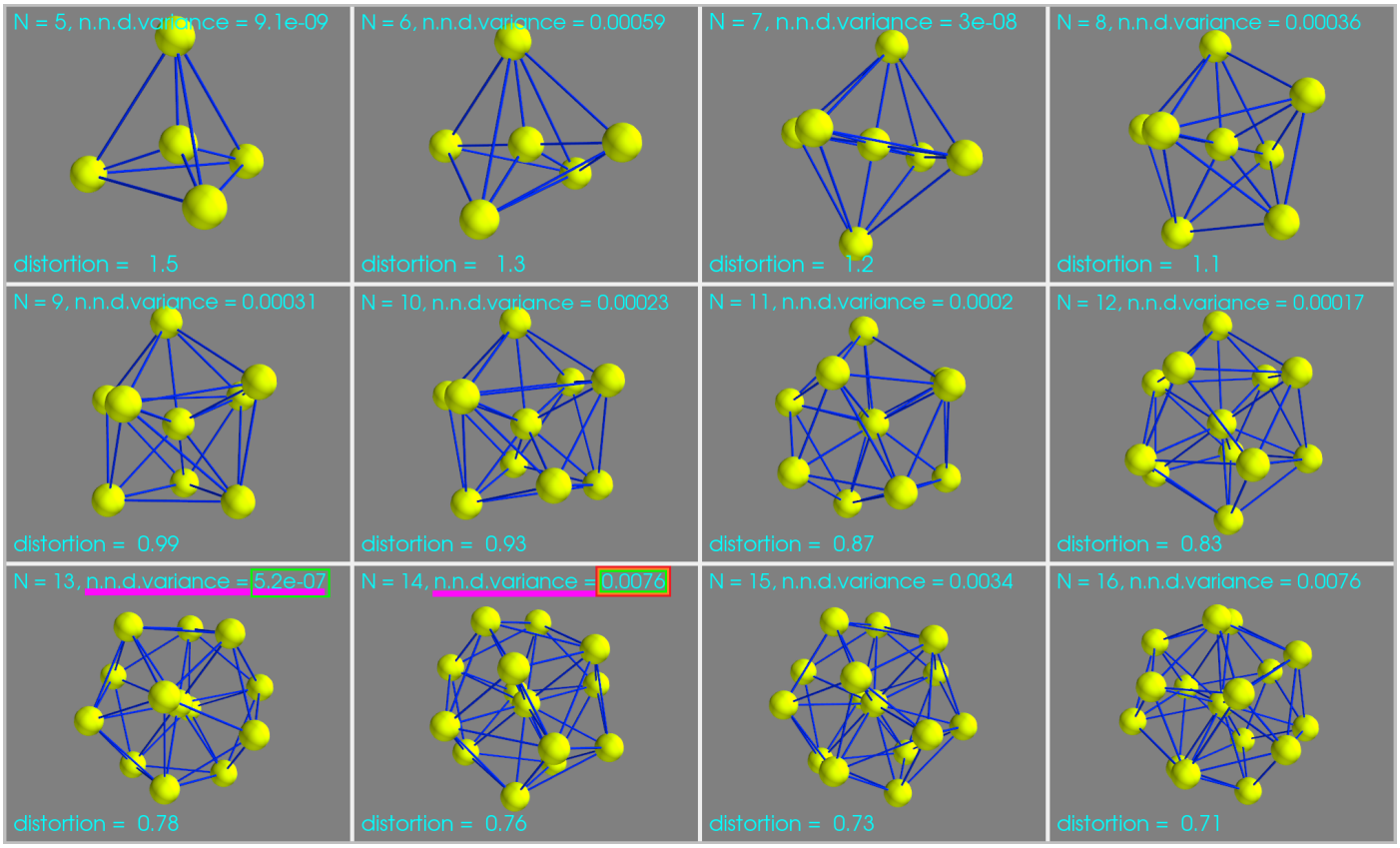

N = 76, n.n.d.variance = 0.023
distortion = 0.05
N = 77, n.n.d.variance = 0.023
distortion = 0.05
N = 78, n.n.d.variance = 0.023
distortion = 0.049
N = 79, n.n.d.variance = 0.024
distortion = 0.049
N = 80, n.n.d.variance = 0.022
distortion = 0.048
N = 81, n.n.d.variance = 0.021
distortion = 0.048
N = 82, n.n.d.variance = 0.022
distortion = 0.047
N = 83, n.n.d.variance = 0.022
distortion = 0.046
N = 84, n.n.d.variance = 0.021
distortion = 0.046
N = 85, n.n.d.variance = 0.021
distortion = 0.046
N = 86, n.n.d.variance = 0.022
distortion = 0.045
N = 87, n.n.d.variance = 0.022
distortion = 0.044

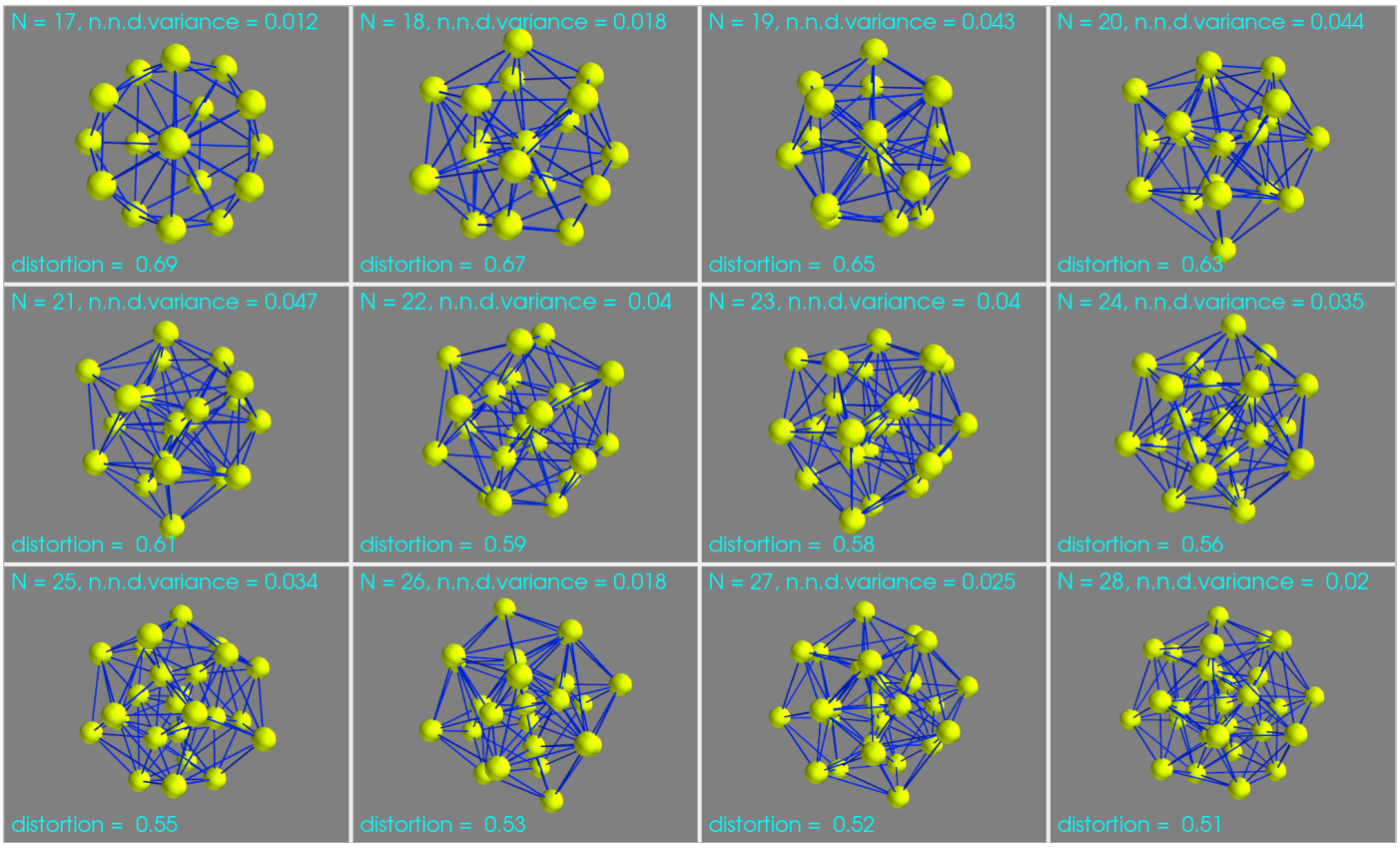
My daughter's sketch of a *rather bad 2D cluster* :

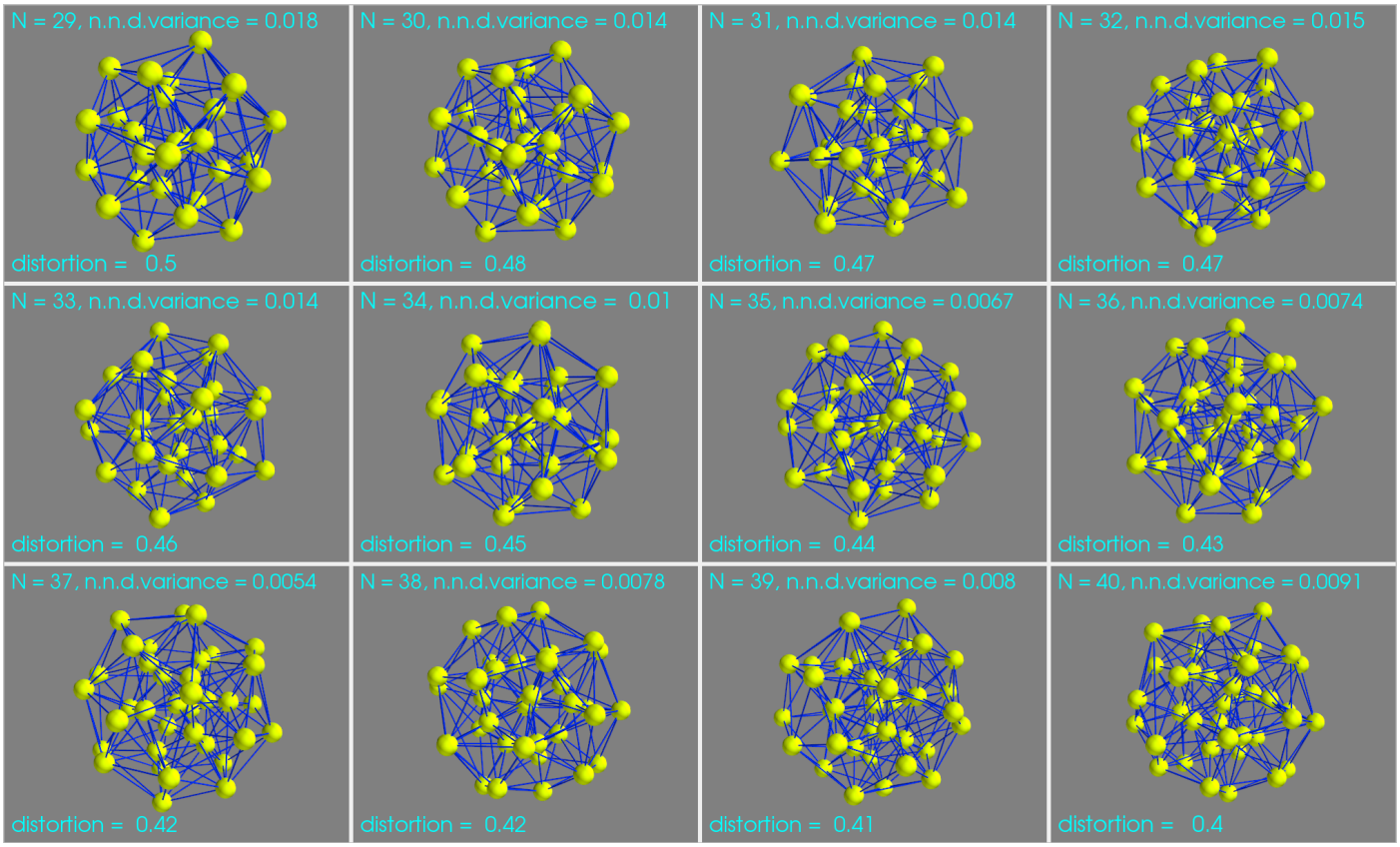


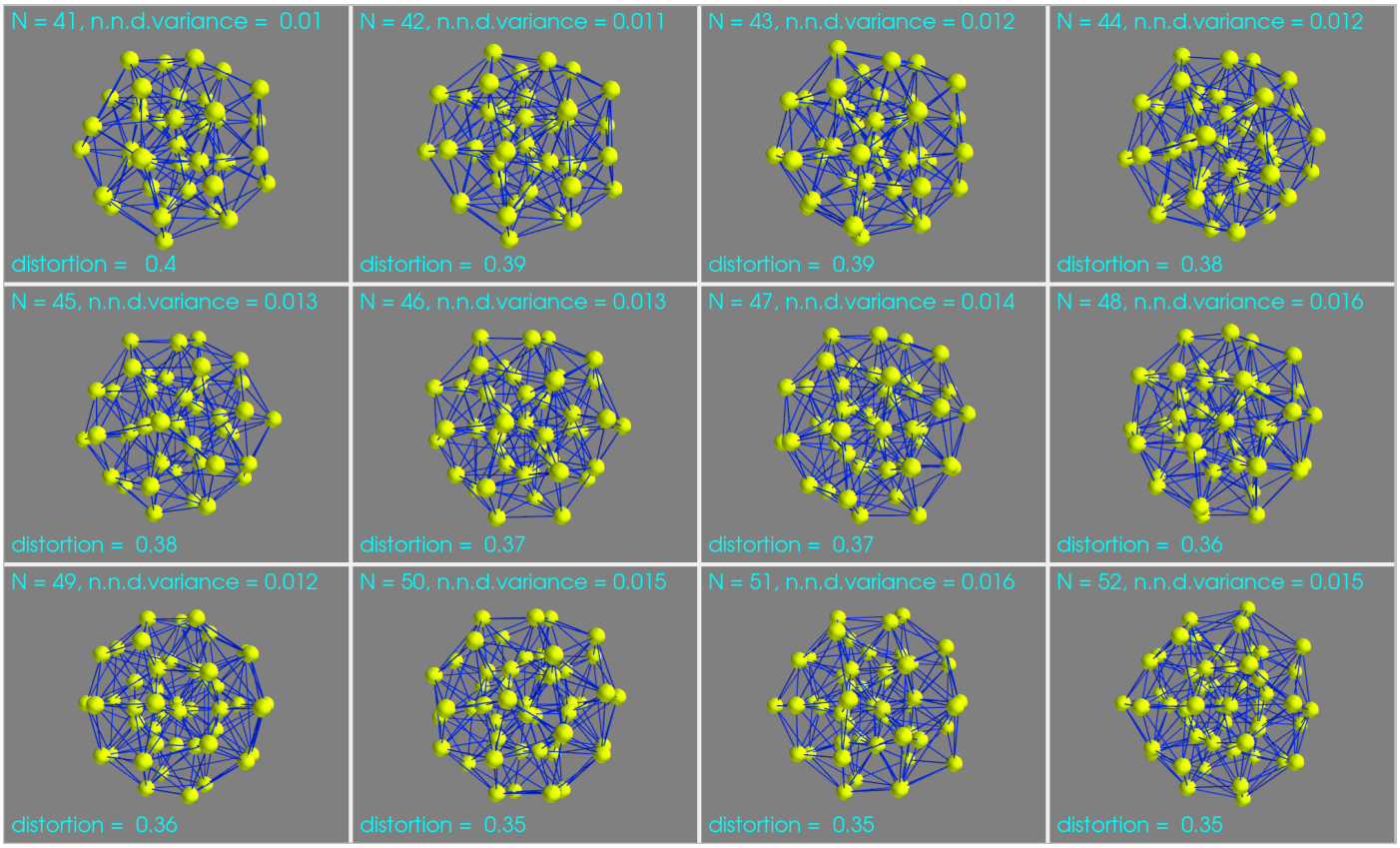
N=13, n.n.d.v. = 0.054

- In 2 dimensions, the *n*earest *n*eighbour *d*istance *v*ariance seems to give us an indication when clusters first become "*irregular*".
- As the cluster size grows, we can visually see how their regularity varies from one size to the next, even if it is non-trivial to quantify this mathematically.
- What about 3 dimensions?

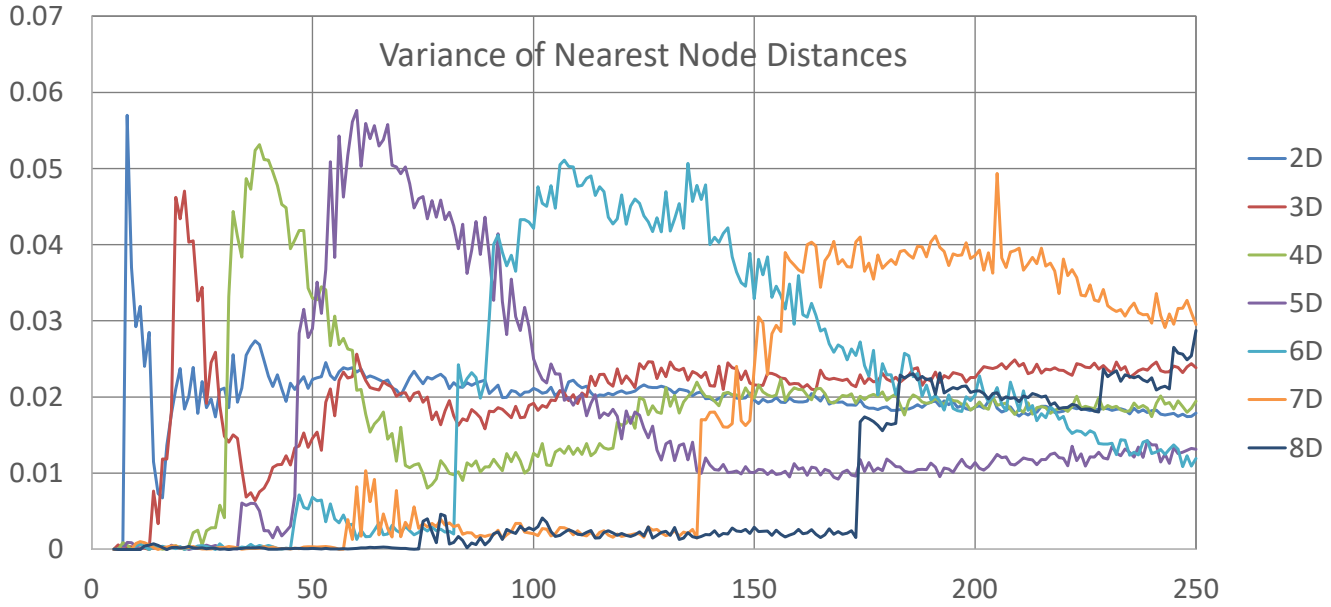

N = 5, n.n.d.variance = 9.1e-09
distortion = 1.5
N = 6, n.n.d.variance = 0.00059
distortion = 1.3
N = 7, n.n.d.variance = 3e-08
distortion = 1.2
N = 8, n.n.d.variance = 0.00036
distortion = 1.1
N = 9, n.n.d.variance = 0.00031
distortion = 0.99
N = 10, n.n.d.variance = 0.00023
distortion = 0.93
N = 11, n.n.d.variance = 0.0002
distortion = 0.87
N = 12, n.n.d.variance = 0.00017
distortion = 0.83
N = 13, n.n.d.variance = 5.2e-07
distortion = 0.78
N = 14, n.n.d.variance = 0.0076
distortion = 0.76
N = 15, n.n.d.variance = 0.0034
distortion = 0.73
N = 16, n.n.d.variance = 0.0076
distortion = 0.71


N = 17, n.n.d.variance = 0.012
distortion = 0.69
N = 18, n.n.d.variance = 0.018
distortion = 0.67
N = 19, n.n.d.variance = 0.043
distortion = 0.65
N = 20, n.n.d.variance = 0.044
distortion = 0.63
N = 21, n.n.d.variance = 0.047
distortion = 0.61
N = 22, n.n.d.variance = 0.04
distortion = 0.59
N = 23, n.n.d.variance = 0.04
distortion = 0.58
N = 24, n.n.d.variance = 0.035
distortion = 0.56
N = 25, n.n.d.variance = 0.034
distortion = 0.55
N = 26, n.n.d.variance = 0.018
distortion = 0.53
N = 27, n.n.d.variance = 0.025
distortion = 0.52
N = 28, n.n.d.variance = 0.02
distortion = 0.51


N = 29, n.n.d.variance = 0.018
distortion = 0.5
N = 30, n.n.d.variance = 0.014
distortion = 0.48
N = 31, n.n.d.variance = 0.014
distortion = 0.47
N = 32, n.n.d.variance = 0.015
distortion = 0.47
N = 33, n.n.d.variance = 0.014
distortion = 0.46
N = 34, n.n.d.variance = 0.01
distortion = 0.45
N = 35, n.n.d.variance = 0.0067
distortion = 0.44
N = 36, n.n.d.variance = 0.0074
distortion = 0.43
N = 37, n.n.d.variance = 0.0054
distortion = 0.42
N = 38, n.n.d.variance = 0.0078
distortion = 0.42
N = 39, n.n.d.variance = 0.008
distortion = 0.41
N = 40, n.n.d.variance = 0.0091
distortion = 0.4

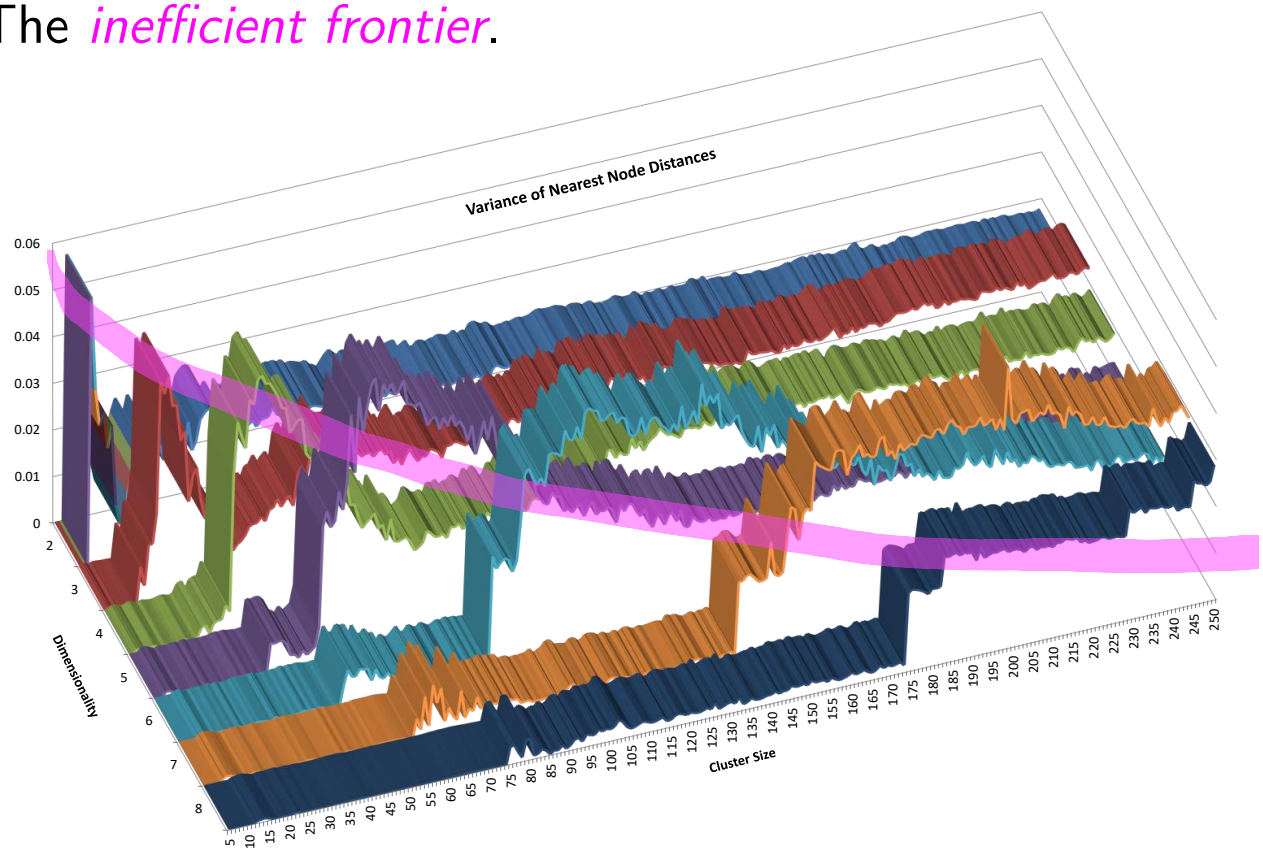

N = 41, n.n.d.variance = 0.01
distortion = 0.4
N = 42, n.n.d.variance = 0.011
distortion = 0.39
N = 43, n.n.d.variance = 0.012
distortion = 0.39
N = 44, n.n.d.variance = 0.012
distortion = 0.38
N = 45, n.n.d.variance = 0.013
distortion = 0.38
N = 46, n.n.d.variance = 0.013
distortion = 0.37
N = 47, n.n.d.variance = 0.014
distortion = 0.37
N = 48, n.n.d.variance = 0.016
distortion = 0.36
N = 49, n.n.d.variance = 0.012
distortion = 0.36
N = 50, n.n.d.variance = 0.015
distortion = 0.35
N = 51, n.n.d.variance = 0.016
distortion = 0.35
N = 52, n.n.d.variance = 0.015
distortion = 0.35

Let's revisit the *nearest neighbour distance variance*:



- We see a sudden jump in each dimensionality.
- In 2 and 3 dimensions, this happens where the ideal spherical packing number is exceeded for the first time.

The *inefficient frontier*.



- It turns out, that the jumps are related to the point when adding one more cluster node would not be able to fit into the quantized Gaussian distribution without causing asymmetry.
- Recall that our multivariate Gaussian distribution has *spherical symmetry*, and that we associate a segment of space with each cluster node.
- Going from n to $n+1$ nodes is similar to trying to fit another spatially extended, let's say, approximately, *spherical* node into the cluster.
- This raises the question: how many (hyper)spheres of (approximately) equal size can be fitted *neatly* around a centre sphere?
- This is known as the ***Kissing Number Problem***.
- Quote Wikipedia [Kis18]:

  In geometry, the kissing number of a mathematical space is defined as the greatest number of non-overlapping unit spheres that can be arranged in that space such that they each touch a common unit sphere.

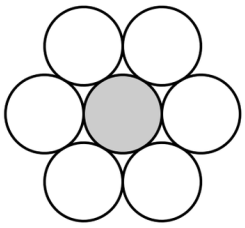
More Wikipedia [Kis18]:

One dimension

In one dimension,[4] the kissing number is 2:

Two dimensions

In two dimensions, the kissing number is 6:



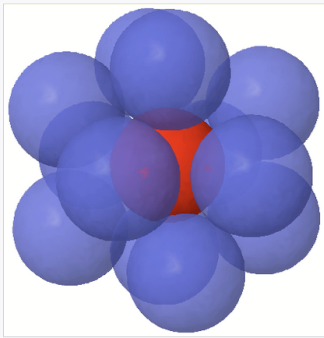
Proof: Consider a circle with center C that is touched by circles with centers C_1, C_2, ... Consider the rays $C\,C_i$. These rays all emanate from the same center C, so the sum of angles between adjacent rays is 360°.

Assume by contradiction that there are more than six touching circles. Then at least two adjacent rays, say $C\,C_1$ and $C\,C_2$, are separated by an angle of less than 60°. The segments $C\,C_i$ have the same length – $2r$ – for all i. Therefore, the triangle $C\,C_1\,C_2$ is isosceles, and its third side – $C_1\,C_2$ – has a side length of less than $2r$. Therefore, the circles 1 and 2 intersect – a contradiction.[5]

Three dimensions

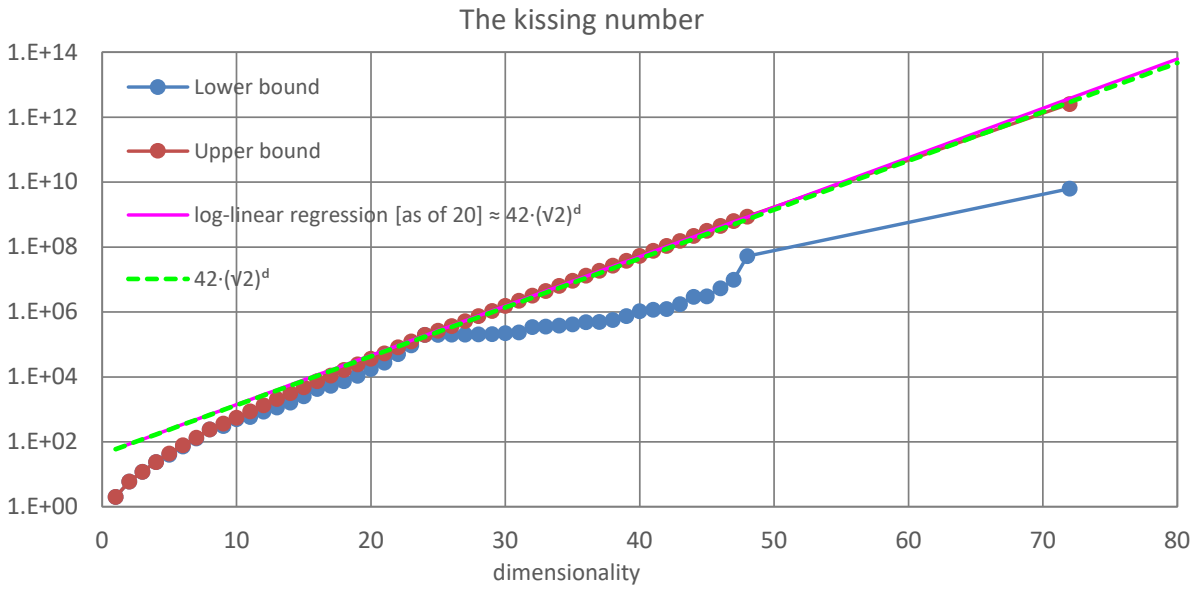
In three dimensions, the kissing number is 12, but the correct value was much more difficult to establish than in dimensions one and two. It is easy to arrange 12 spheres so that each touches a central sphere, with a lot of space left over, and it is not obvious that there is no way to pack in a 13th sphere. (In fact, there is so much extra space that any two of the 12 outer spheres can exchange places through a continuous movement without any of the outer spheres losing contact with the center one.) This was the subject of a famous disagreement between mathematicians Isaac Newton and David Gregory. Newton correctly thought that the limit was 12; Gregory thought that a 13th could fit. Some incomplete proofs that Newton was correct were offered in the nineteenth century, most notably one by Reinhold Hoppe, but the first correct proof (according to Brass, Moser, and Pach) did not appear until 1953.[1][2][6]

The twelve neighbors of the central sphere correspond to the maximum bulk coordination number of an atom in a crystal lattice in which all atoms have the same size (as in a chemical element). A coordination number of 12 is found in a cubic close-packed or a hexagonal close-packed structure.



A highly symmetrical realization of the kissing number 12 in three dimensions is by aligning the centers of outer spheres with vertices of a regular icosahedron. This leaves slightly more than 0.1 of the radius between two nearby spheres.

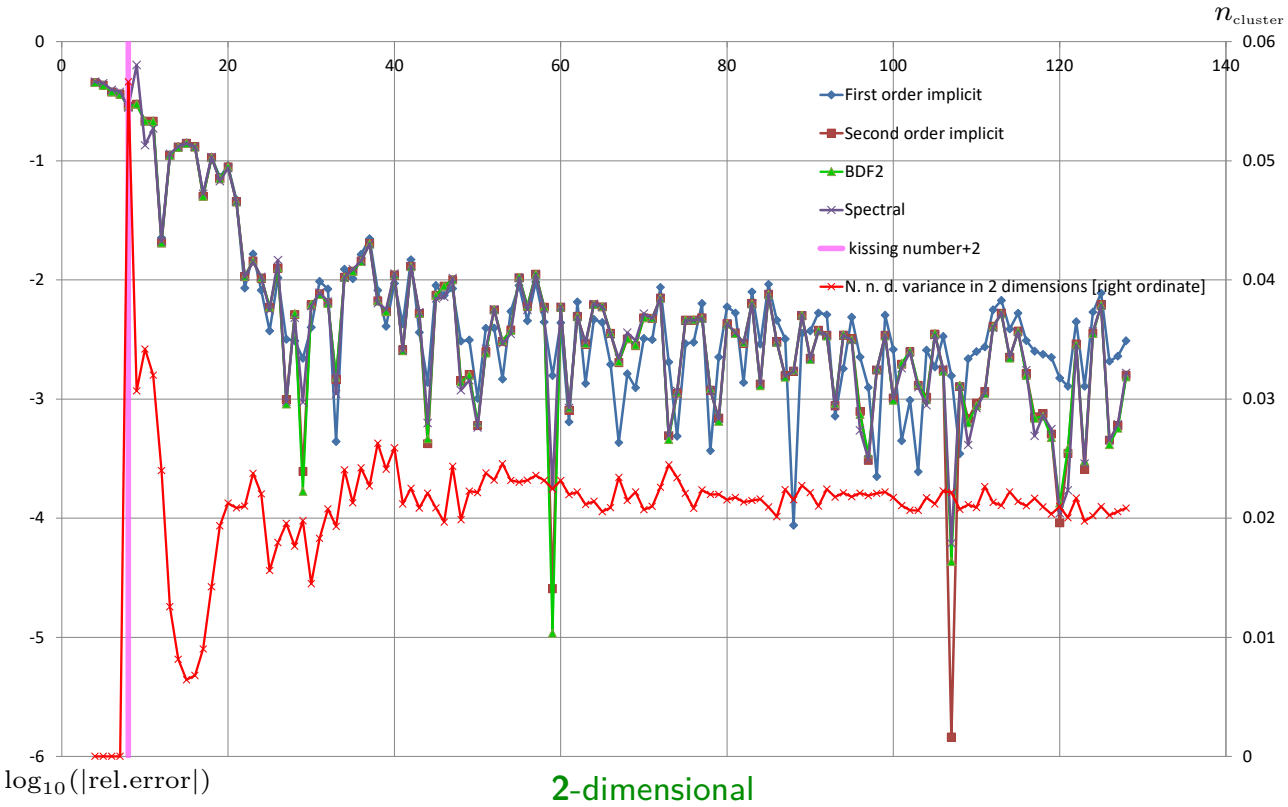
- In 1, 2, 3, 4, 8, and 24 dimensions, the kissing number is known exactly.
- We also have some upper and lower bounds up to 72D [Kis18, Coh18].



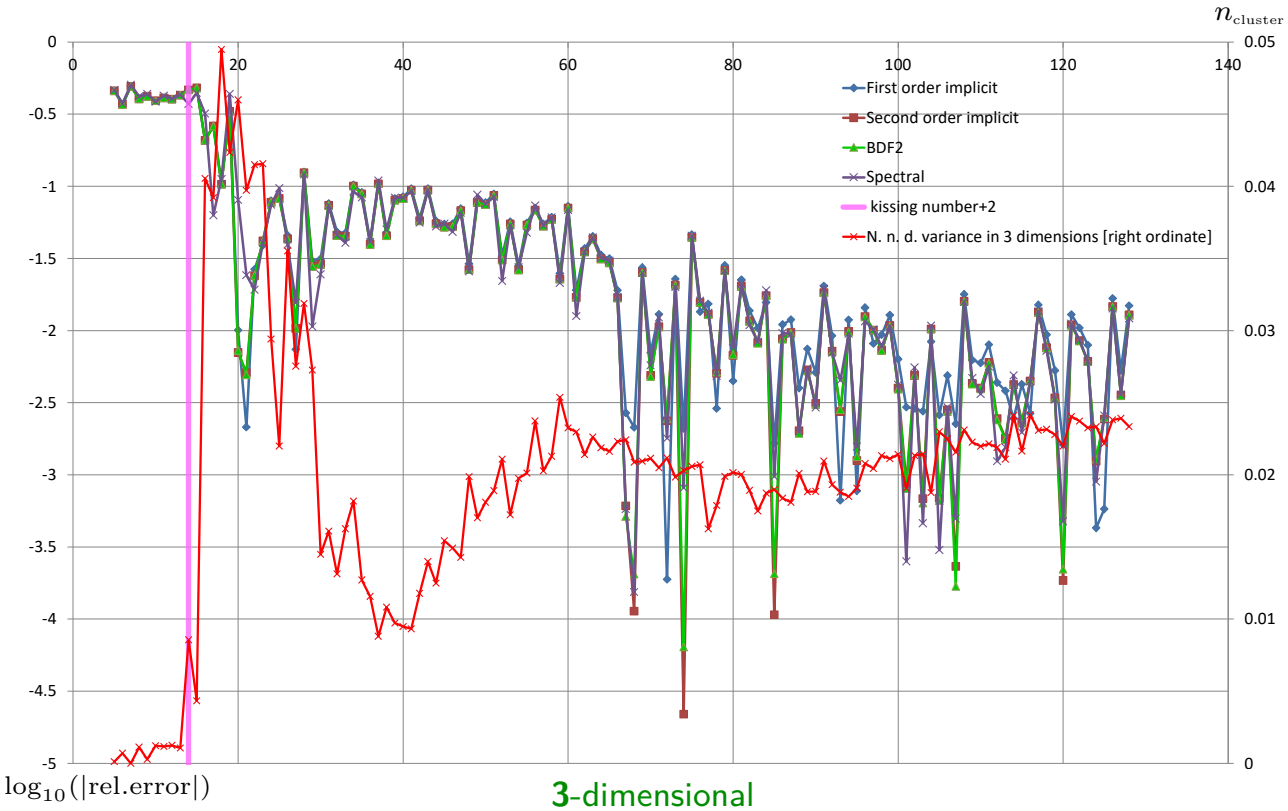
- The spherical kissing number (upper bound) scales approximately like

$$42 \cdot \sqrt{2}^{\,d} \quad . \tag{7.1}$$

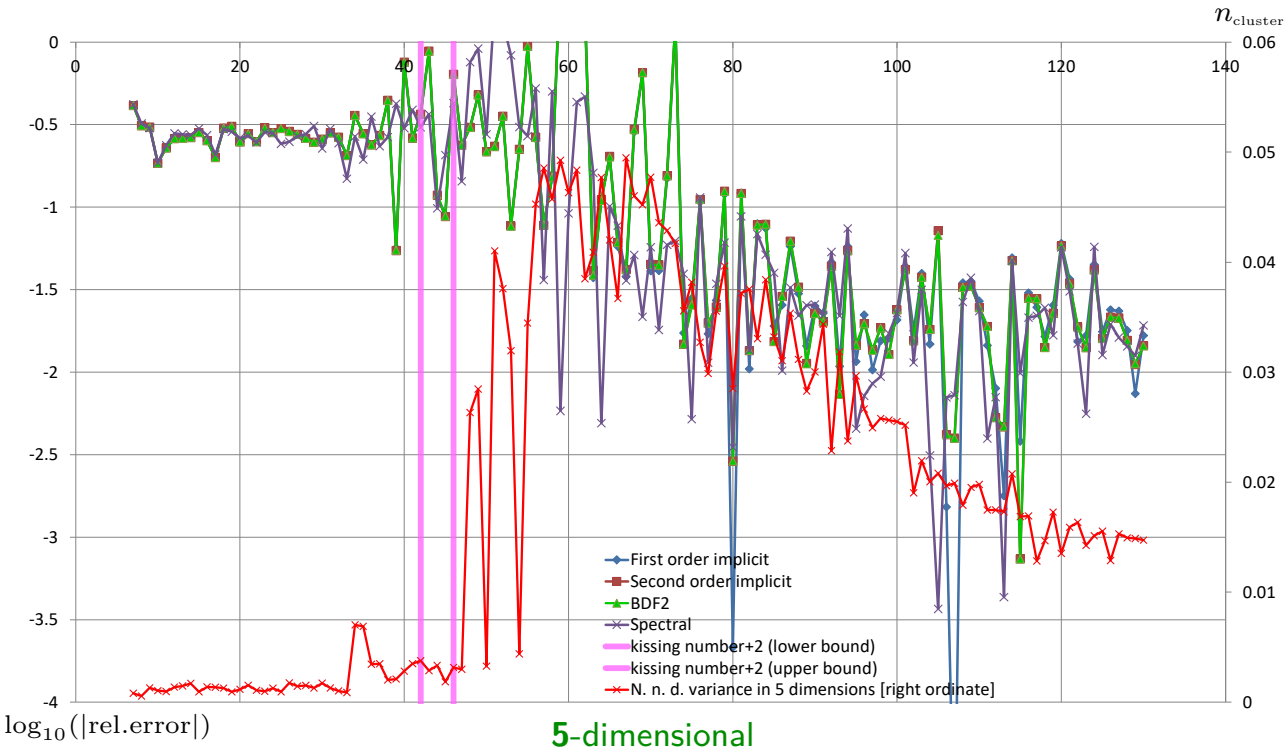
Comparing Gaussian cluster results with spherical kissing numbers: 2D.

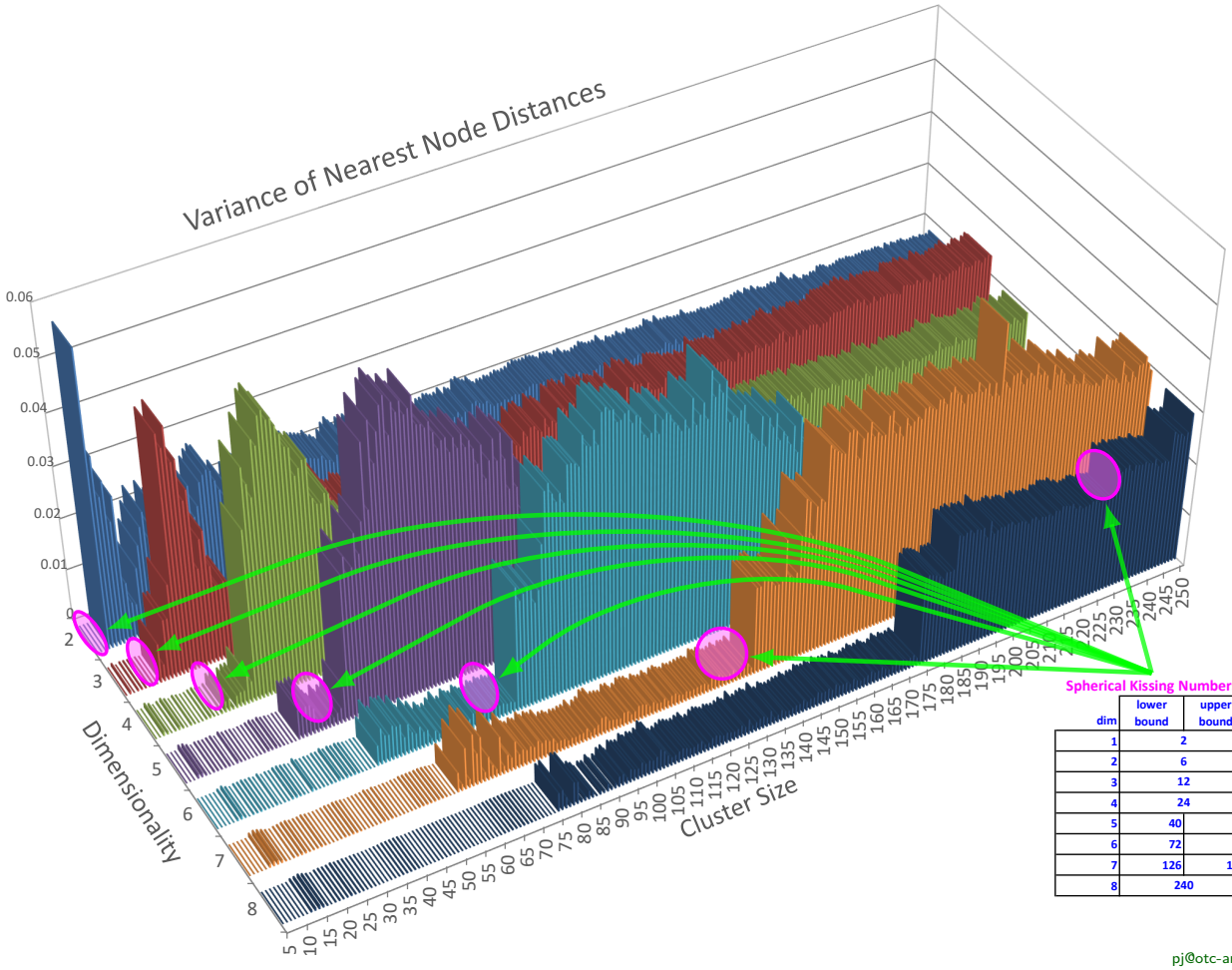


Comparing Gaussian cluster results with spherical kissing numbers: 3D.



Comparing Gaussian cluster results with spherical kissing numbers: 5D.





Spherical Kissing Numbers

dim	lower bound	upper bound
1	2	
2	6	
3	12	
4	24	
5	40	44
6	72	78
7	126	134
8	240	

Conclusions

- Backward induction can be done lattice-free for advection-diffusions.
  I have been able to run calculations with cluster induction in up to 11 dimensions[13].
- Radial basis functions are immensely useful.
- We can compute *Minimum Distortion Clusters*, aka

  Centroidal Voronoi Tesselations

  in a timely fashion[14] by the aid of Sobol' sampling and the

  "Hamerly-k-Means Clustering"

  algorithm [Ham10].

[13] though perhaps as yet not robustly enough for industrial deployment
[14] Note that optimal clusters are problem-independent and can be cached.

- We define the

  Gaussian Kissing Number $\mathcal{N}^{\mathcal{G}}(d)$

  in d dimensions as the largest number n such that the

  Centroidal Voronoi Tesselation

  of $n+1$ nodes[15] under an uncorrelated d-dimensional standard Gaussian density is comprised by all n non-central nodes having equal distance r to the origin:

$$\mathcal{N}^{\mathcal{G}}(d) \;=\; \max N \,\Big|\; |\boldsymbol{\xi}| = r \;\forall\; \boldsymbol{\xi} \;\in\; \mathcal{C}_d^*(N) \setminus \mathbf{0}, \;\text{ for some } r \;\in\; \mathbb{R}^+ \tag{7.2}$$

- In 1, 2, and 3 dimensions, the Gaussian Kissing Number $\mathcal{N}^{\mathcal{G}}(d)$ is equal to the Spherical Kissing Number $\mathcal{N}^{\mathcal{S}}(d)$.

[15] one central node is pinned at the origin

- We cannot expect $\mathcal{N}^{\mathcal{G}}(d) = \mathcal{N}^{\mathcal{S}}(d)$ to hold exactly for all d since the central and the peripheral zones are neither the same shape nor the same volume, nor have the same probability associated with them.
- We may guess, though, that

$$\mathcal{N}^{\mathcal{G}}(d) \approx \mathcal{N}^{\mathcal{S}}(d) \tag{7.3}$$

by virtue of the spherical symmetry of the Gaussian distribution.

- Experimental evidence appears to support (7.3).
- Some exact values and some bounds are known [Kis18, Coh18] for $\mathcal{N}^{\mathcal{S}}(d)$:

d	1	2	3	4	5	6	7	8	9	10	11	12
lower bound	2	6	12	24	40	72	126	240	306	500	582	840
upper bound					44	78	134		363	553	869	1356

- For numerical purposes, choose a cluster size that is either well above, or (just) below $\mathcal{N}^{\mathcal{G}}(d)$ to avoid an unfortunately ill formed cluster.
- Freed from the confines of any lattice, the curse of dimensionality "scales" approximately like

$$\boxed{42 \cdot \sqrt{2}^{d}} \; . \tag{7.1}$$

- This compares favourably with the lattice-based node number scaling

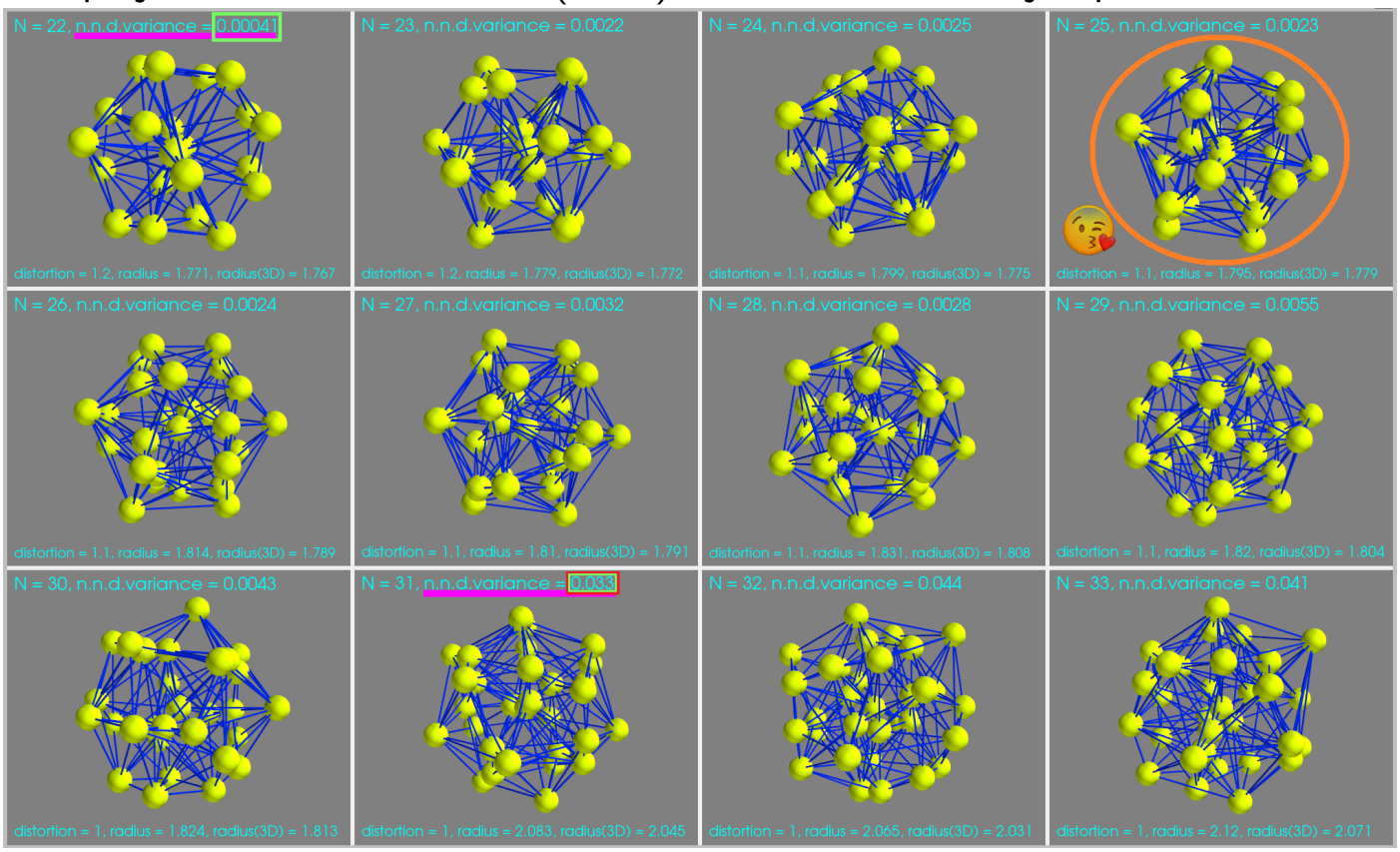
$$n^{d} \tag{7.4}$$

where n is the number of nodes in each direction (typically at least 7).

- In practice, for a variety of reasons, we typically use only 200-500 nodes even when $d = 12$.

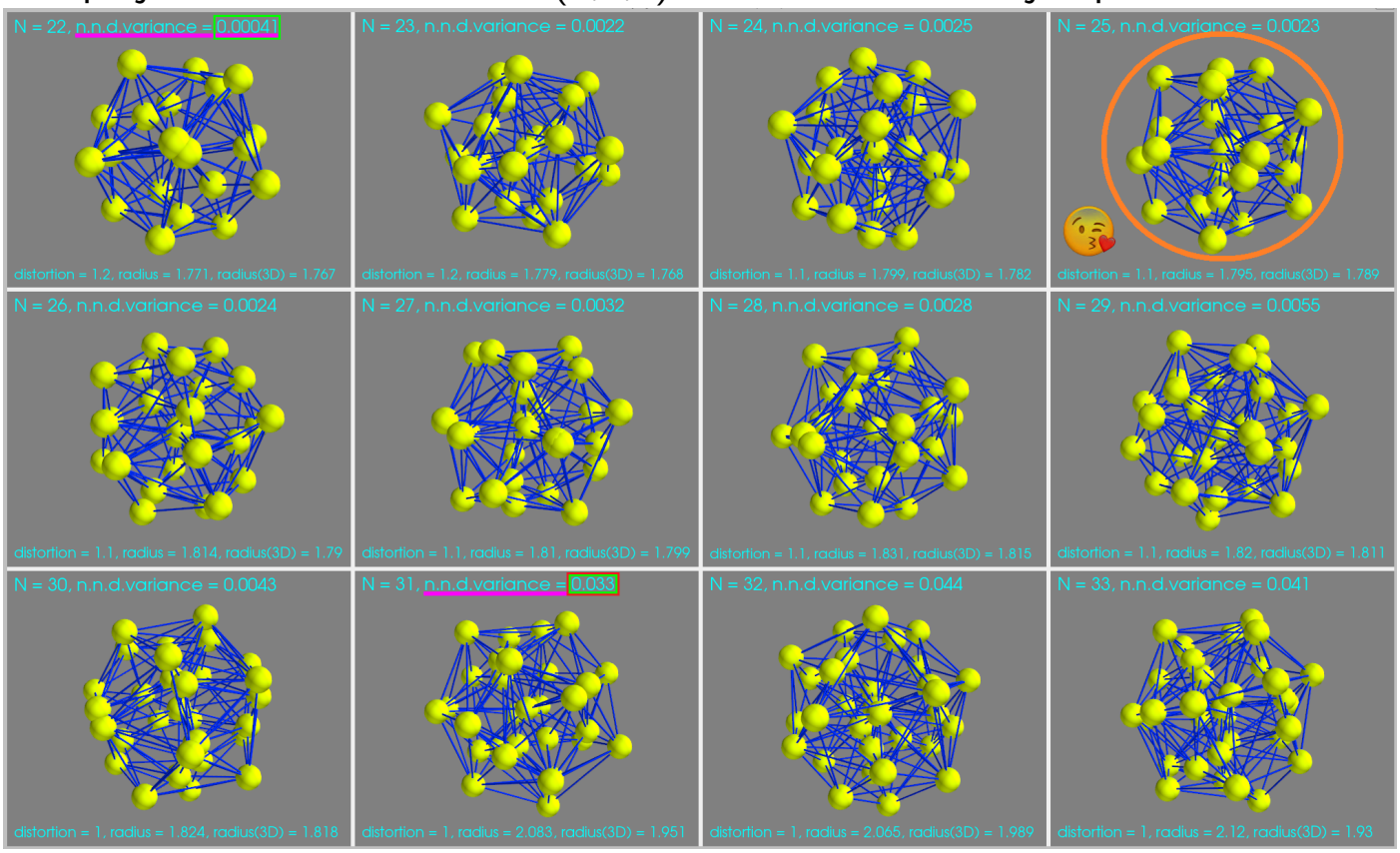
Appendix

4D projected onto dimensions (1,2,3) — no clear sudden jump

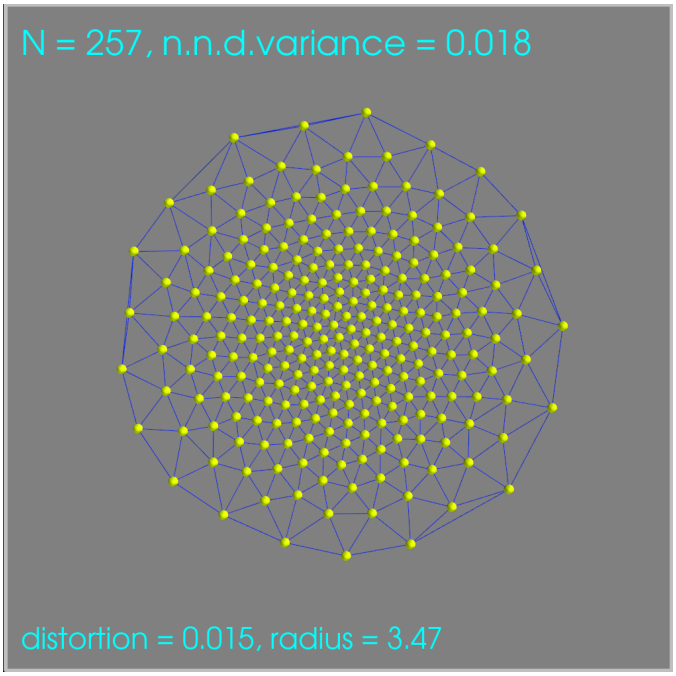


Note: blue edges are the Delaunay triangularization in the 3D projection and not part of the cluster.

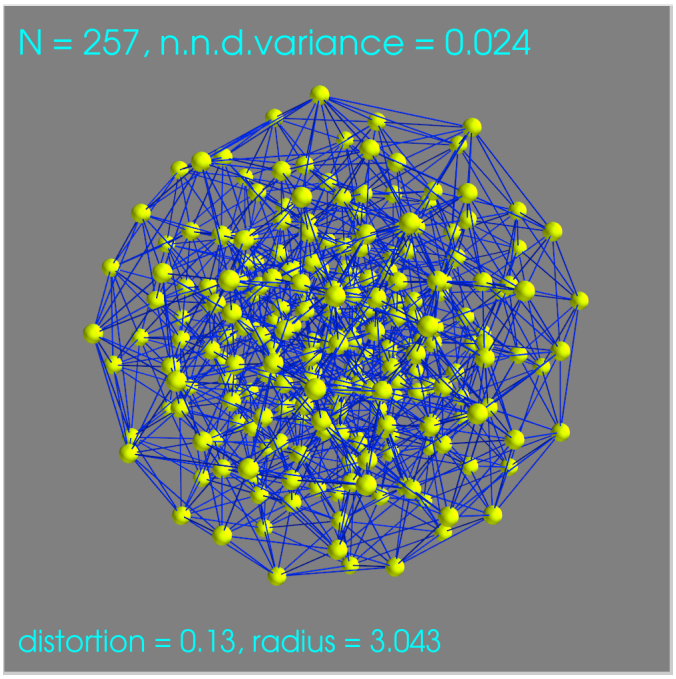
4D projected onto dimensions (2,3,4) — no clear sudden jump



2D



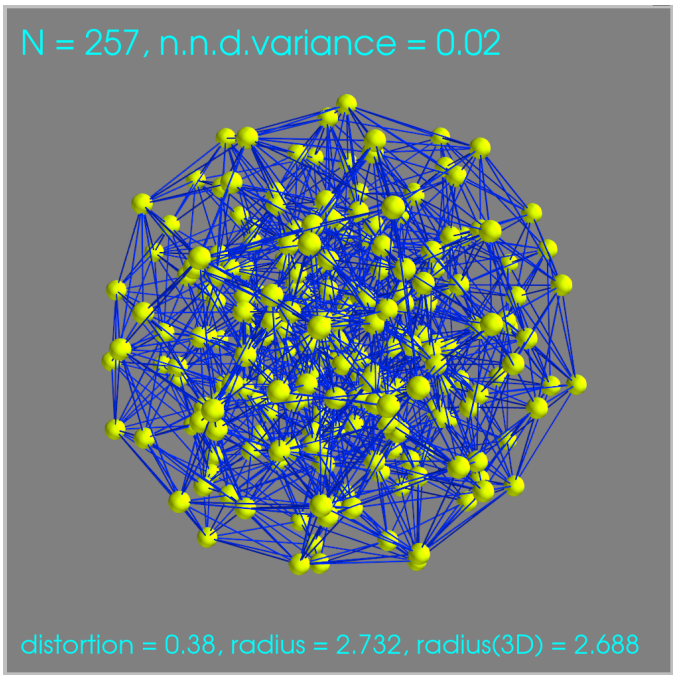
3D



$$N \gg \mathcal{N}^{\mathcal{G}}(2) = 2$$

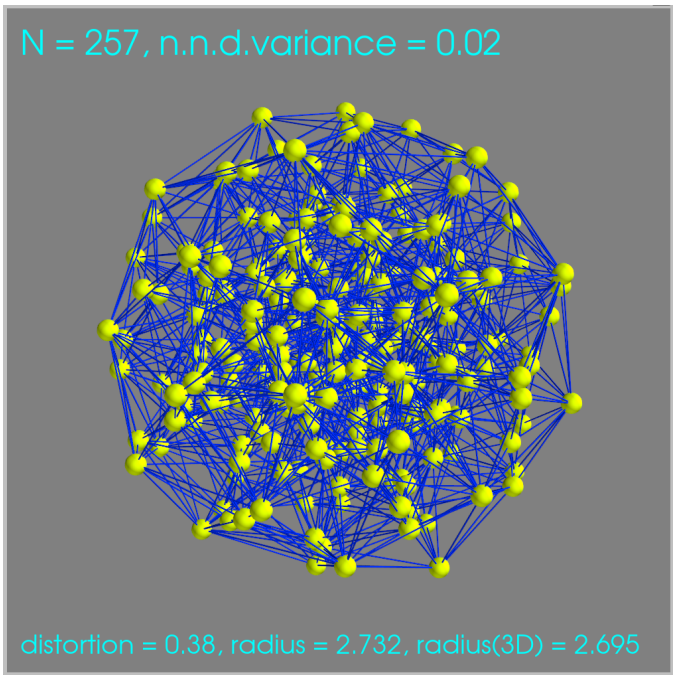
$$N \gg \mathcal{N}^{\mathcal{G}}(3) = 6$$

4D on dimensions (1,2,3)



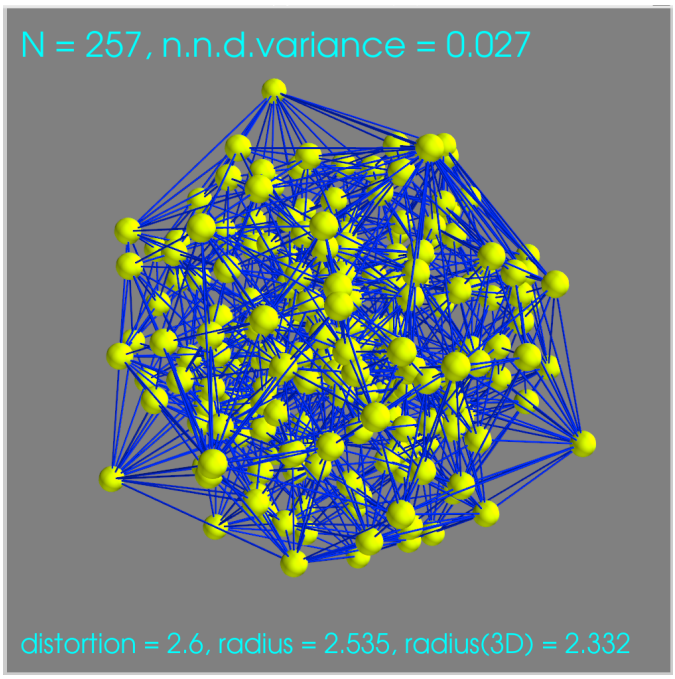
$$N \gg \mathcal{N}^{\mathcal{G}}(4) \approx 24$$

4D on dimensions (2,3,4)



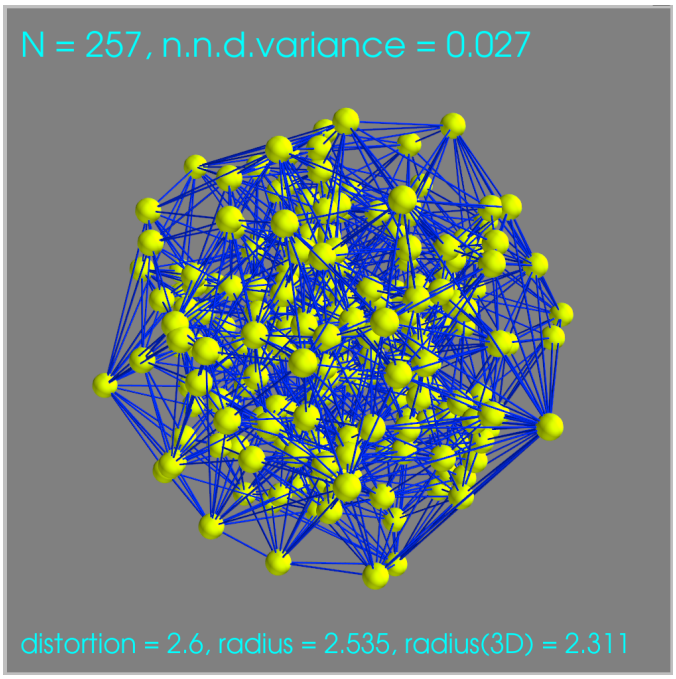
$$N \gg \mathcal{N}^{\mathcal{G}}(4) \approx 24$$

8D on dimensions (1,2,3)



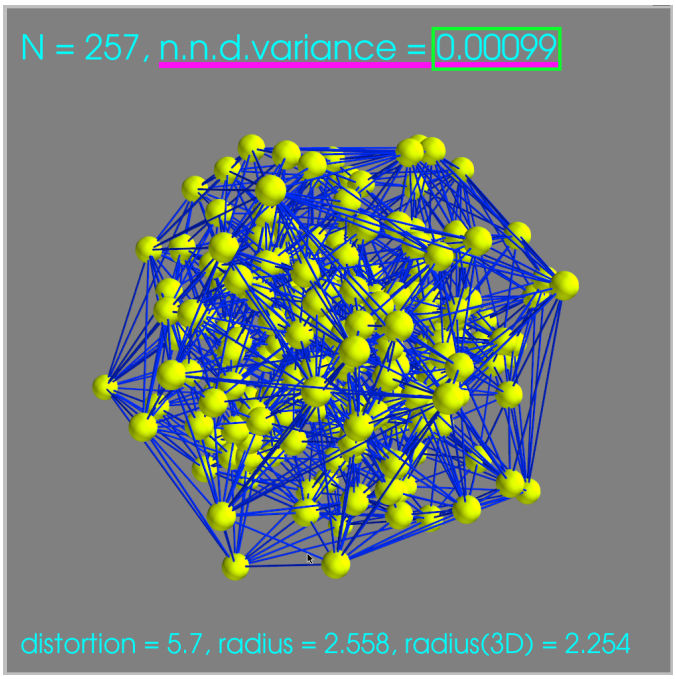
$$N \gtrsim \mathcal{N}^{\mathcal{G}}(8) \approx 240$$

8D on dimensions (6,7,8)



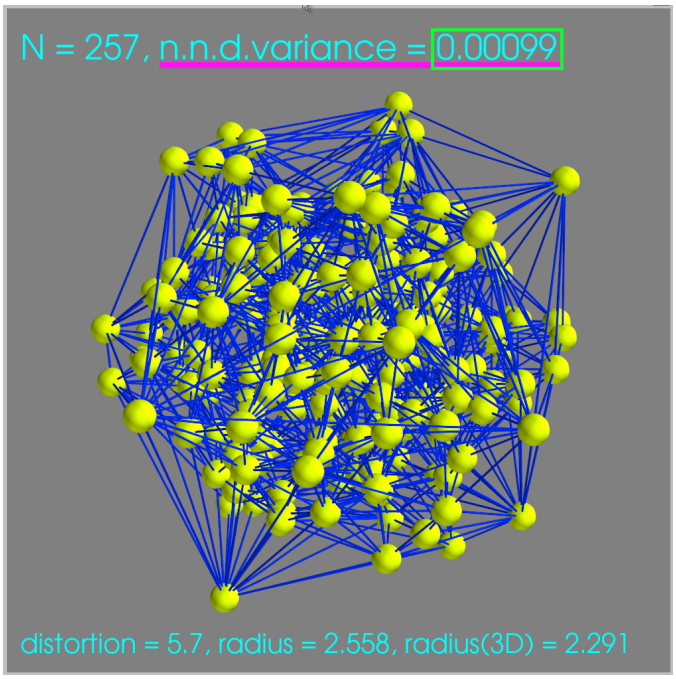
$$N \gtrsim \mathcal{N}^{\mathcal{G}}(8) \approx 240$$

12D on dimensions (1,2,3)



$$N \ll \mathcal{N}^{\mathcal{G}}(12) \lessgtr [840, 1356]$$

12D on dimensions (10,11,12)



$$N \ll \mathcal{N}^{\mathcal{G}}(12) \lessgtr [840, 1356]$$

Note the low nearest neighbour distance variance!

References

[Bac22] F. Bach.
On the unreasonable effectiveness of Richardson extrapolation.
https://francisbach.com/richardson-extrapolation, 2022.

[Coh18] H. Cohn.
Kissing numbers, 2018.
[Online; accessed 12-October-2023].

[Con10] R. Cont, editor.
Encyclopedia of Quantitative Finance.
John Wiley and Sons, 2010.
www.wiley.com/go/eqf.

[Cur58] P.C.J. Curtis.
n-parameter families and best approximation.
Pacific J. Math., 93:1013–1027, 1958.

[FF15a] B. Fornberg and N. Flyer.
Radial Basis Function Generated Finite Differences (RBF-FD): New Computational Opportunities for Solving PDEs, 2015.
amath.colorado.edu/faculty/fornberg/Docs/2015%20Oxford.pdf.

[FF15b] B. Fornberg and N. Flyer.
Solving PDEs with radial basis functions.
Acta Numerica, 24:215–258, 2015.
amath.colorado.edu/faculty/fornberg/Docs/2015_FF_Solving%20PDEs%20with%20RBFs_Acta_Numerica.pdf.

[FP07] B. Fornberg and C. Piret.
A stable algorithm for flat radial basis functions on a sphere.
SIAM J. Sci. Comput., 30:60–80, 2007.

[Fra79] Richard Franke.
A critical comparison of some methods for interpolation of scattered data.
Technical report, Monterey, California: Naval Postgraduate School., 1979.
www.dtic.mil/dtic/tr/fulltext/u2/a081688.pdf.

[FWL04] B. Fornberg, G. Wright, and E. Larsson.
Some observations regarding interpolants in the limit of flat radial basis functions.
Comput. Math. Appl., 47:37–55, 2004.

[FZ07] B. Fornberg and J. Zuev.
The Runge phenomenon and spatially variable shape parameters in RBF interpolation.
Computers & Mathematics with Applications, 54(3):379–398, 2007.

[Ham10] G. Hamerly.
Making k-means even faster.
In *Proceedings of the 2010 SIAM International Conference on Data Mining*, pages 130–140, 2010.

[Har71] R.L. Hardy.
Multiquadric equations of topography and other irregular surfaces.
Journal of Geophysical Research, 76(8):1905–1915, 1971.

[JA10] P. Jäckel and L. Andersen.
Exercise boundary optimization methods.
In Cont [Con10].
www.jaeckel.org/ExerciseBoundaryOptimization.pdf.

[Jäc13] P. Jäckel.
Finite differencing schemes as Padé approximants, April 2013.
www.jaeckel.org/FiniteDifferencingSchemesAsPad%C3%A9Approximants.pdf.

[Jäc17] P. Jäckel.
Cluster Induction.
WBS Fixed Income Conference Florence (September 2017) and QuantMinds Lisbon (May 2018), 2017.
www.jaeckel.org/ClusterInduction(presentation_version).pdf and www.jaeckel.org/ClusterInduction(print_version).pdf.

[Kan90a] E.J. Kansa.
Multiquadrics - A scattered data approximation scheme with applications to computational fluid-dynamics—I surface approximations and partial derivative estimates.
Computers & Mathematics with Applications, 19(8–9):127–145, 1990.

[Kan90b] E.J. Kansa.
Multiquadrics - A scattered data approximation scheme with applications to computational fluid-dynamics—II solutions to parabolic, hyperbolic and elliptic partial differential equations.
Computers & Mathematics with Applications, 19(8–9):147–161, 1990.

[Kis18] Kissing number.
Kissing number — Wikipedia, 2018.
[Online; accessed 12-October-2023].

[Llo57] SP. Lloyd.
Least Squares Quantization in PCM.
Technical report, Bell Laboratories, 1957.
Unpublished Bell Laboratories Technical Note. Portions presented at the Institute of Mathematical Statistics Meeting Atlantic City New Jersey September 1957. Published in the March 1982 special issue on quantization of the IEEE Transactions on Information Theory.

[Llo18] Lloyd's algorithm.
Lloyd's algorithm — Wikipedia, 2018.
[Online; accessed 02-March-2018].

[Mai56] J.C. Mairhuber.
On Haar's theorem concerning Chebyshev approximation problems having unique solutions.
Proc. Amer. Math. Soc, 7:609–615, 1956.

[Mic84] C.A. Micchelli.
Interpolation of scattered data: distance matrices and conditionally positive definite functions.
In *Approximation theory and spline functions*, pages 143–145. Springer, 1984.

[PD51] PF. Panter and W. Dite.
Quantization distortion in pulse-count modulation with nonuniform spacing of levels.
Proc. I.R.E., 39, 1 1951.

[Sha19] S.B. Shaw.
Radial Basis Function Finite Difference Approximations of the Laplace-Beltrami Operator.
Master's thesis, Boise State University, 2019.
https://scholarworks.boisestate.edu/cgi/viewcontent.cgi?article=2711&context=td.