Cluster Induction

Peter Jäckel

May 2018

Outline

Introduction

- CVA, FVA, and other XVA calculations, even in their simplest form, all involve calculations that require some kind of conditional expectation of a contract's future (deflated) value $v(t, \boldsymbol{x})$ for some $t > 0$ with x being a suitable (Markovian) state vector subject to some conditioning, e.g., conditional on being positive:

$$F(t) = \mathrm{E}\left[(v(t, \boldsymbol{x}))_+ \cdot f(x)\right] \tag{2.1}$$

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

- This is formally a forward-conditional expectation:

$$\mathrm{E}\left[(v(t, \boldsymbol{x}))_+ \cdot f(x)\right] = \mathrm{E}\left[v(t, \boldsymbol{x}) \cdot \mathbf{1}_{\{v(t,\boldsymbol{x})>0\}} \cdot f(x)\right] \tag{2.2}$$

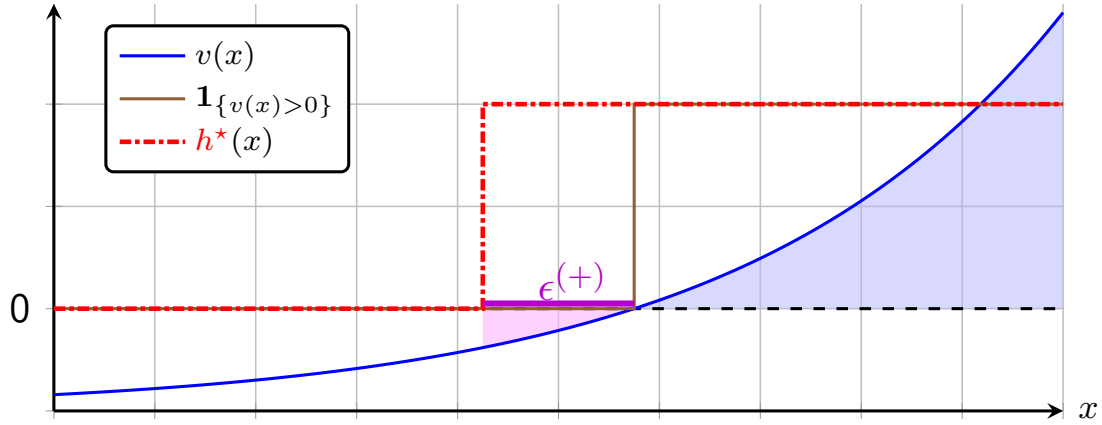
- In low dimensionalities, this can be computed directly with conventional finite differencing methods.

- In moderate or higher dimensions, it can, in principle, be computed with a Monte-Carlo simulation if at least some kind of Markovian approximation for the indicator function $h^\star(t, \boldsymbol{x}) \approx \mathbf{1}_{\{v(t,\boldsymbol{x})>0\}}$ can be found, i.e. $F^\star \approx F$ with

$$F^\star(t) = \mathrm{E}\left[v(t, \boldsymbol{x}) \cdot h^\star(t, \boldsymbol{x}) \cdot f(x)\right] \tag{2.3}$$

- Note that any small error ϵ in the domain in x on which the approximation for $h^\star(t, \boldsymbol{x})$ is 1 only enters the calculation in second order in ϵ.

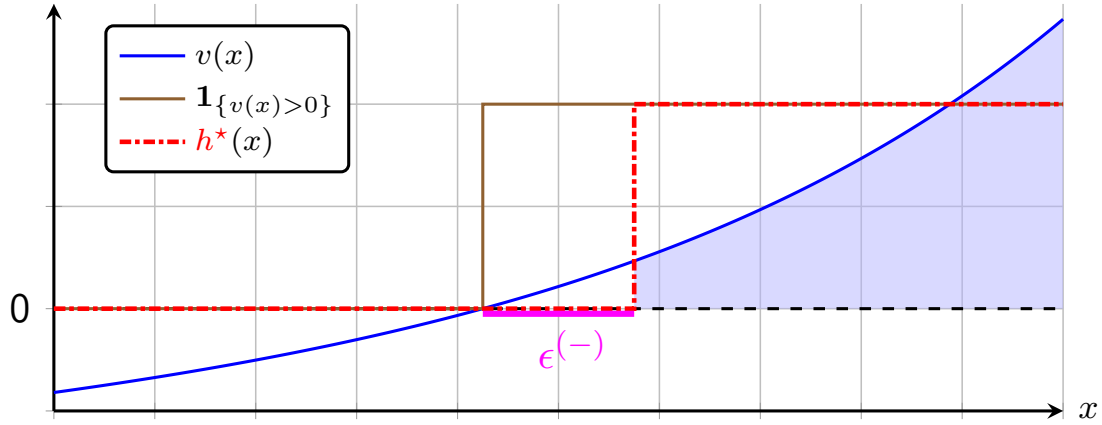
- To see this, consider that $h^\star(t,\boldsymbol{x}) = 1$ for $x \in \epsilon^{(+)}$ where $\epsilon^{(+)}$ is a small subdomain on which an accurate calculation would tell us that $v(t,\boldsymbol{x}) < 0$, but otherwise $h^\star(t,\boldsymbol{x})$ is telling us the truth.



In other words, $h^\star(t,\boldsymbol{x})$ is giving us some *false positives* and the calculation (2.3) will add up all of the required positive contributions, but also some erroneous negative contributions from $\epsilon^{(+)}$.

The net result is $F^\star < F$.

- Now consider the reverse situation that $h^\star(t,\boldsymbol{x}) = 0$ for $x \in \epsilon^{(-)}$ where $\epsilon^{(-)}$ is a small subdomain on which an accurate calculation would tell us that $v(t,\boldsymbol{x}) > 0$, but otherwise $h^\star(t,\boldsymbol{x})$ is telling us the truth.



In other words, $h^\star(t,\boldsymbol{x})$ is giving us some *false negatives* and the calculation (2.3) will not add up all of the required positive contributions since some positive contributions from $\epsilon^{(-)}$ will be missing.

The net result is $F^\star < F$.

- We see that any error ϵ in the domain where $h^\star(t,\boldsymbol{x})$ is positive leads to $F^\star < F$.

- Thus, $\epsilon = \{\}$ is a maximum for $F^\star$, i.e.,

$$\left.\frac{\partial F^\star}{\partial \epsilon}\right|_{\epsilon=\{\}} = 0\,. \tag{2.4}$$

- Thus, a local expansion of $F^\star$ in ϵ *leads with a second order term*.

- The conventional methodology to establish an approximate indicator function $h^\star(t,\boldsymbol{x})$ is to use a regression of path values of futures payments onto a chosen functional form, e.g., typically, a multivariate polynomial to approximate the value function itself, e.g.,

$$v^\star(t,\boldsymbol{x}) = \sum_{i,j} \alpha_{ijkl}\cdot x_i^k x_j^l \tag{2.5}$$

$$\alpha_{ijk\ell} = \arg\min_{\alpha_{ijkl}} \sum_{q=1}^{N} \left(v^\star(t,\boldsymbol{x}_q) - v_q(t)\right)^2 \tag{2.6}$$

$$v_q(t) = \text{(deflated) value of payments on path } q \text{ after t} \tag{2.7}$$

$$h^\star(t,\boldsymbol{x}) = \mathbf{1}_{\{v^\star(t,\boldsymbol{x})>0\}} \tag{2.8}$$

- Then, in a second simulation, $h^\star(t,\boldsymbol{x})$ is used to trigger the contribution of each path to approximate $F^\star$ by virtue of the tower law:

$$F^\star(t) = \mathrm{E}\left[h^\star(t,\boldsymbol{x})\cdot f(x)\cdot \mathrm{E}\left[v(T,\boldsymbol{x})|\mathcal{F}_t\right]\right] \tag{2.9}$$

- When the valuation of $v(t, \boldsymbol{x})$ itself involves callability, it may in turn require another layer of regression approximation(s).
- In (very) low dimensions, this works well.
- However, as the dimensionality increases, the technical aspects of the regressions to approximate the value function increase considerably, not least because:-
  - the number n of coefficients grows fast with the dimensionality d:

    $$n = \left(\begin{array}{c} d+k \\ k \end{array} \right)$$

    with k being the maximum total power of the polynomial.
  - the shape of $v(t, \boldsymbol{x})$ may be poorly matched by the used basis function set.
  - the regression effort grows like $\mathcal{O}(\max(m, n) \cdot n^2)$ for m sample paths.

Scattered (cluster) data interpolation

Let's recapitulate:-

- Finite differencing methods suffer the curse of dimensionality due to their requirement of a lattice of points.
- Monte Carlo regressions (e.g., onto polynomials) become awkward in high dimensions.

  An important aspect of this is the difficulty to choose the right basis function set, or a *suitable interpolation candidate*.

Engineers have long grappled with the issue of (multivariate) interpolation to fit non-trivial topologies over scattered nodes.

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.*

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).

This was an important breakthrough since the Mairhuber-Curtis theorem [Mai56, Cur58] declared that, in two or more dimensions, a generic interpolant ***leads to a singular system*** for infinitely many configurations of interpolation nodes.

The crucial factor that leads to the linear system being always regular is that

the basis functions must depend on the interpolation data!

$$v^\star(t,\boldsymbol{x}) = \sum_{s=1}^{m} c_s \cdot \psi_s(\boldsymbol{x}) \tag{3.1}$$

REPEAT:

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

This, and the radial nature of the multiquadric hyperboloids, led subsequently to a flurry of developments of

Radial Basis Functions.

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

Radial Basis Functions

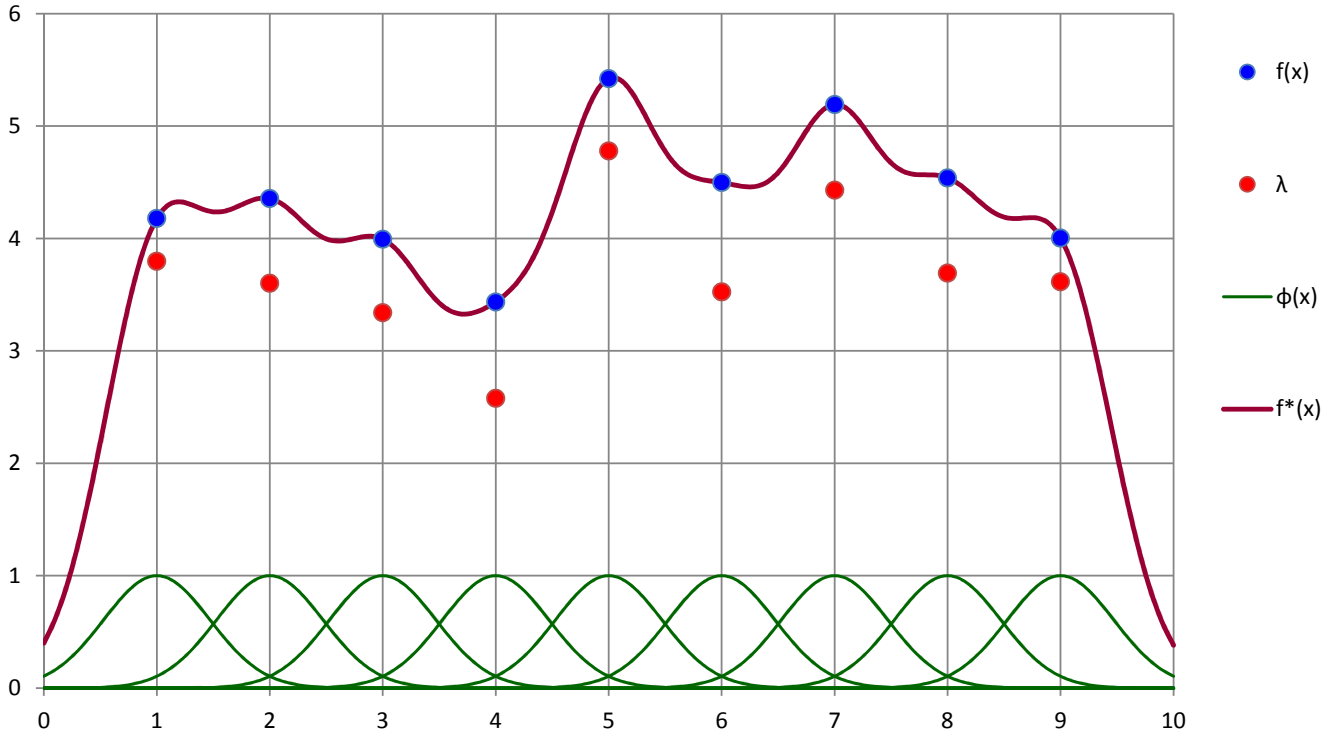
A radial basis function decomposition for a function $f(\boldsymbol{x}) : \mathbb{R}^d \rightarrow \mathbb{R}$ is typically defined by

$$f^\star(\boldsymbol{x}) = \sum_{j=1}^{m} \lambda_j \cdot \phi(|\boldsymbol{x} - \boldsymbol{x}_j|; \epsilon_j) \tag{3.2}$$

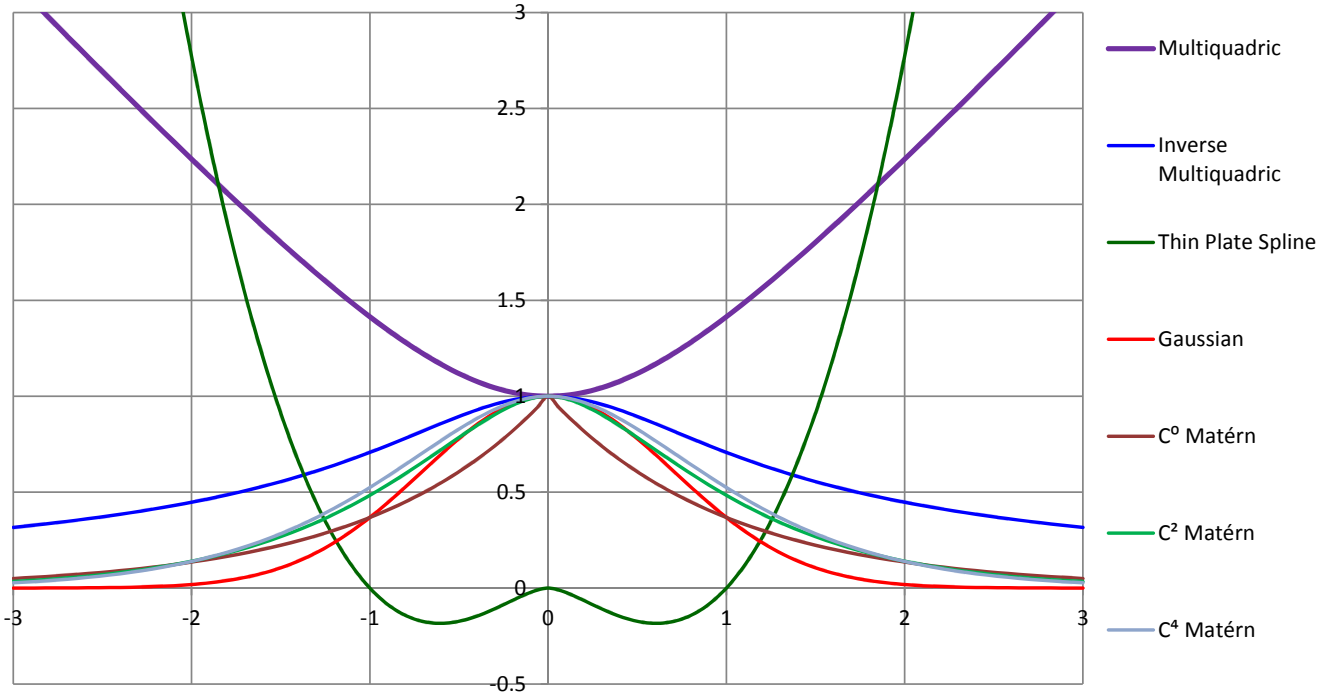
with $\phi(r; \epsilon)$ being one of:-

Multiquadric [Har71]	$\sqrt{1 + (\epsilon r)^2}$
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	$\mathrm{e}^{-\sqrt{5}\lvert\epsilon r\rvert} \cdot (1 + \sqrt{5}\lvert\epsilon r\rvert + 5(\epsilon r)^2/3)$

Anyone coming from a background of conventional finite differencing or finite elements, might be forgiven to expect something like this to be the result of the decomposition (seen as a 1D interpolation here):



So you may be forgiven to be somewhat suprised that the so praised *multi-quadric* basis is the complete opposite of a what a locally confined function would look like:



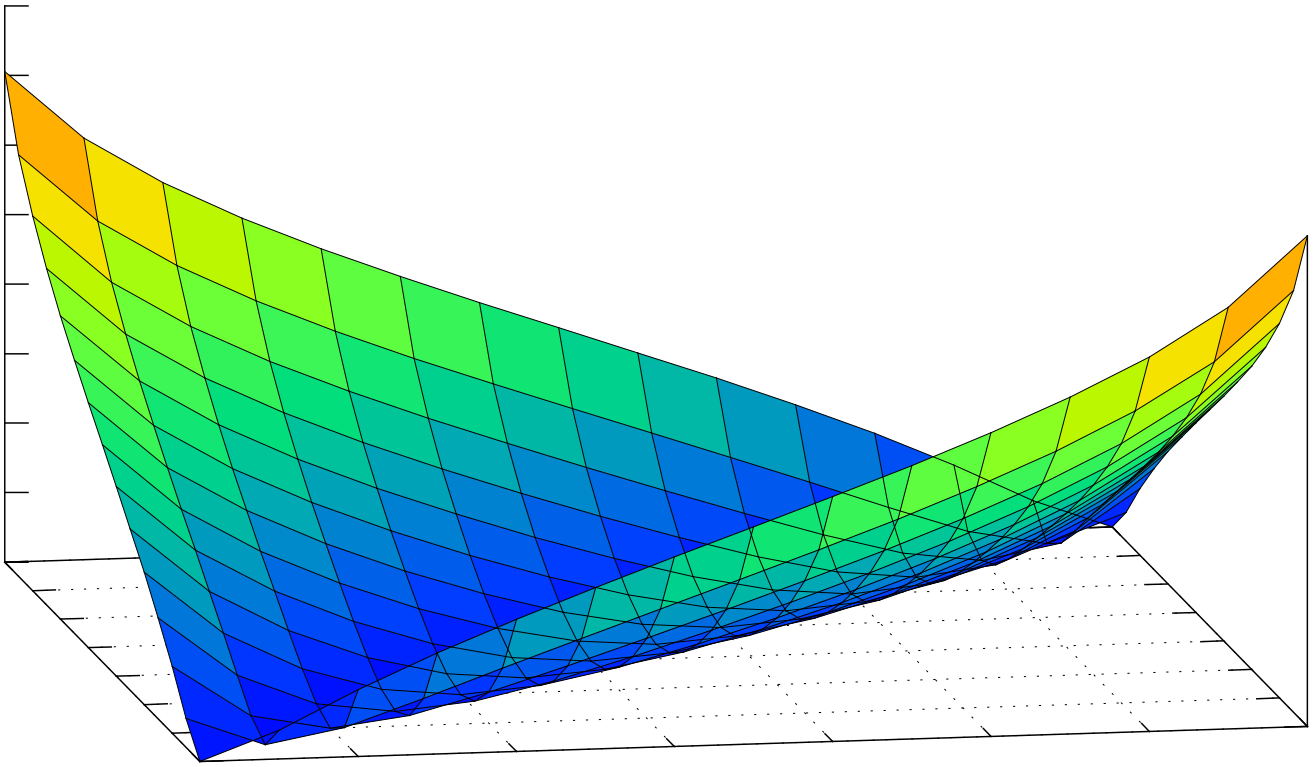
Scattered data interpolation over n cluster points is (usually) accomplished by centering n basis functions in the cluster data locations themselves, and solving the linear system

$$\Phi \cdot \boldsymbol{\lambda} \;=\; \boldsymbol{f} \tag{3.3}$$

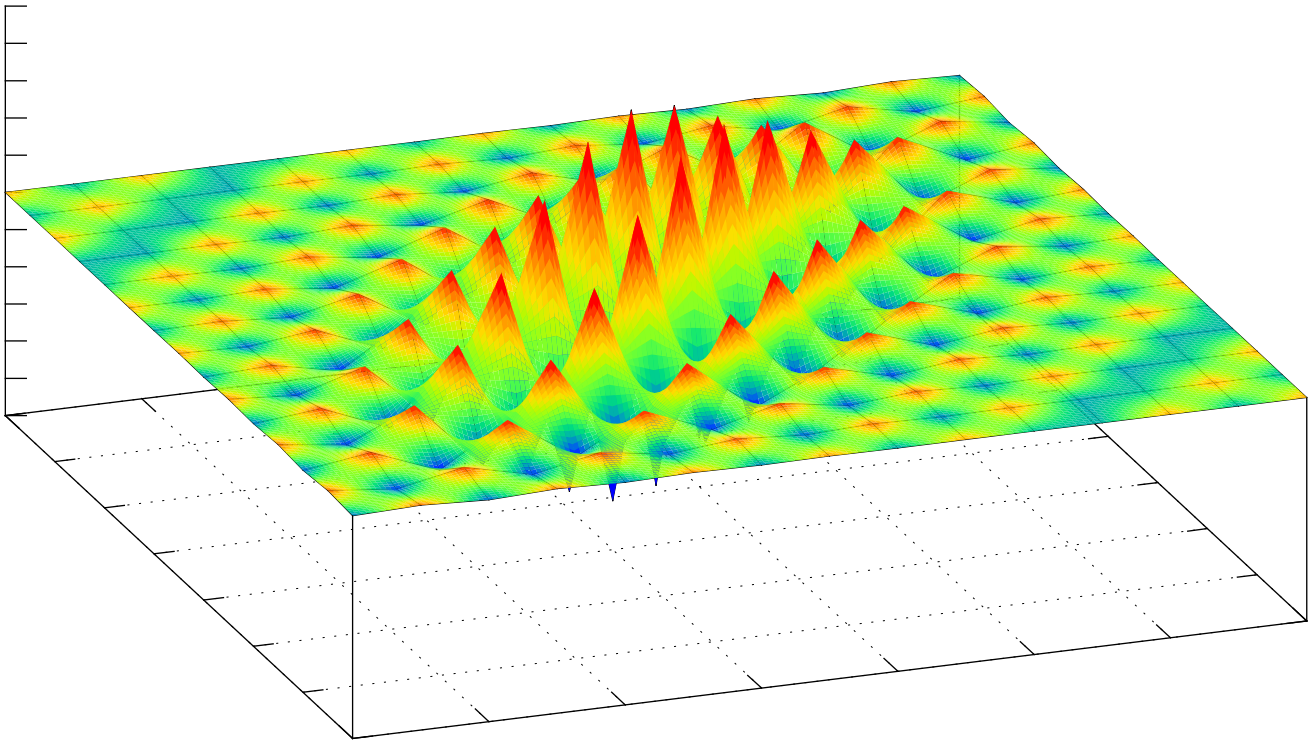
with

$$(\Phi)_{ij} \;=\; \phi(|\boldsymbol{x}_i - \boldsymbol{x}_j|; \epsilon_j) \tag{3.4}$$

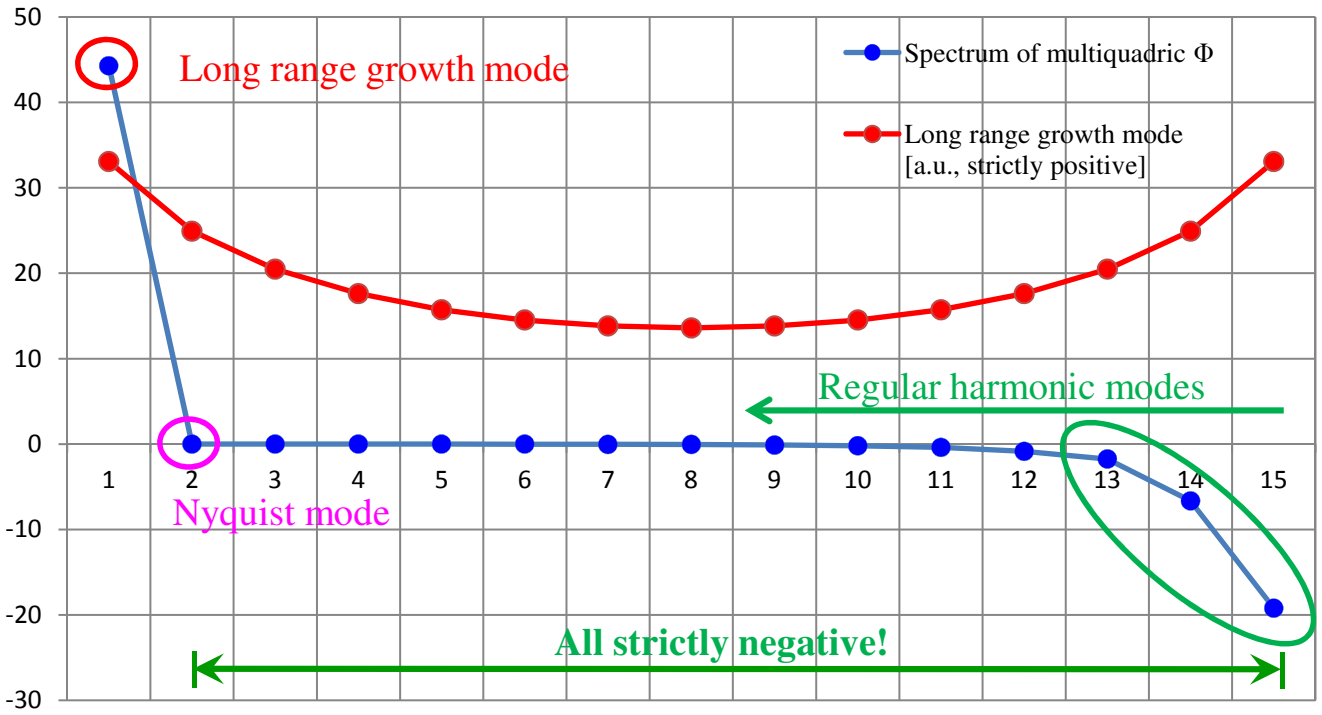
Multiquadric Φ matrix ($\epsilon = 2.6$) for 15 Gaussian Sobol' nodes in 1D:



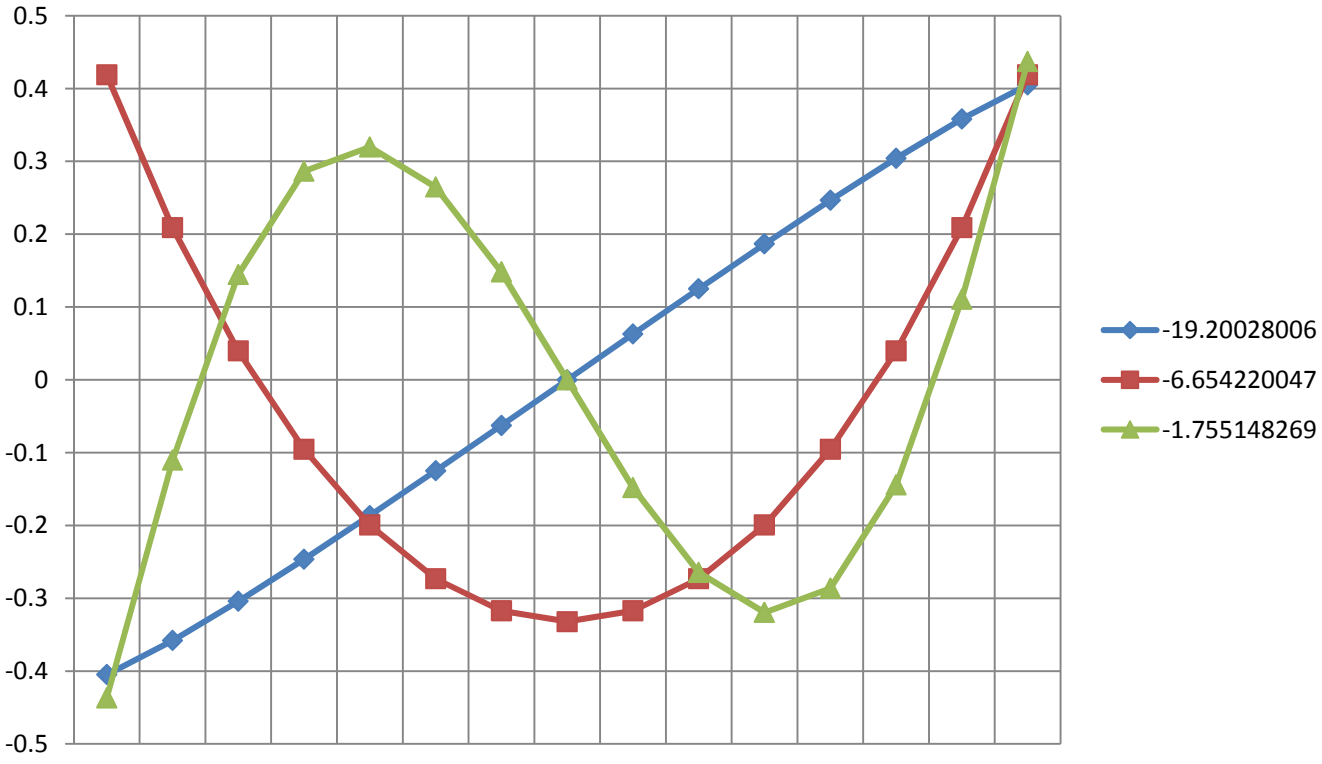
Multiquadric Φ^{-1} matrix for 15 Gaussian Sobol' nodes in 1D:



The spectrum of the multiquadric Φ matrix:

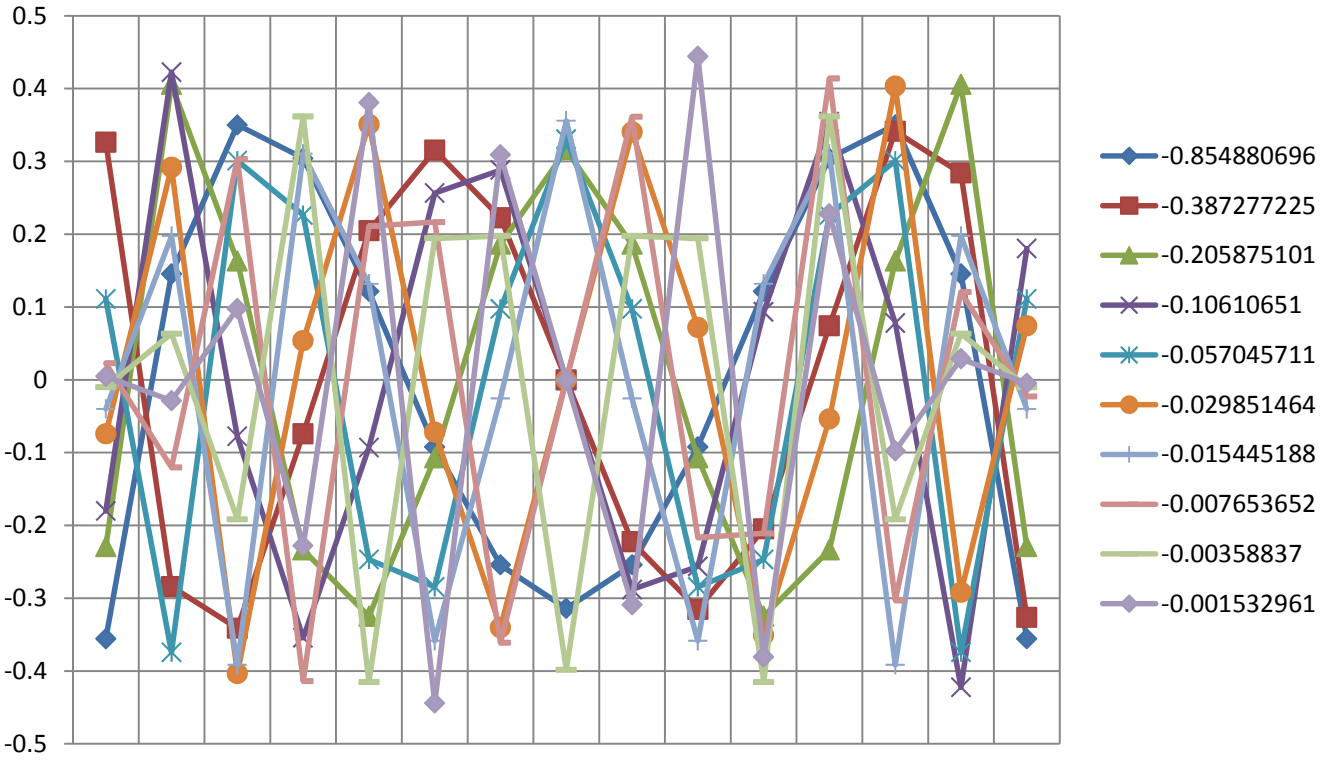


The first three harmonic modes of Φ and their eigenvalues:



The Nyquist mode of Φ:

And all other modes of Φ in between:



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

$$|f(x) - f^*(x)| \sim \mathcal{O}\left(\mathrm{e}^{-c/h}\right) \tag{3.5}$$

which is obviously much faster than any power convergence where

$$|f(x) - f^*(x)| \sim \mathcal{O}\left(h^p\right) \tag{3.6}$$

for any p.

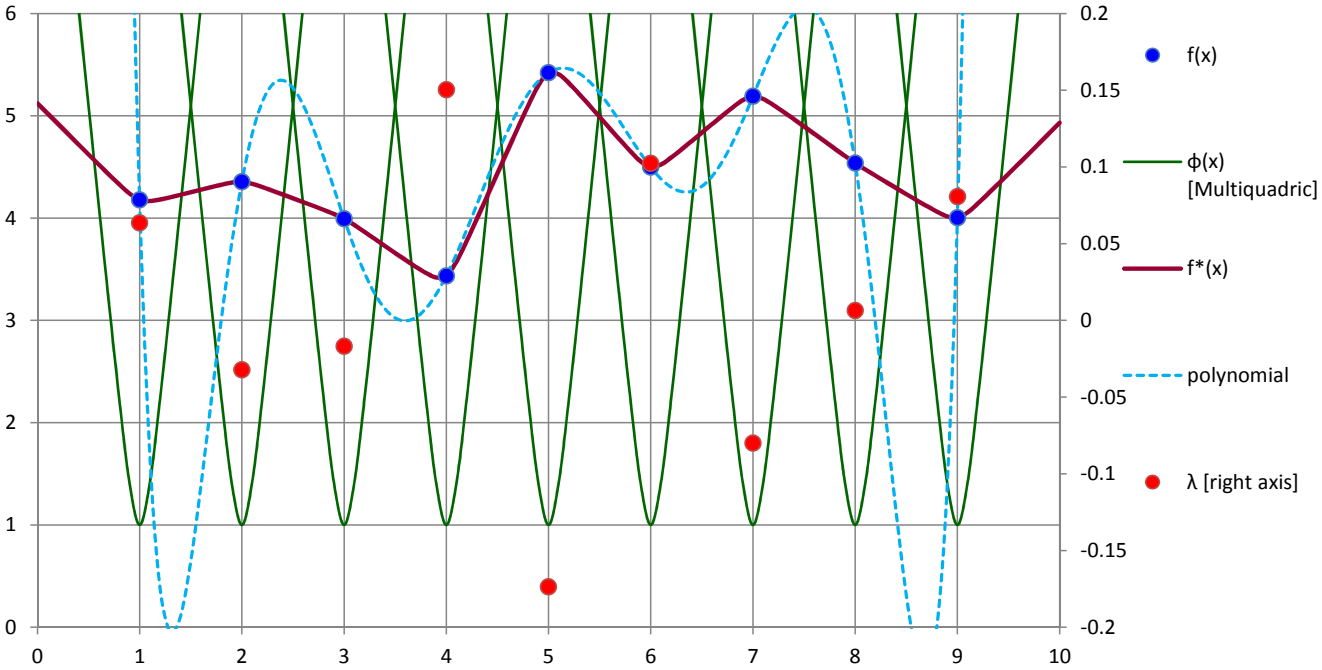
This result was derived more than twenty years after Hardy picked them out as the best practical choice for scattered data interpolation!

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].

These findings finally justified why in practice they tend to be the best choice (and this also holds for higher dimensions).

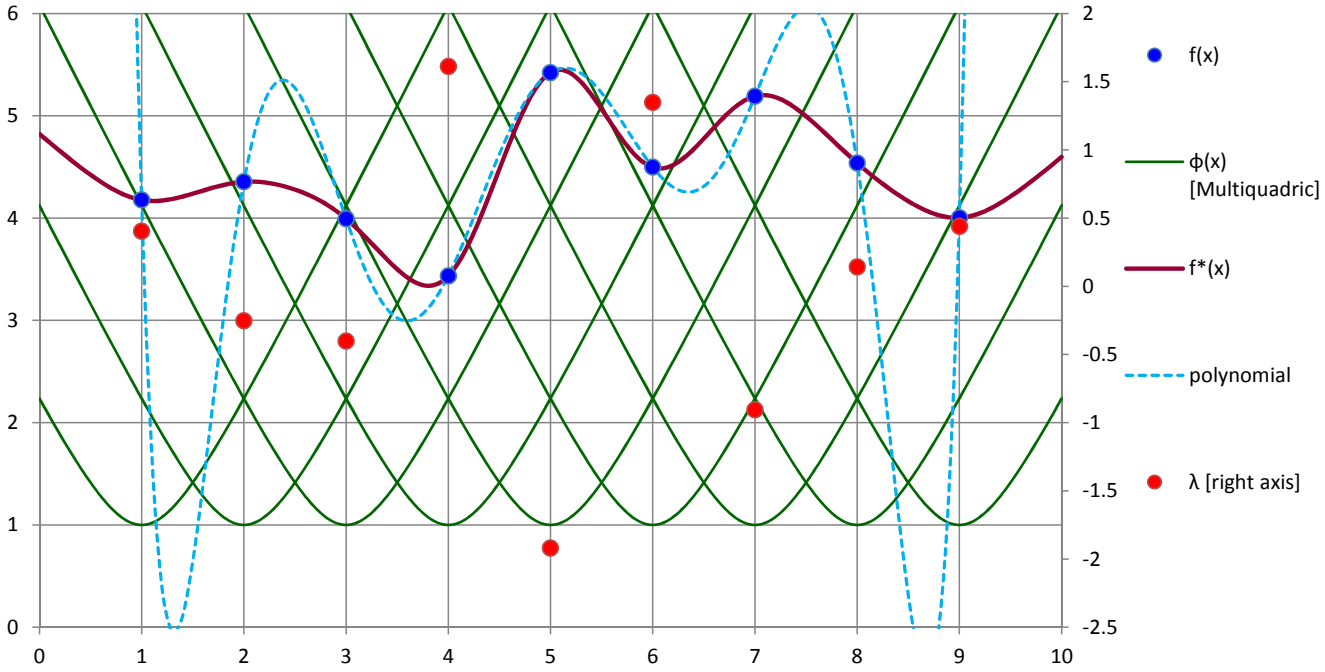
RBF-interpolation converges to *polynomial interpolation* as $\epsilon \to 0$.

$\epsilon = 10$:



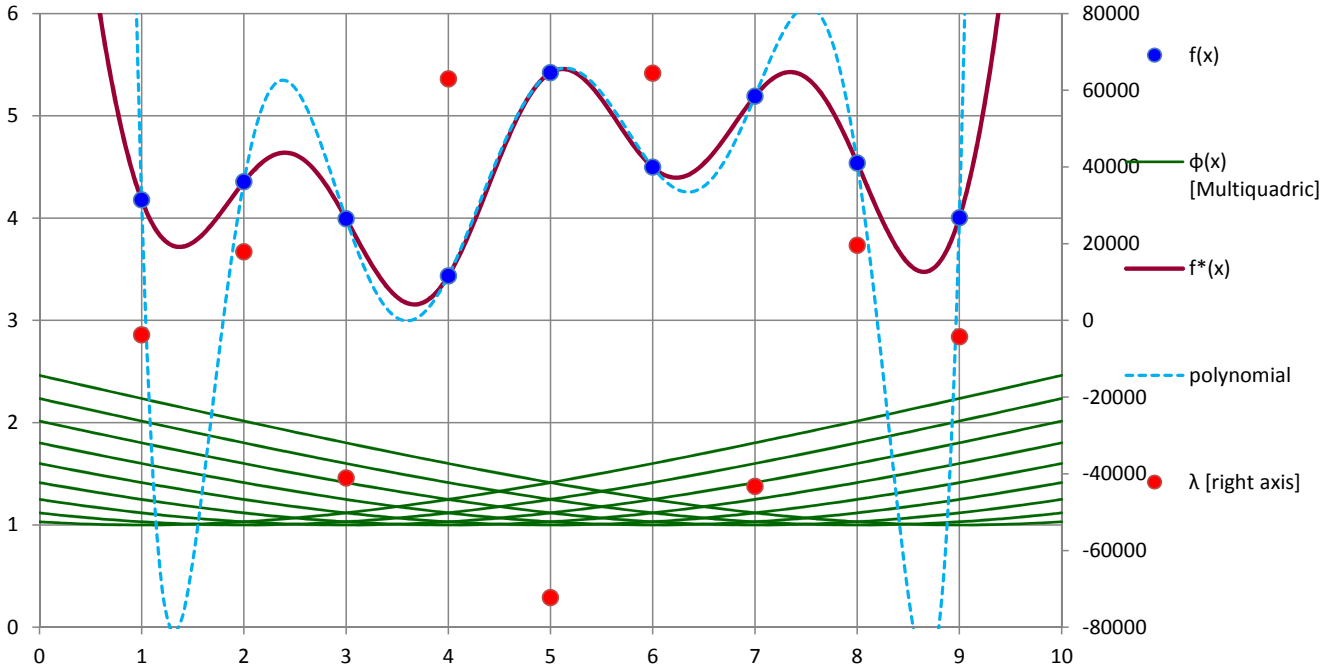
Reciprocal condition number $\kappa(\Phi)^{-1} = 0.0152$.

$\epsilon = 2$:



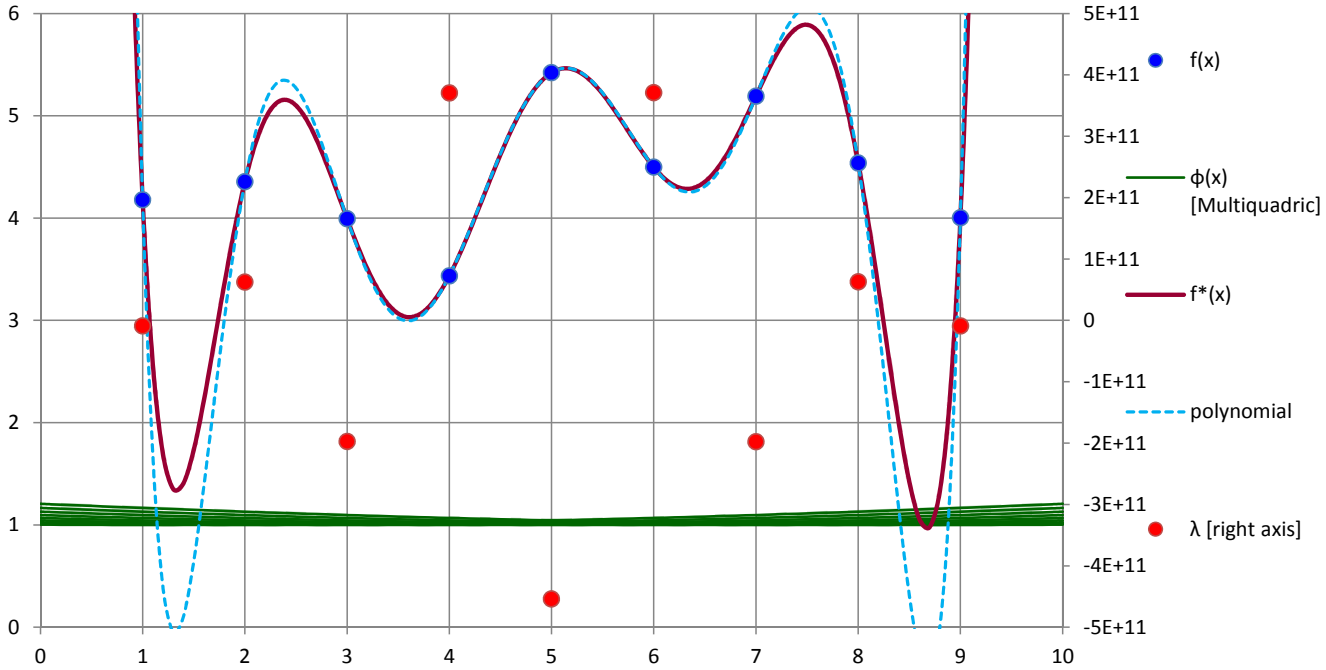
Reciprocal condition number $\kappa(\Phi)^{-1} = 0.0061$.

$\epsilon = 0.25$:



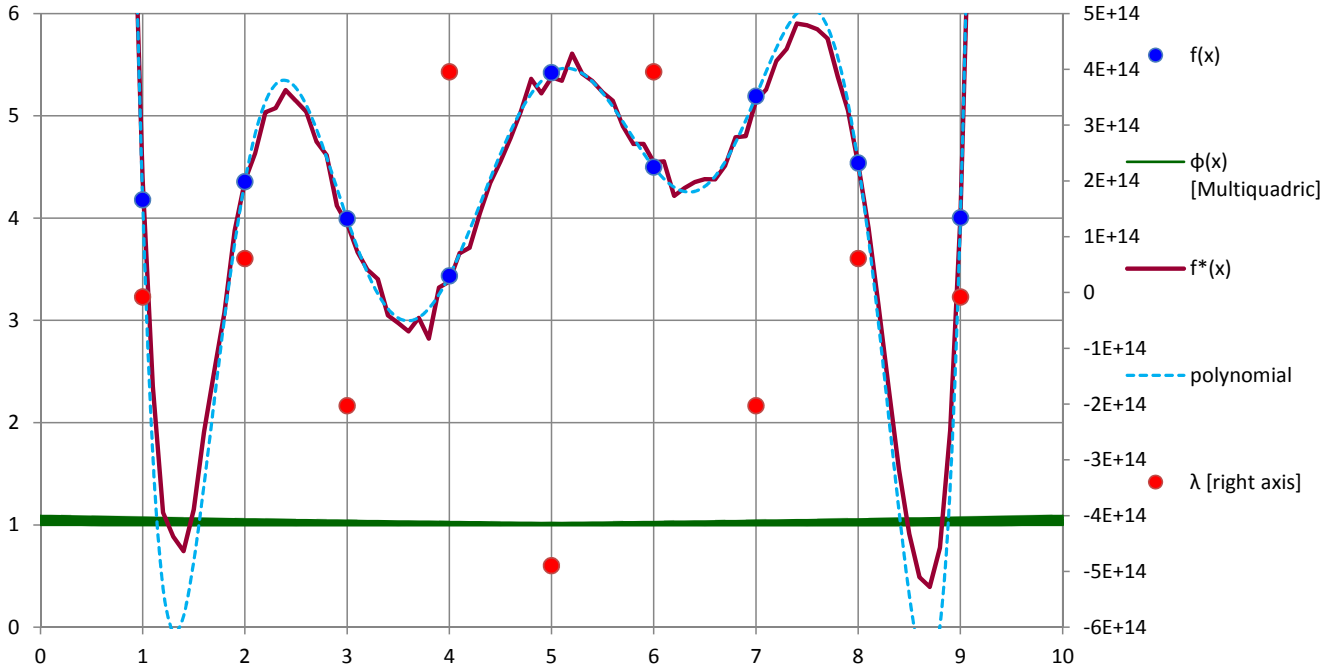
Reciprocal condition number $\kappa(\Phi)^{-1} = 6.65\text{E-}07$.

$\epsilon = 0.075$:



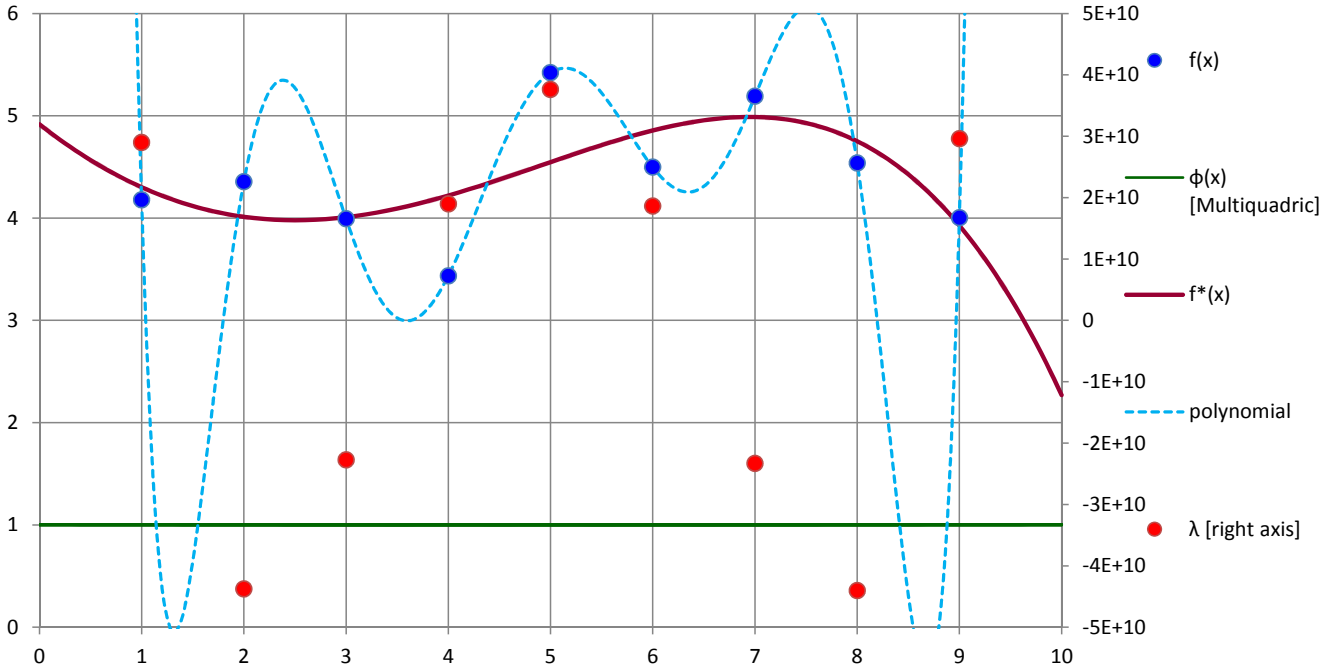
Reciprocal condition number $\kappa(\Phi)^{-1} = 1.61\text{E-}13$.

$\epsilon = 0.0475$:



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 also referred to as the *flat limit*.
- 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...

- The Moore-Penrose (SVD) solution must be used.
  Use Lapack's "Divide-and-Conquer" algorithm (DGELSD).

Meshless induction

Obviously, it is possible to use Radial Basis Functions for the purpose of a Monte Carlo regression, to evaluate any callability or future conditional value, such as for the XVA valuation in (2.2).

However, that's not what this presentation is about.

Instead, we try to go one step further.

Wouldn't it be nice if we could not only interpolate over scattered data, but **directly use such a *cluster of scattered nodes* as base locations for a backward induction of a partial differential equation?**

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.

Armed with this result on the partial derivatives, Kansa [Kan90b] then proceeded as follows.

Given an advection-diffusion (forward) PDE

$$\partial_t f + u \cdot \partial_x f - D \cdot \partial_x^2 f = 0 \tag{4.1}$$

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.

Changing notation to a generic spatial differential operator $\mathcal{L}$ in d dimensions, we obtain

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

for the backward Kolmogorov (pricing) equation.

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

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

i.e., a

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

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

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.

Crunching out the analytics...

In a practical implementation, it is useful to define the cluster vertices in terms of some 'cluster coordinate system' that *scales away any relative magnitude differences* of the individual state space coordinates, i.e., the elements of the state vector $\boldsymbol{x} \in \mathbb{R}^d$.

Consider then a given coordinate-scaling diagonal matrix Ω such that we have the transformation from the cluster coordinates $\boldsymbol{\xi}$ to state space coordinates $\boldsymbol{x}$ via

$$\boldsymbol{x} = \Omega \cdot \boldsymbol{\xi} \tag{4.9}$$

Obviously, the same scaling applies to the cluster vertices, i.e., the basis function centres:

$$\boldsymbol{x}_i = \Omega \cdot \boldsymbol{\xi}_i \tag{4.10}$$

for $i = 1, .., n$ for n cluster points.

We then only ever evaluate all basis functions in terms of cluster coordinates, i.e. the j-th basis function as a function of $\boldsymbol{x}$ evaluates according to

$$\phi_j(\boldsymbol{x}) := \phi(|\boldsymbol{\xi} - \boldsymbol{\xi}_j|, \epsilon_j) \tag{4.11}$$

with

$$\boldsymbol{\xi} = \frac{\boldsymbol{x}}{\Omega} \tag{4.12}$$

where $\phi(r; \epsilon)$ is one of the fundamental radial basis functions, e.g., a multi-quadric (which is really a hyperbola and not a 'quadr[at]ic' function).

For the evaluation of the operator $\mathcal{L}$ applied to the radial basis functions, we note then

$$\frac{\partial}{\partial x_k} = \frac{1}{\omega_k} \cdot \frac{\partial}{\partial \xi_k} \tag{4.13}$$

for $k = 1, .., d$.

For a generic advection-diffusion generator with advection coefficients that are linear in the state variables, we obtain

$$\mathcal{L} = \frac{1}{2}\sum_{k,l=1}^{d} \sigma_k \cdot \rho_{kl} \cdot \sigma_l \cdot \frac{\partial^2}{\partial x_k \partial x_l} + \sum_{k,l=1}^{d} x_k \cdot \varkappa_{kl} \cdot \frac{\partial}{\partial x_l} \tag{4.14}$$

$$= \frac{1}{2}\sum_{k,l=1}^{d} \rho'_{kl} \cdot \frac{\partial^2}{\partial \xi_k \partial \xi_l} + \sum_{k,l=1}^{d} \xi_k \cdot \varkappa'_{kl} \cdot \frac{\partial}{\partial \xi_l} \tag{4.15}$$

with

$$\rho'_{kl} := \frac{\sigma_k}{\omega_k} \cdot \rho_{kl} \cdot \frac{\sigma_l}{\omega_l} \tag{4.16}$$

$$\varkappa'_{kl} := \frac{\omega_k}{\omega_l} \cdot \varkappa_{kl} \tag{4.17}$$

though we note that *we will drop the primes in the following*.

After a considerable amount of tedious calculations, we arrive at the following condensed expressions for $\mathcal{L}$ applied to $\phi(|\Delta\boldsymbol{\xi}|, \epsilon)$, with $\Delta\boldsymbol{\xi} := \boldsymbol{\xi} - \boldsymbol{\xi}^*$, i.e., a radial basis function centered in some cluster node $\boldsymbol{\xi}^*$:-

$$\mathcal{L}\phi = \tfrac{1}{2}(\tilde{d}\cdot\alpha + \beta\cdot\gamma) + \alpha\cdot\chi_1 \tag{4.18}$$

$$\begin{aligned}\mathcal{L}^2\phi = {} & \tfrac{1}{4}\cdot\Big(\eta\cdot\gamma^2 + 2\cdot\zeta\cdot(\tilde{d}\cdot\gamma + 2\cdot\theta) + \beta\cdot(\tilde{d}^2 + 2\cdot\iota)\Big) \\ & + \tfrac{1}{2}\cdot\Big(2\cdot\alpha\cdot\tilde{d}_2 + \beta\cdot\tilde{d}\cdot\chi_1 + 2\cdot\beta\cdot(\chi_4+\chi_5) + \zeta\cdot\gamma\cdot\chi_1\Big) \\ & + \tfrac{1}{2}\cdot((\zeta\cdot\gamma + \beta\cdot\tilde{d})\cdot\chi_1 + 2\cdot\beta\cdot\chi_5) \\ & + \alpha\cdot(\chi_2+\chi_3) + \beta\cdot\chi_1^2\end{aligned} \tag{4.19}$$

where, with $(R)_{kl} := \rho_{kl}$, $(K)_{kl} := \varkappa_{kl}$, and $r := |\Delta\boldsymbol{\xi}|$,

$$
\begin{aligned}
\alpha &= \frac{\phi'}{r} & \gamma &= \Delta\boldsymbol{\xi}^\top \cdot R \cdot \Delta\boldsymbol{\xi} \\
\beta &= \frac{1}{r^2}\left(\phi'' - \frac{\phi'}{r}\right) & \theta &= \Delta\boldsymbol{\xi}^\top \cdot R^2 \cdot \Delta\boldsymbol{\xi} \\
\zeta &= \frac{1}{r^3}\left[\phi''' - \frac{3}{r}(\phi'' - \frac{\phi'}{r})\right] & \chi_1 &= \boldsymbol{\xi}^\top \cdot K \cdot \Delta\boldsymbol{\xi} \\
\eta &= \frac{1}{r^4}\left[\phi'''' - 6\frac{\phi'''}{r} + \frac{15}{r^2}(\phi'' - \frac{\phi'}{r})\right] & \chi_2 &= \boldsymbol{\xi}^\top \cdot K \cdot K^\top \cdot \boldsymbol{\xi} \\
\tilde{d} &= \mathrm{Tr}(R) & \chi_3 &= \boldsymbol{\xi}^\top \cdot K^2 \cdot \Delta\boldsymbol{\xi} \\
\iota &= \mathrm{Tr}(R^2) & \chi_4 &= \Delta\boldsymbol{\xi}^\top \cdot R \cdot K \cdot \Delta\boldsymbol{\xi} \\
\tilde{d}_2 &= \mathrm{Tr}(R \cdot K) & \chi_5 &= \boldsymbol{\xi}^\top \cdot K \cdot R \cdot \Delta\boldsymbol{\xi} \quad .
\end{aligned}
\tag{4.20}
$$

Only α, β, ζ, and η depend on the type of RBF!

Further, we obtain:-

$$
\begin{aligned}
&\text{when} & &\text{when} \\
\phi &= \sqrt{1+(\epsilon r)^2}: & \phi &= \mathrm{e}^{-(\epsilon r)^2}: \\
\alpha &= \frac{\epsilon^2}{\phi} & \alpha &= -2\cdot\epsilon^2\cdot\phi \\
\beta &= -\alpha\cdot\frac{\epsilon^2}{\phi^2} & \beta &= -2\cdot\epsilon^2\cdot\alpha \\
\zeta &= -3\cdot\beta\cdot\frac{\epsilon^2}{\phi^2} & \zeta &= -2\cdot\epsilon^2\cdot\beta \\
\eta &= -5\cdot\zeta\cdot\frac{\epsilon^2}{\phi^2} & \eta &= -2\cdot\epsilon^2\cdot\zeta
\end{aligned}
\tag{4.21}
$$

Before we proceed to numerical integration scheemes of our system of ODEs, we make a few observations.

If we assume the existence of Φ^{-1}, the system (4.8) can be written as

$$\dot{\boldsymbol{\lambda}}(t) + \left(\Phi^{-1} \cdot \mathcal{L}\Phi\right) \cdot \boldsymbol{\lambda}(t) \;=\; 0 \tag{4.22}$$

with formal solution

$$\boldsymbol{\lambda}(t - \Delta t) \;=\; \mathrm{e}^{\Delta t \cdot \left(\Phi^{-1} \cdot \mathcal{L}\Phi\right)} \cdot \boldsymbol{\lambda}(t)\;. \tag{4.23}$$

Equally, if we use

$$\Phi \cdot \boldsymbol{\lambda} \;=\; \boldsymbol{f} \qquad \Longleftrightarrow \qquad \boldsymbol{\lambda} \;=\; \Phi^{-1} \cdot \boldsymbol{f} \tag{4.24}$$

and define $G \;:=\; \mathcal{L}\Phi \cdot \Phi^{-1}$, we have

$$\dot{\boldsymbol{f}}(t) + G \cdot \boldsymbol{f}(t) \;=\; 0\;. \tag{4.25}$$

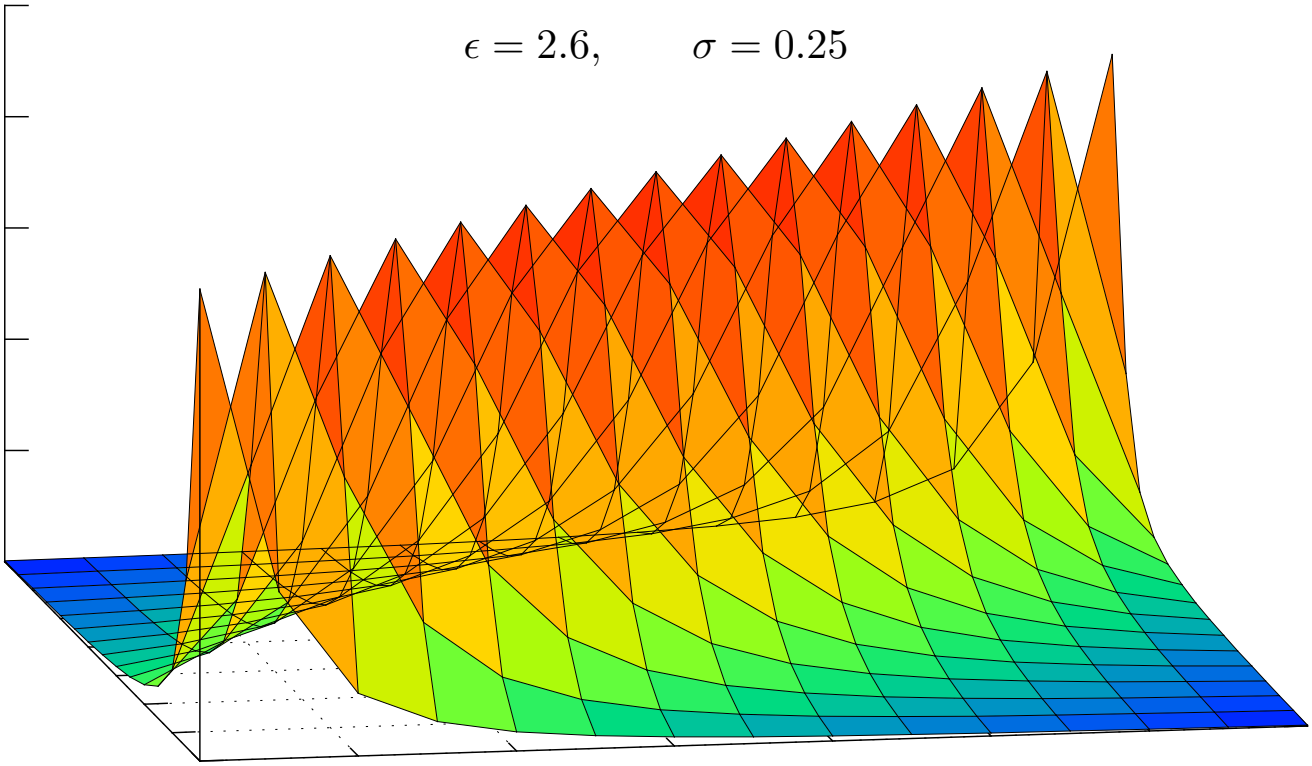
Note that (4.25) is ***a system of ODEs directly for the value vector $\boldsymbol{f}$!*** **This means that we can in principle carry out a backward induction directly on a set of arbitrarily scattered cluster vertices, without ever computing any form of local (finite-differencing or otherwise) partial derivatives from the value function!**

Finally, unsurprisingly, the formal solution of (4.25) is

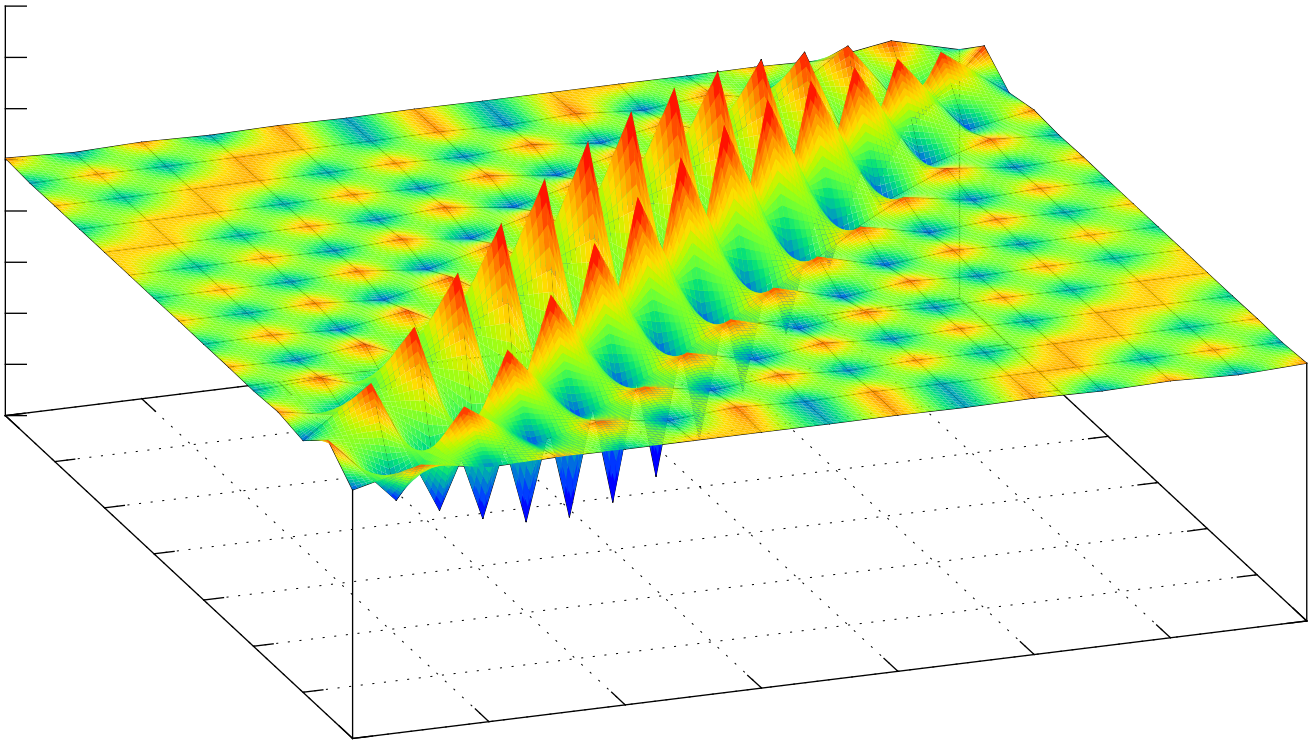
$$\boldsymbol{f}(t - \Delta t) \;=\; \mathrm{e}^{\Delta t \cdot G} \cdot \boldsymbol{f}(t)\;. \tag{4.26}$$

Remark: in practice, due to the need for SVD with cut-off, the use of G induces an unnecessary early projection onto the subspace of functions that can be represented by the basis.

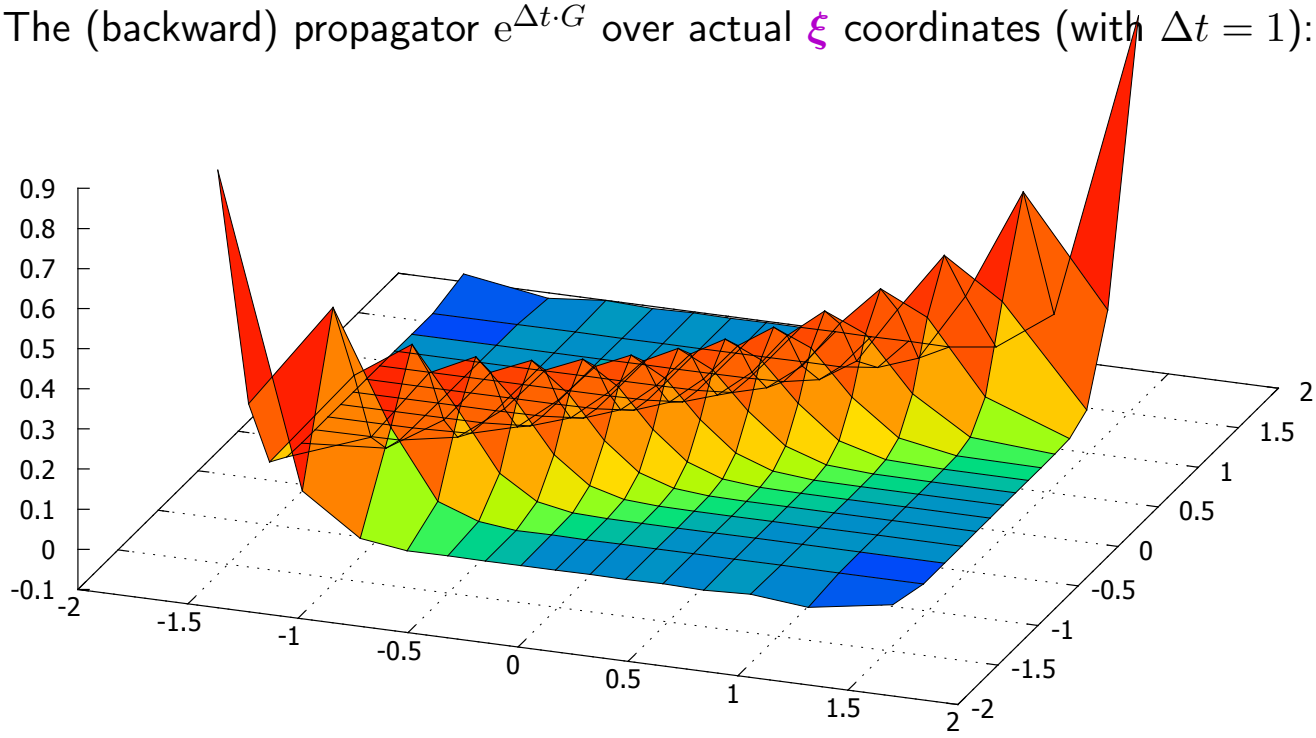
$\mathcal{L}\Phi$ matrix for 15 Gaussian Sobol' nodes in 1D (for a standard diffusion):



The generator matrix $G = \mathcal{L}\Phi \cdot \Phi^{-1}$:

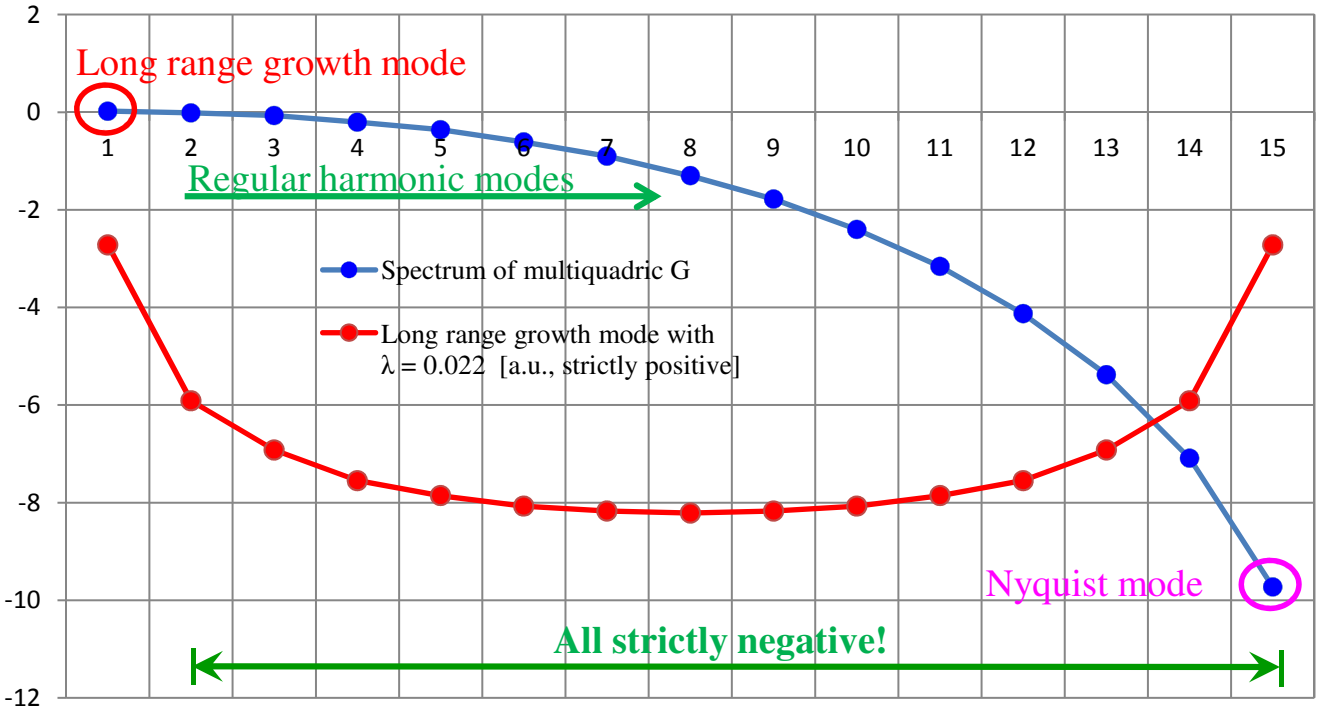


The (backward) propagator $\mathrm{e}^{\Delta t \cdot G}$ over actual $\boldsymbol{\xi}$ coordinates (with $\Delta t = 1$):

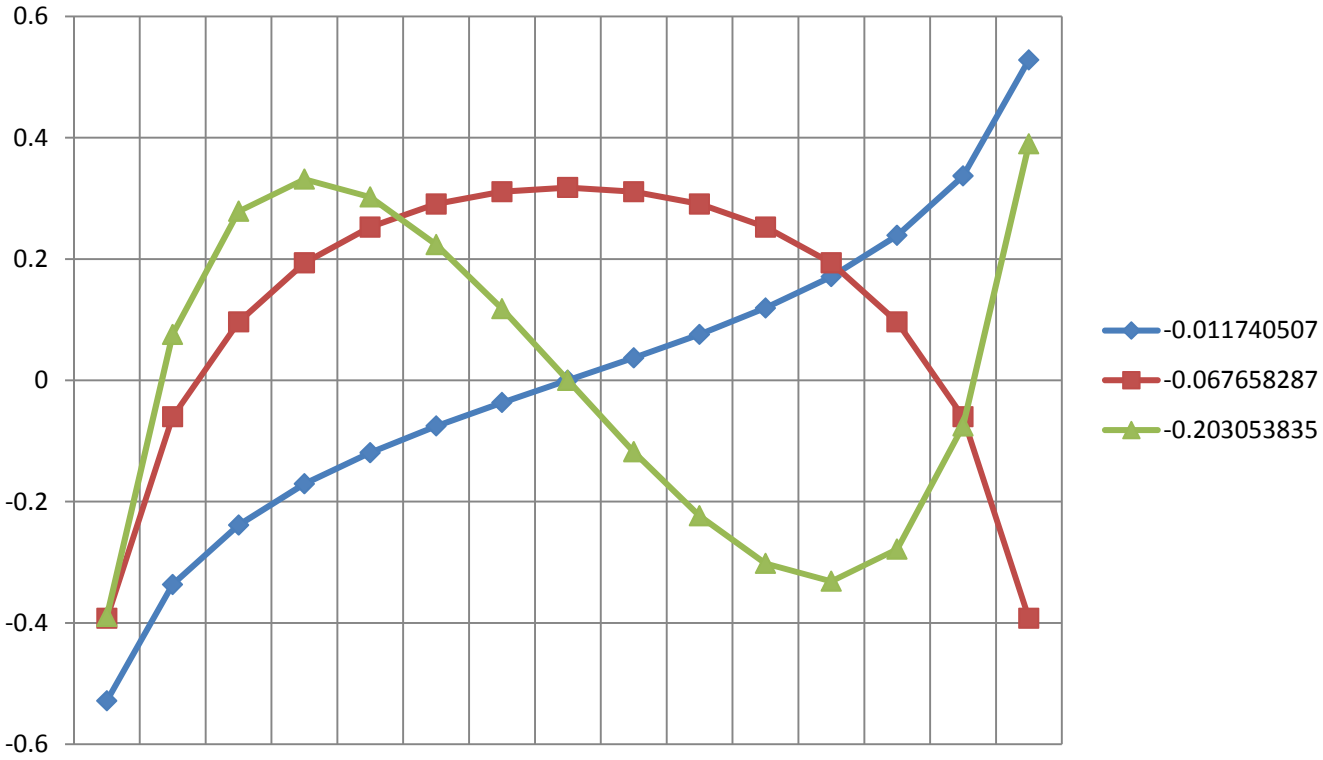


This looks like the typical Viking ship we expect for a proper transition probability matrix!!!

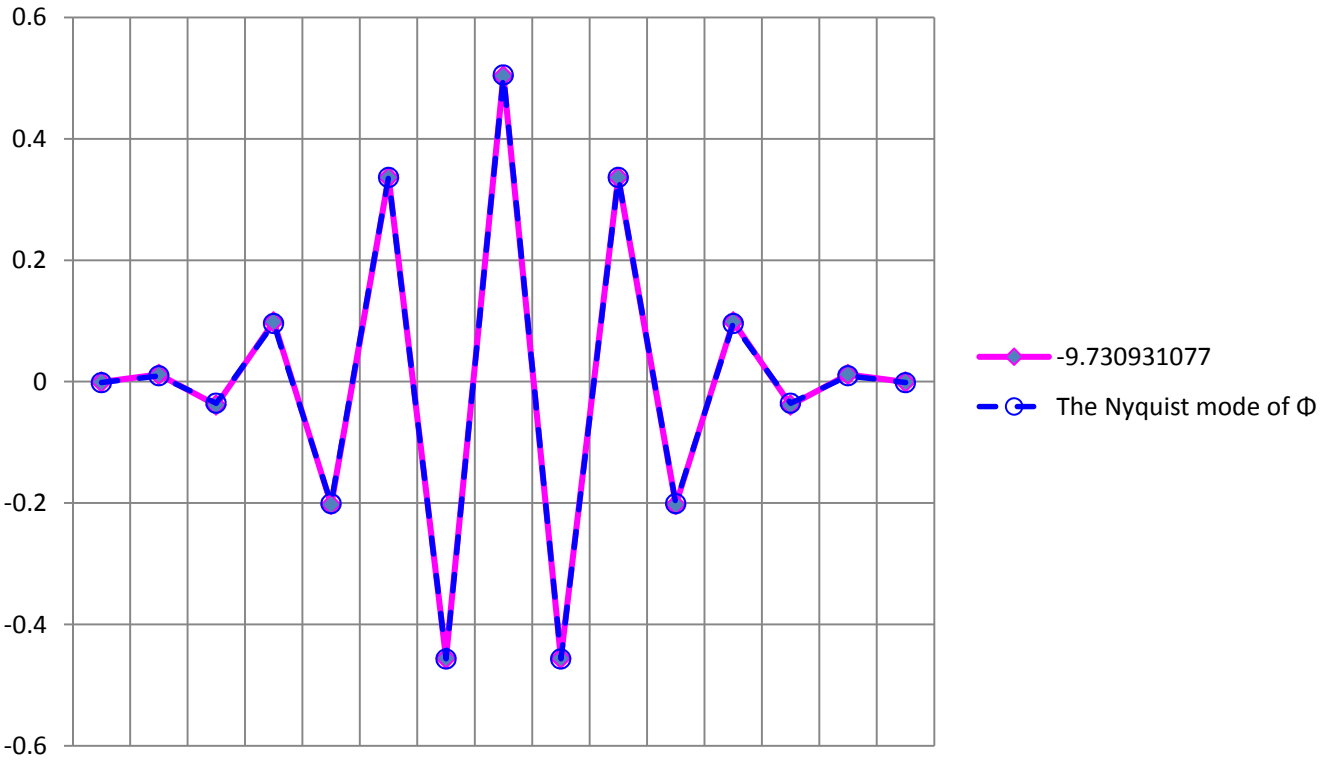
The spectrum of the multiquadric generator matrix $G = \mathcal{L}\Phi \cdot \Phi^{-1}$:



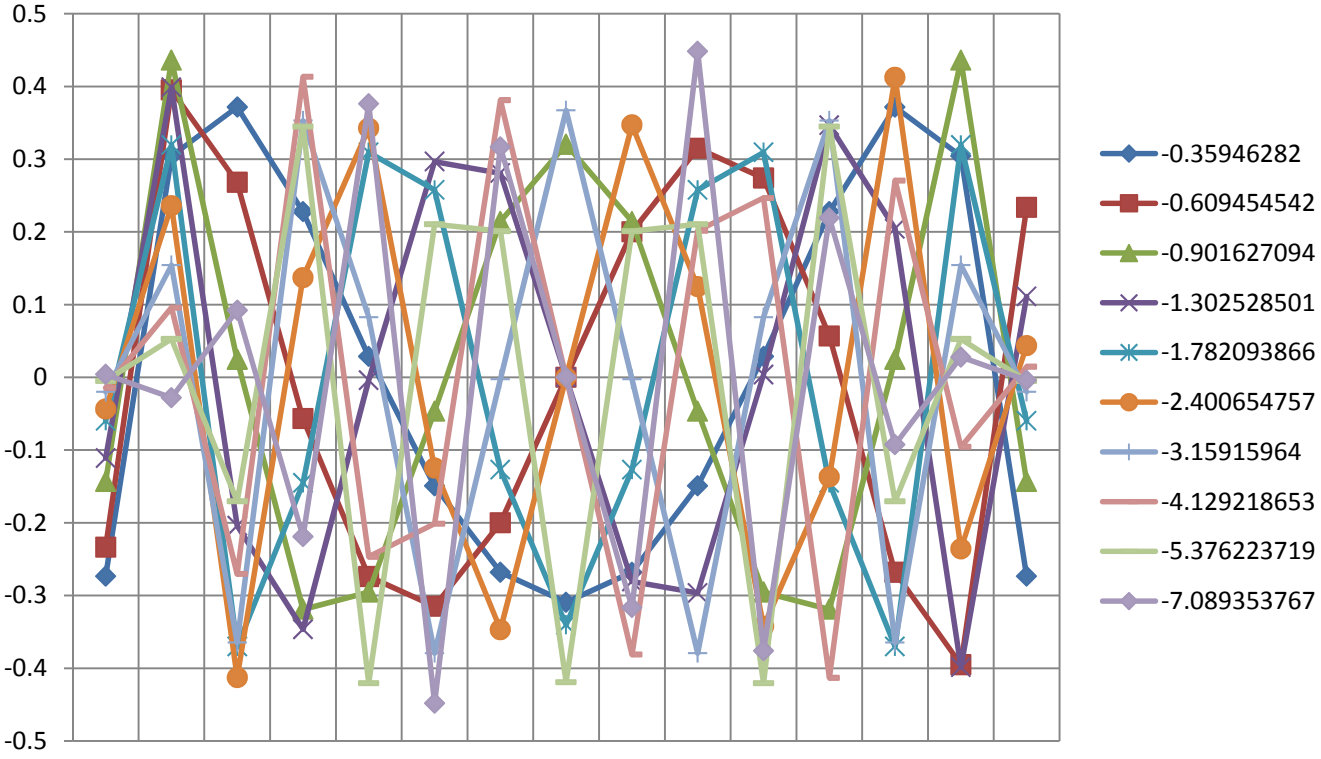
The first three harmonic modes of G and their eigenvalues:



The Nyquist mode of G (*not identical* to that of Φ):



And all other modes of G in between:



There's more to life than Crank-Nicolson

Despite there being all the mentioned possible ways to transform the ODE

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

numerical experience suggests it is best to proceed with it directly as it is!
As a first order implicit scheme:

$$(\Phi - \Delta t \cdot \mathcal{L}\Phi) \cdot \boldsymbol{\lambda}(t - \Delta t) \;=\; \Phi \cdot \boldsymbol{\lambda}(t)\,. \tag{5.1}$$

Note that since $\boldsymbol{f}(t) = \Phi \cdot \boldsymbol{\lambda}(t)$, we can always evaluate the right hand side directly from the respectively given terminal boundary conditions, i.e., the (final) pay-off (where applicable):

$$(\Phi - \Delta t \cdot \mathcal{L}\Phi) \cdot \boldsymbol{\lambda}(T - \Delta t) \;=\; \boldsymbol{f}(T)\,. \tag{5.2}$$

This avoids the projection of any pay-off value functions onto the RBF basis!

This may not seem like much of an advantage since this projection, by its nature of being an exact interpolation, ought to return the exact pay-off values at the cluster vertices.

However, since some trades may be long-dated, the consistent valuation of all the legs of a portfolio may involve laying out a calculation for tens of years, with some of the trades in the portfolio being only short dated.

If we fix the cluster-to-state-space scaling matrix Ω to suit the longest trade, the short-dated components can incur valuation over a set of (state space) nodes that are effectively *many* standard deviations away from the money.

It is for this reason that it can be beneficial to use a different $\Omega_q := \Omega(t_q)$ for each time horizon t_q, $q = 0, .., m$ for m time steps from the valuation date to the last observation date.
Since we always use the same (de-scaled) cluster coordinate vertices $\{\boldsymbol{\xi}_i\}$ for $i = 1, .., n$, and since, typically, we choose

$$\Omega_{q-1} < \Omega_q \tag{5.3}$$

in all of its (diagonal) elements. This means that our cluster, in state space coordinates, typically *contracts* as we carry out the backward induction:

$$\boldsymbol{x}_i(t_q) = \Omega_q \cdot \boldsymbol{\xi}_i \qquad \Longrightarrow \qquad |\boldsymbol{x}_i(t_{q-1})| < |\boldsymbol{x}_i(t_q)| \,. \tag{5.4}$$

Note that this has no impact on the elements of the matrix Φ since it is fully evaluated simply on cluster coordinates, but it does mean that $\mathcal{L}\Phi$ varies from one time step to the next due to its dependence on Ω, i.e.,

$$\mathcal{L}\Phi_q \;:=\; \mathcal{L}\Phi(t_q) \,. \tag{5.5}$$

Having rolled back to some time horizon t_q, i.e., having computed

$$\boldsymbol{\lambda}_q \;:=\; \boldsymbol{\lambda}(t_q)\;, \tag{5.6}$$

and taking into account *additional payoff contributions happening at t_q denoted as $\pi_q(\boldsymbol{x}, t_q)$*, the next standard fully implicit step backwards to t_{q-1} is then

$$(\Phi - \Delta t \cdot \mathcal{L}\Phi_{q-1}) \cdot \boldsymbol{\lambda}_{q-1} \;=\; \Phi_{\langle q-1,1\rangle} \cdot \boldsymbol{\lambda}_q + \boldsymbol{\pi}_q \tag{5.7}$$

with the matrix $\Phi_{\langle q-1,1\rangle}$ defined by

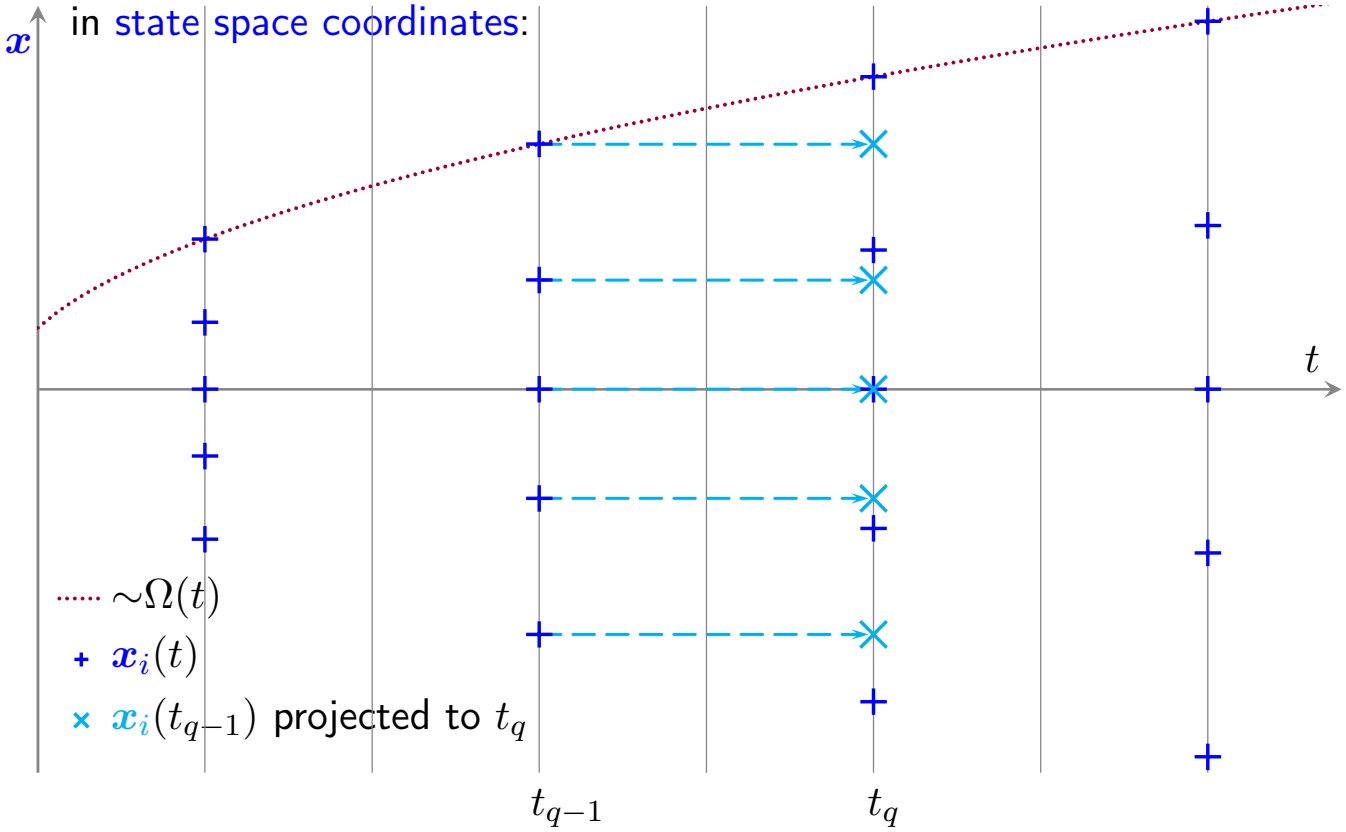
$$\begin{aligned} \left(\Phi_{\langle q-1,1\rangle}\right)_{ij} &= \phi(|(\boldsymbol{x}_i(t_{q-1}) - \boldsymbol{x}_j(t_q))/\Omega_q|; \epsilon_j) \\ &= \phi(|\tfrac{\Omega_{q-1}}{\Omega_q} \cdot \boldsymbol{\xi}_i - \boldsymbol{\xi}_j|; \epsilon_j) \end{aligned} \tag{5.8}$$

and the elements of the vector $\boldsymbol{\pi}_q$ given by

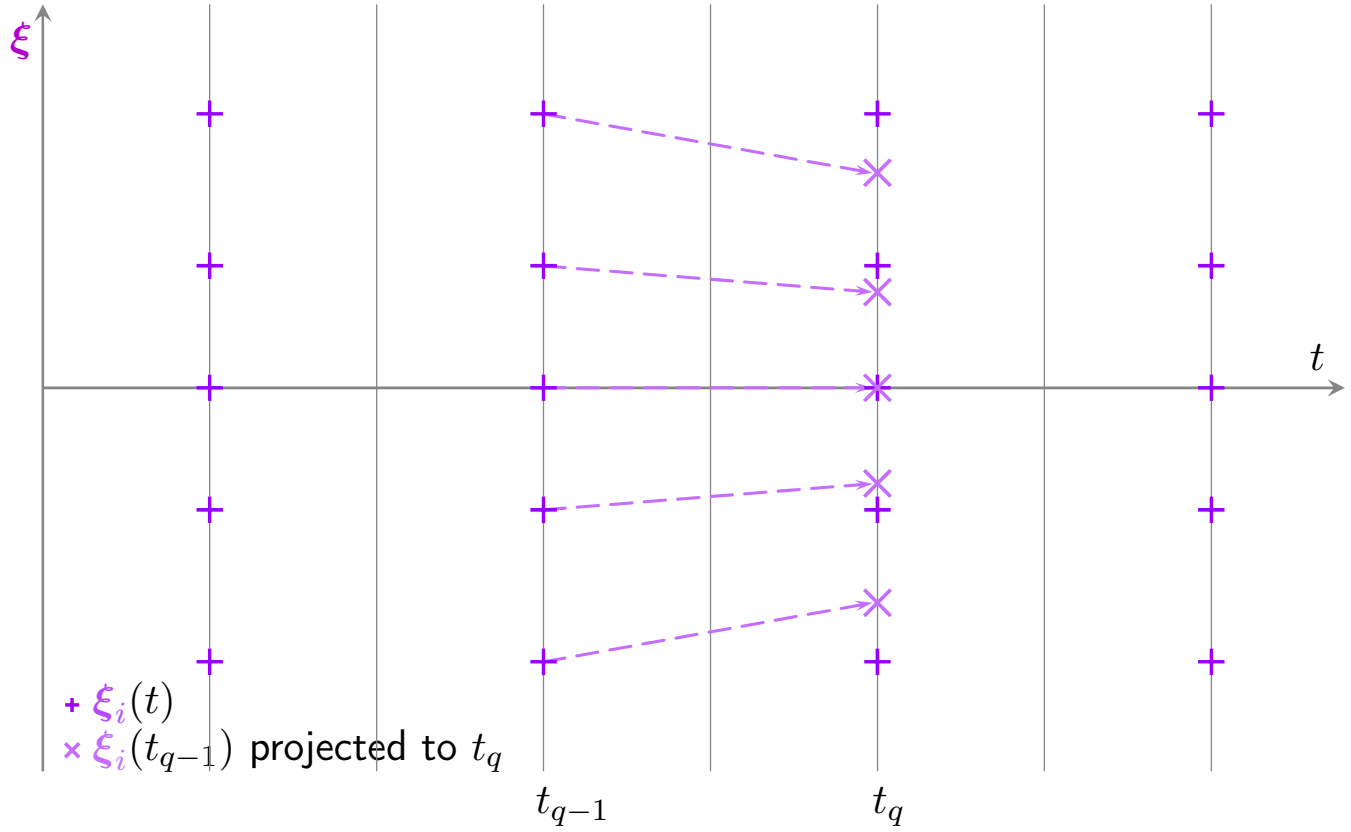
$$(\boldsymbol{\pi}_q)_i \;=\; \pi_q(\boldsymbol{x}_i(t_{q-1}), t_q) \;=\; \pi_q(\Omega_{q-1} \cdot \boldsymbol{\xi}_i, t_q) \tag{5.9}$$

Note that we avoid the projection of the extra payment π_q onto the basis!

To illustrate the basis function evaluation logic in (5.8)



And, for comparison, in cluster coordinates:



To analyze, say, the stability of the first order implicit scheme

$$(\Phi - \Delta t \cdot \mathcal{L}\Phi_{q-1}) \cdot \boldsymbol{\lambda}_{q-1} \;=\; \Phi \cdot \boldsymbol{\lambda}_q \tag{5.10}$$

we investigate the spectrum of

$$(1 - \Delta t \cdot \tilde{G})^{-1} \tag{5.11}$$

where

$$\tilde{G}_{q-1} \;:=\; \Phi^{-1} \cdot \mathcal{L}\Phi_{q-1} \tag{5.12}$$

which is the transpose of G_{q-1} when all ϵ_j are the same (and thus $\Phi = \Phi^\top$).
In comparison, to analyze, say, the stability of the first order explicit scheme

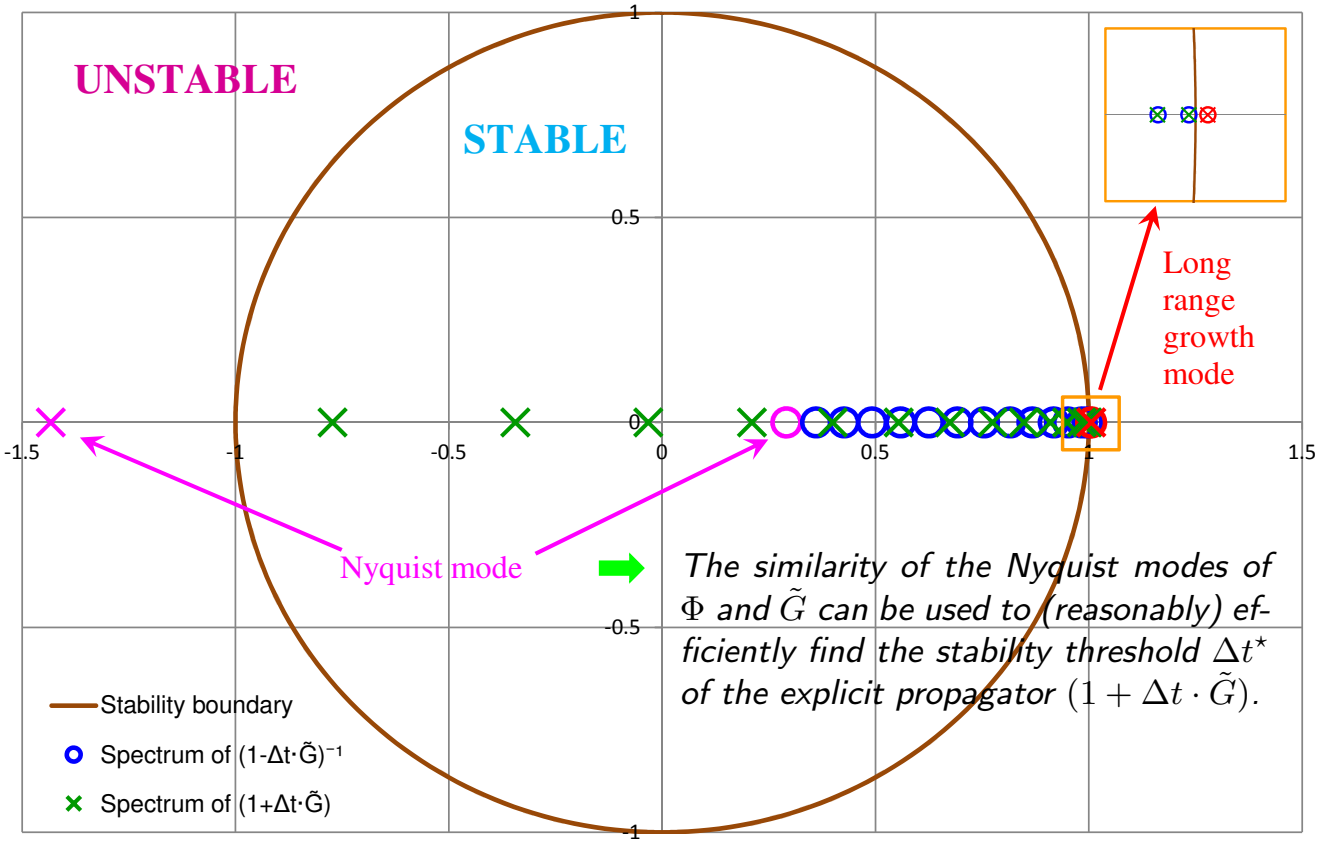
$$\Phi \cdot \boldsymbol{\lambda}_{q-1} \;=\; (\Phi + \Delta t \cdot \mathcal{L}\Phi_q) \cdot \boldsymbol{\lambda}_q \;. \tag{5.13}$$

we need to see the spectrum of

$$(1 + \Delta t \cdot \tilde{G}_q) \;. \tag{5.14}$$

Note that both schemes require the solution of a dense linear system!

For our running multiquadric example, with $\Delta t = 0.25$:



The similarity of the Nyquist modes of Φ and $\tilde{G}$ can be used to (reasonably) efficiently find the stability threshold $\Delta t^\star$ of the explicit propagator $(1 + \Delta t \cdot \tilde{G})$.

Given that the (first order) *explicit* and *implicit* schemes *both* require the solution of a dense linear system, the advantage of the explicit scheme reduces to the fact that we can pre-factorise Φ *once* for all time steps.

However, caution should be applied for the first step backward from any payment date, i.e., from any point in time that adds a time-inhomogeneous source term.

Since financial contingent payment formulæ often contain non-differentiable terms, attempting a fully explicit step straight out of a payment date amounts to a projection of the payoff onto the radial basis, and that rarely works well!

All in all, implicit schemes appear to be generally the better choice.

Since the Crank-Nicolson scheme contains an explicit component, we equally (generally) don't favour it for higher order convergence in Δt.

Instead, the following schemes have been found to be of merit:-

- **BDF2**. This is a second-order fully implicit *stiff* scheme [CH52, Gea71] originally for the forward ODE

$$\dot{y} \;=\; f(y) \tag{5.15}$$

given by

$$y(t_{n+2}) \;=\; \tfrac{4}{3}y(t_{n+1}) - \frac{1}{3}y(t_n) + \tfrac{2}{3}\cdot\tau\cdot f\left(y(t_{n+2})\right) \tag{5.16}$$

with time steps of length τ. In this context here, this becomes

$$(\Phi - \tfrac{2}{3}\Delta t\cdot\mathcal{L}\Phi_q)\cdot\boldsymbol{\lambda}_q \;=\; \tfrac{4}{3}\cdot\Phi\cdot\boldsymbol{\lambda}_{q+1} - \tfrac{1}{3}\cdot\underbrace{\Phi\cdot\boldsymbol{\lambda}_{q+2}}_{\boldsymbol{f}(t_{q+2})}\,. \tag{5.17}$$

This scheme requires a start-up.

When started with Crank-Nicolson, it is also referred to as **TR-BDF2**.

- **Padé(0,2)** is based on the (0,2)-Padé expansion $\mathrm{e}^t \approx \dfrac{1}{1-t+t^2/2}$.

It is a second-order fully implicit scheme, sometimes (though rarely) also referred to as *second order backwards Euler* or simply as

second order implicit.

In this context here, we have

$$(\Phi - \Delta t\cdot\mathcal{L}\Phi_q + \tfrac{1}{2}\Delta t^2\cdot\mathcal{L}^2\Phi_q)\cdot\boldsymbol{\lambda}_q \;=\; \underbrace{\Phi\cdot\boldsymbol{\lambda}_{q+1}}_{\boldsymbol{f}(t_{q+1})}\,. \tag{5.18}$$

NOTE that $\mathcal{L}^2\Phi$ is the matrix given by the analytical evaluation of $\mathcal{L}^2\cdot\phi$, i.e., its elements are

$$\left(\mathcal{L}^2\Phi\right)_{ij} \;=\; \mathcal{L}^2\cdot\phi(|\boldsymbol{x}-\boldsymbol{x}_j|)\big|_{\boldsymbol{x}=\boldsymbol{x}_i}\,. \tag{5.19}$$

The matrix $\mathcal{L}^2\Phi$ is ***not the same*** as $(\mathcal{L}\Phi)^2$!

Details for the calculation of $\mathcal{L}^2\Phi$ were given in equation (4.19) on page 39.

The second order implicit method also serves well to start-up the BDF2 scheme.

Some results

We look at a 1Y ATM vanilla FX option with 20% volatility.

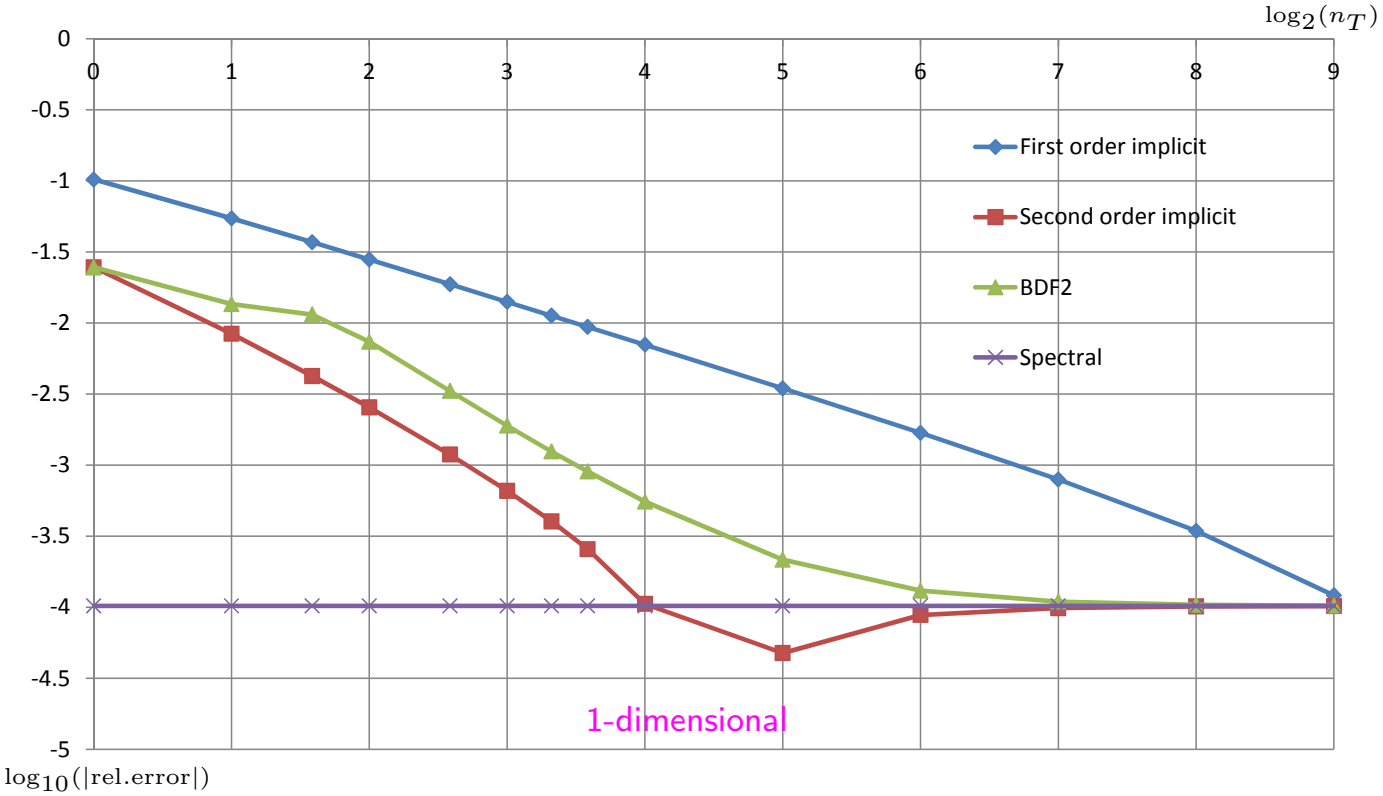
We show its price convergence to the exact value as a function of:-

- the number of time steps n_T for fixed n_{cluster}, and
- the number of cluster nodes n_{cluster} fixed n_T for,

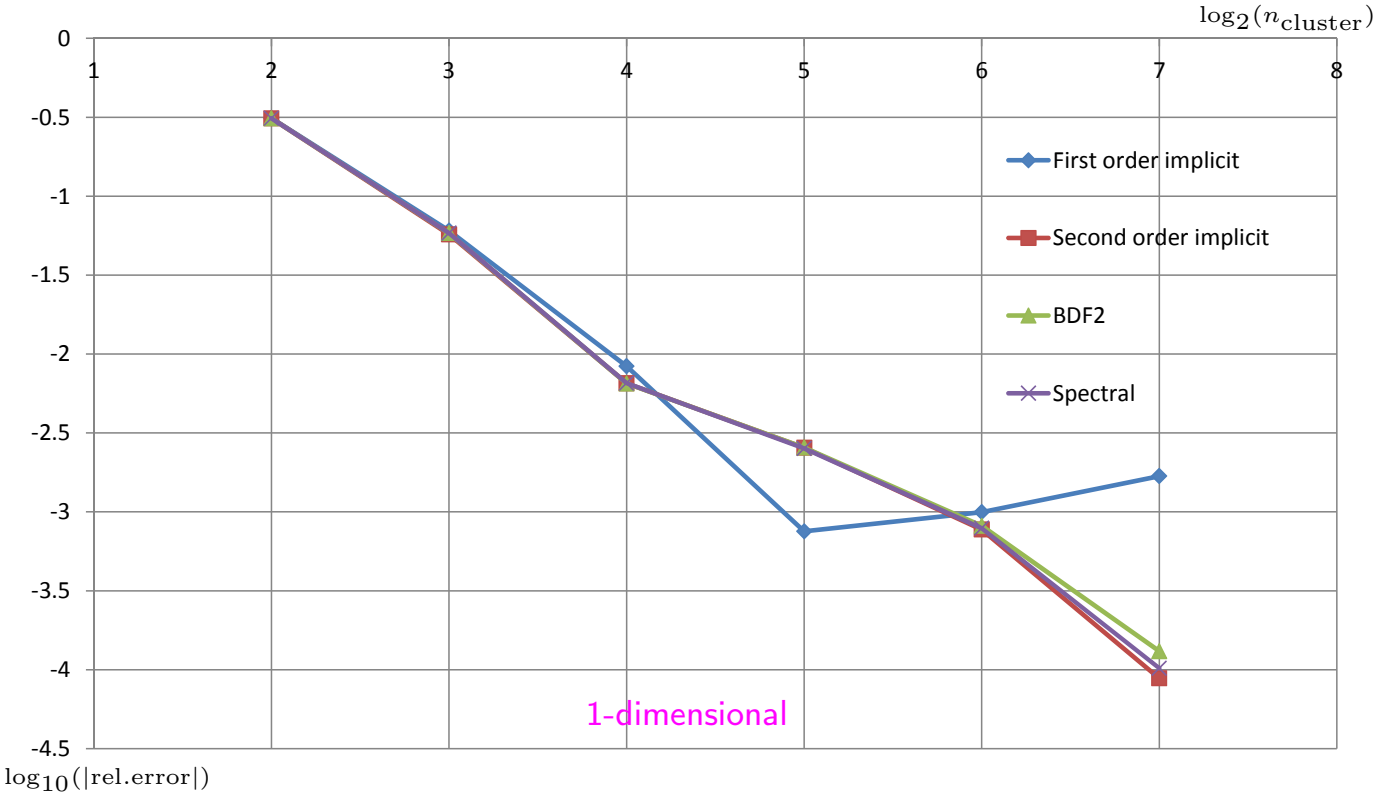
priced with Black-Scholes-Merton model and a cross-currency Hull-White (aka 'Linear Gaussian Markov') model with normal rates volatilites of 200bps and mean reversion speeds of 5% for USD and 10% else.

The cluster nodes are all from a Sobol' sequence mapped to normal variates.

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

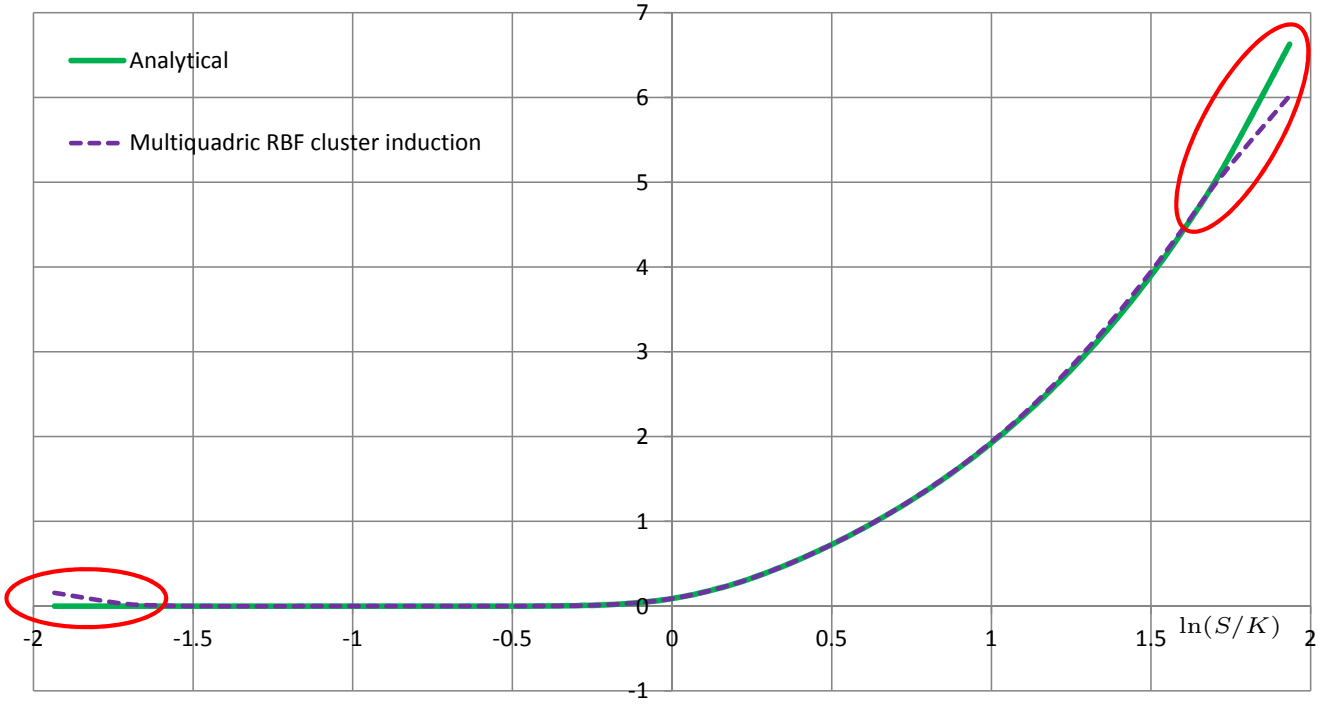


Vanilla 1Y ATM FX option with Black-Scholes and $n_T = 2^6 = 64$.

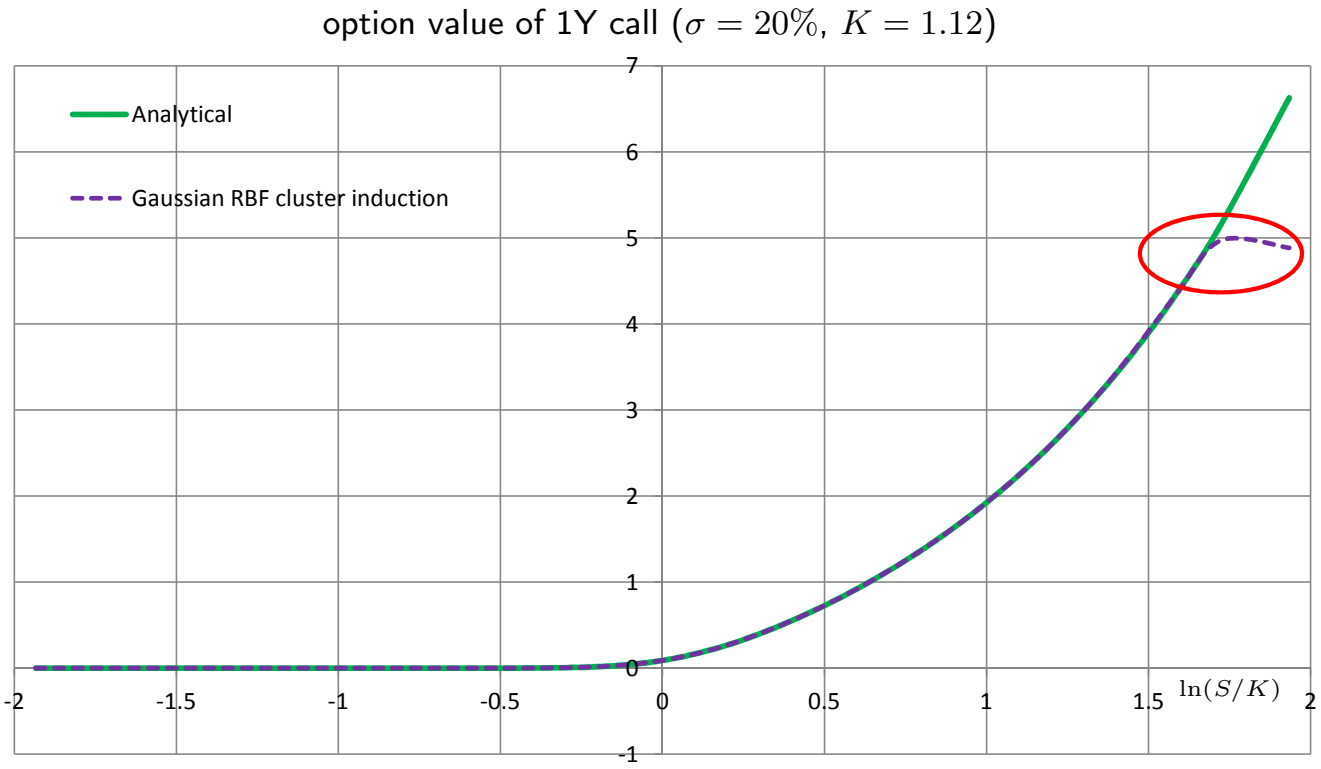


The detachment effect of the long range growth mode of multiquadrics:

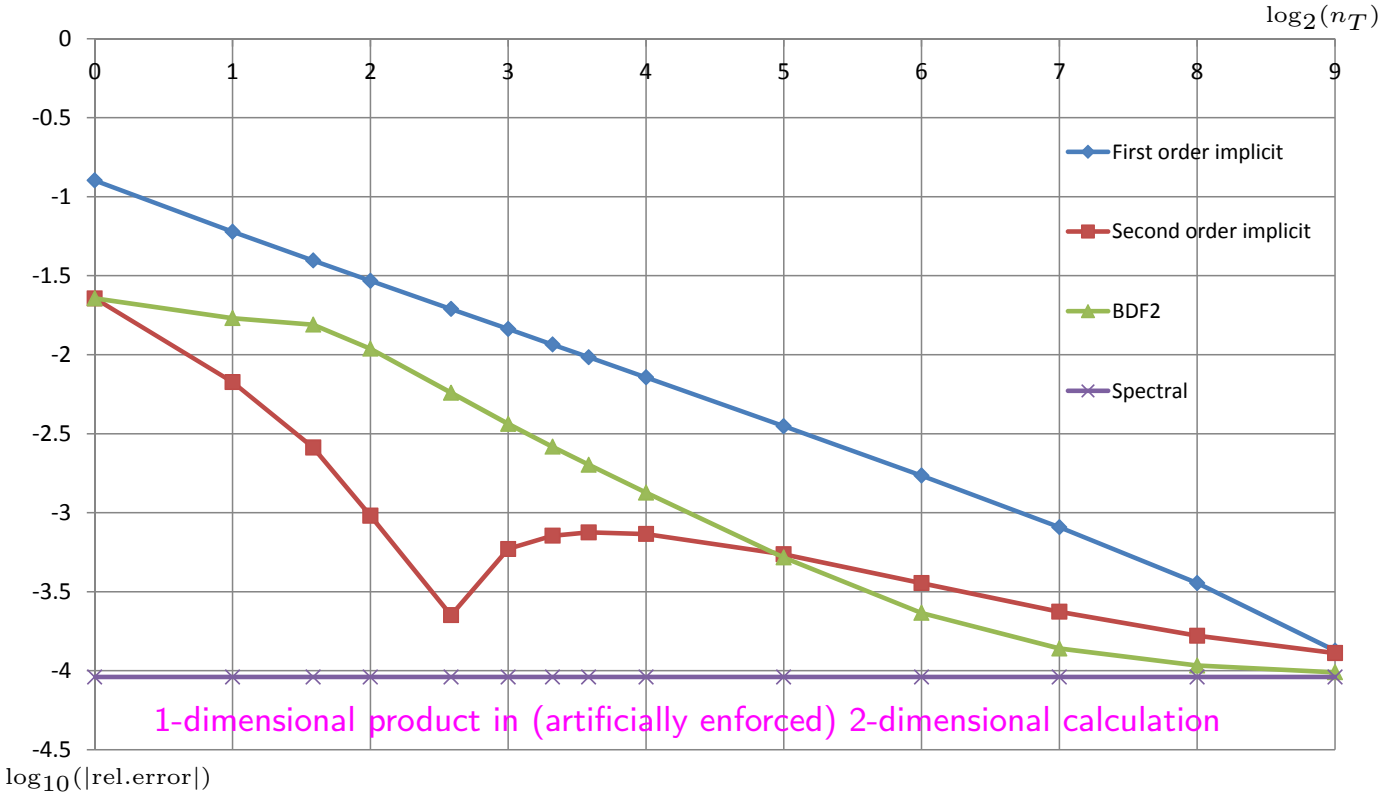
option value of 1Y call ($\sigma = 20\%$, $K = 1.12$)



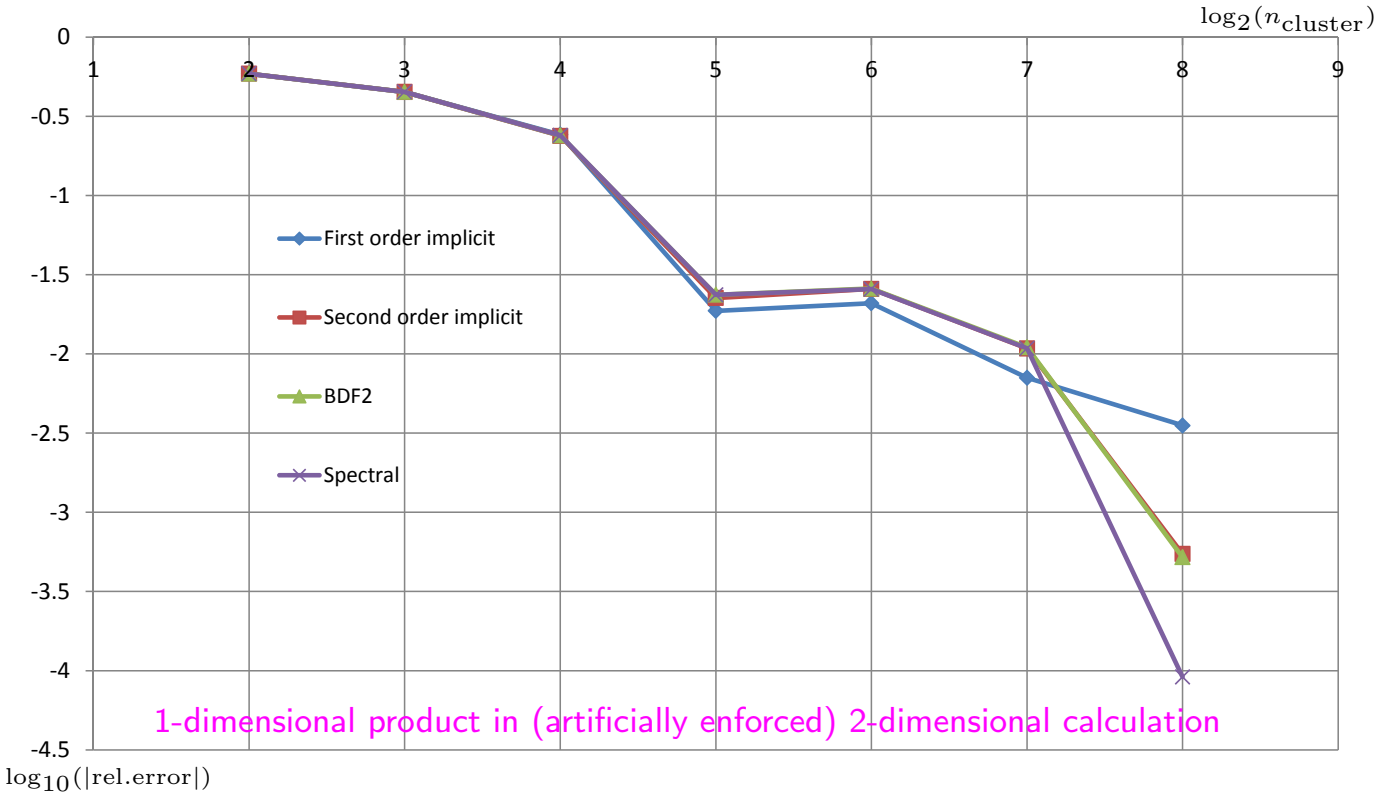
The decay effect of Gaussian RBFs (1Y ATM call):



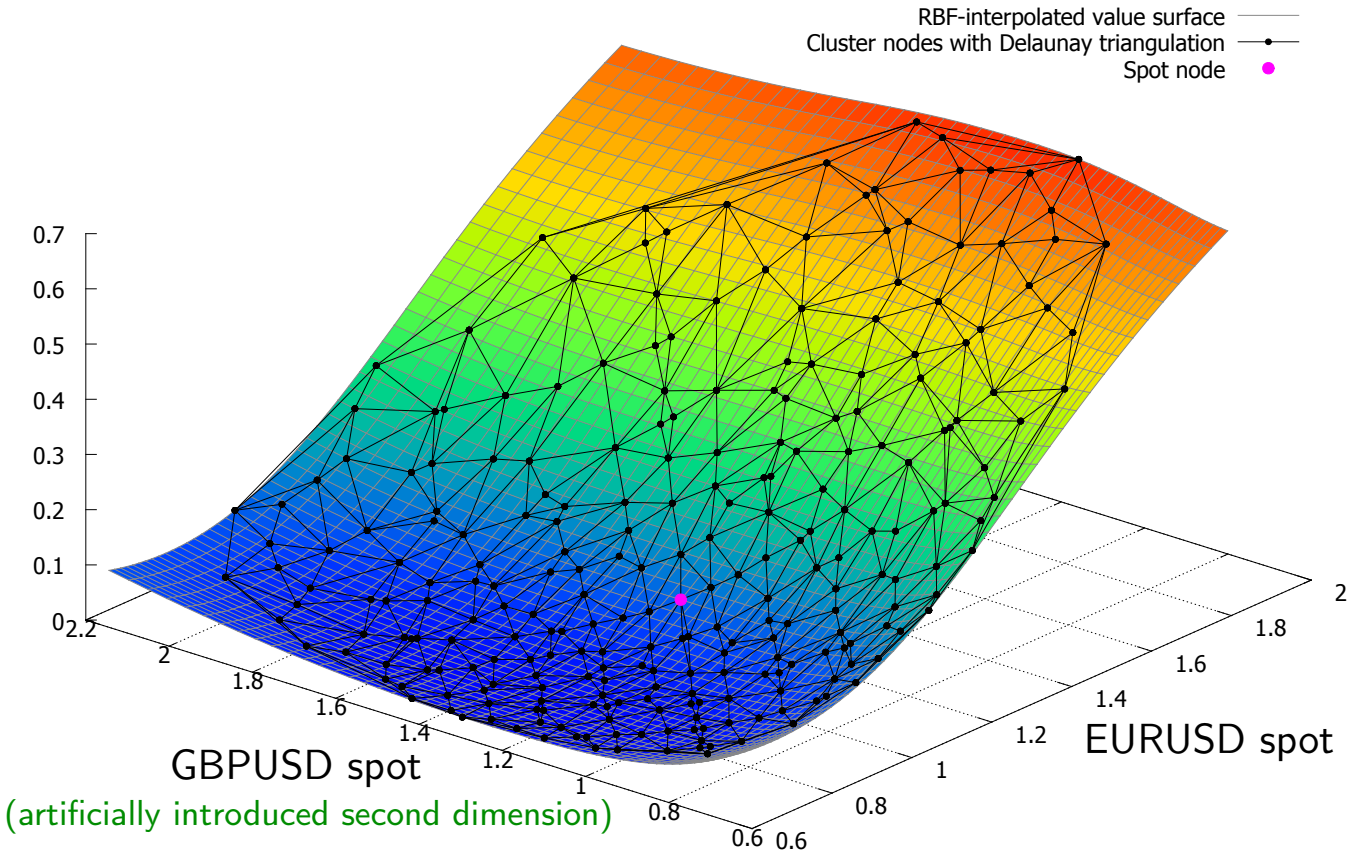
Vanilla 1Y ATM FX option with Black-Scholes and $n_{\text{cluster}} = 2^8 - 1 = 255$.



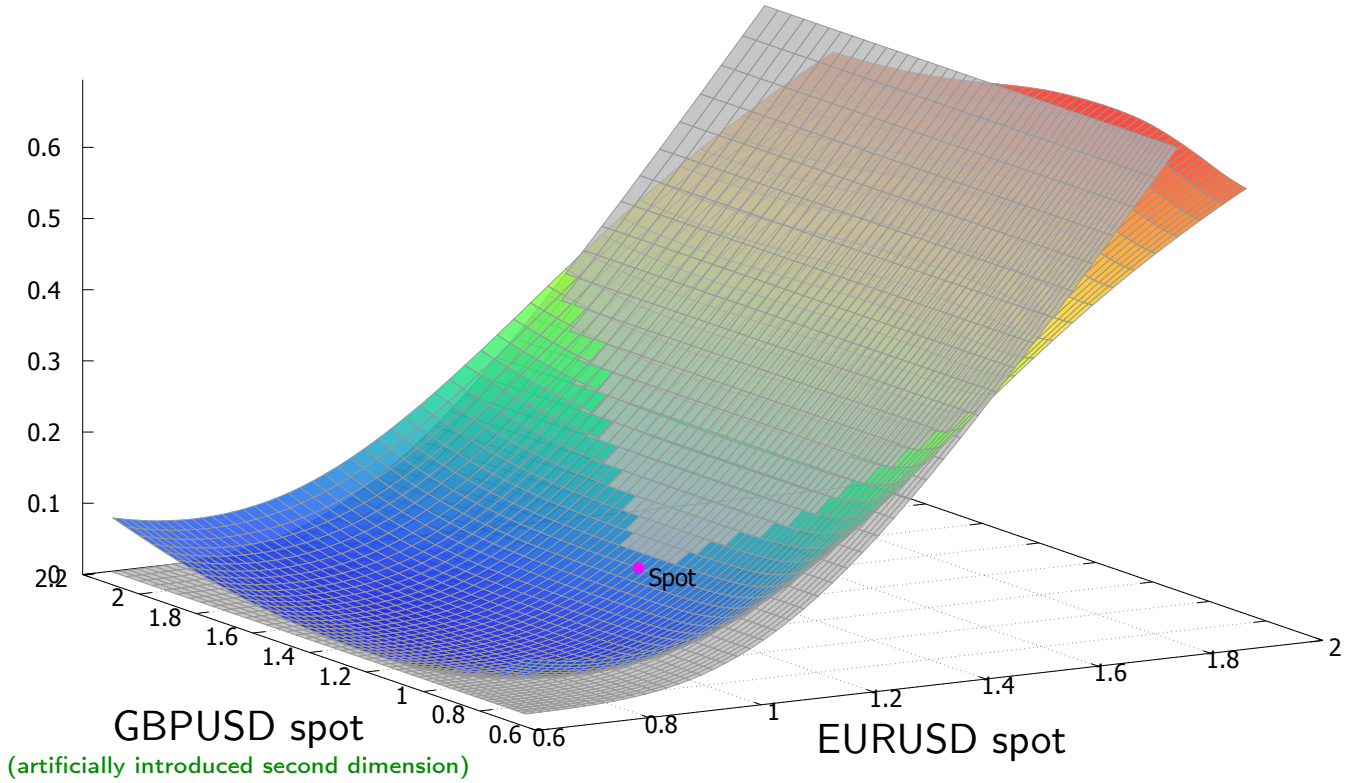
Vanilla 1Y ATM FX option with Black-Scholes and $n_T = 2^5 = 32$.



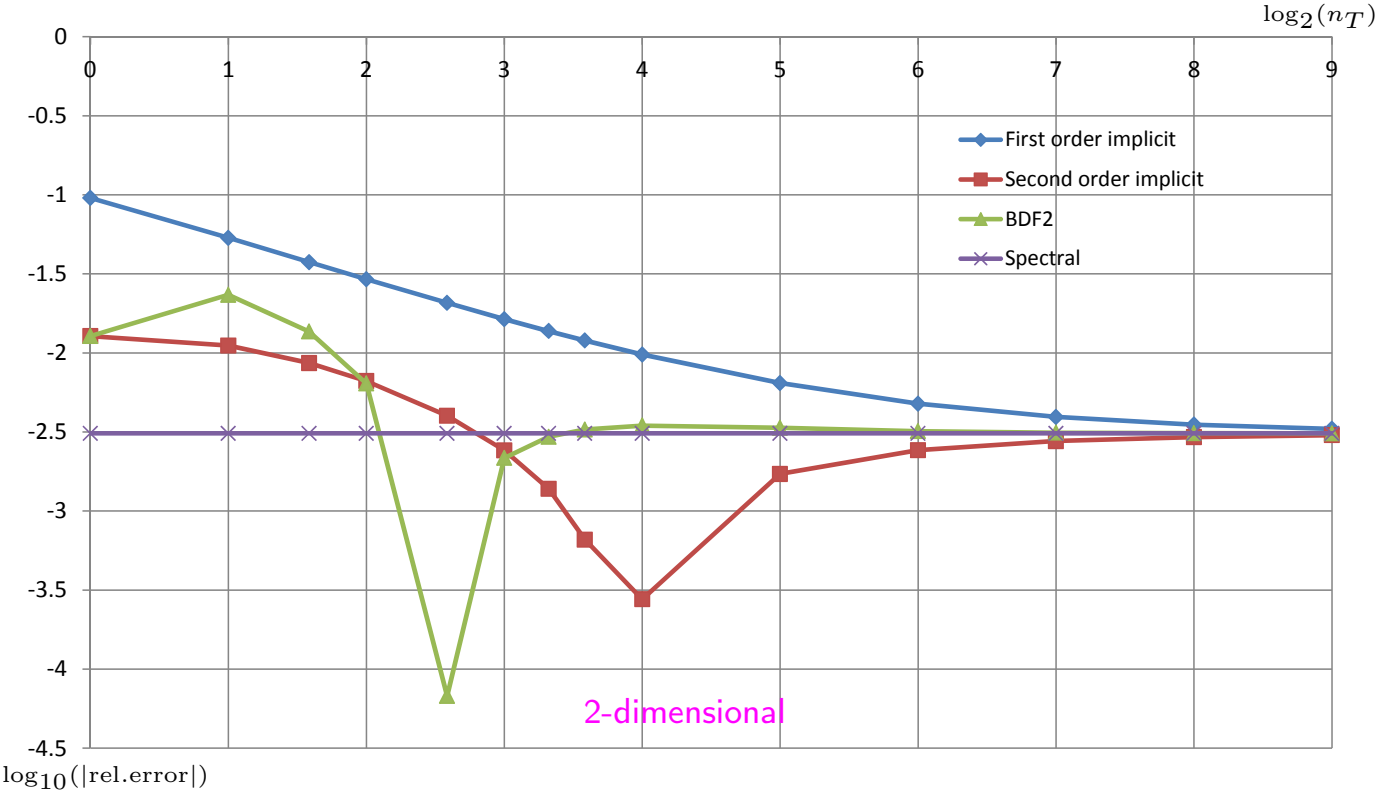
EURUSD call option with Black-Scholes ($S = K = 1.12$).



Comparison with exact analytical values (in gray):



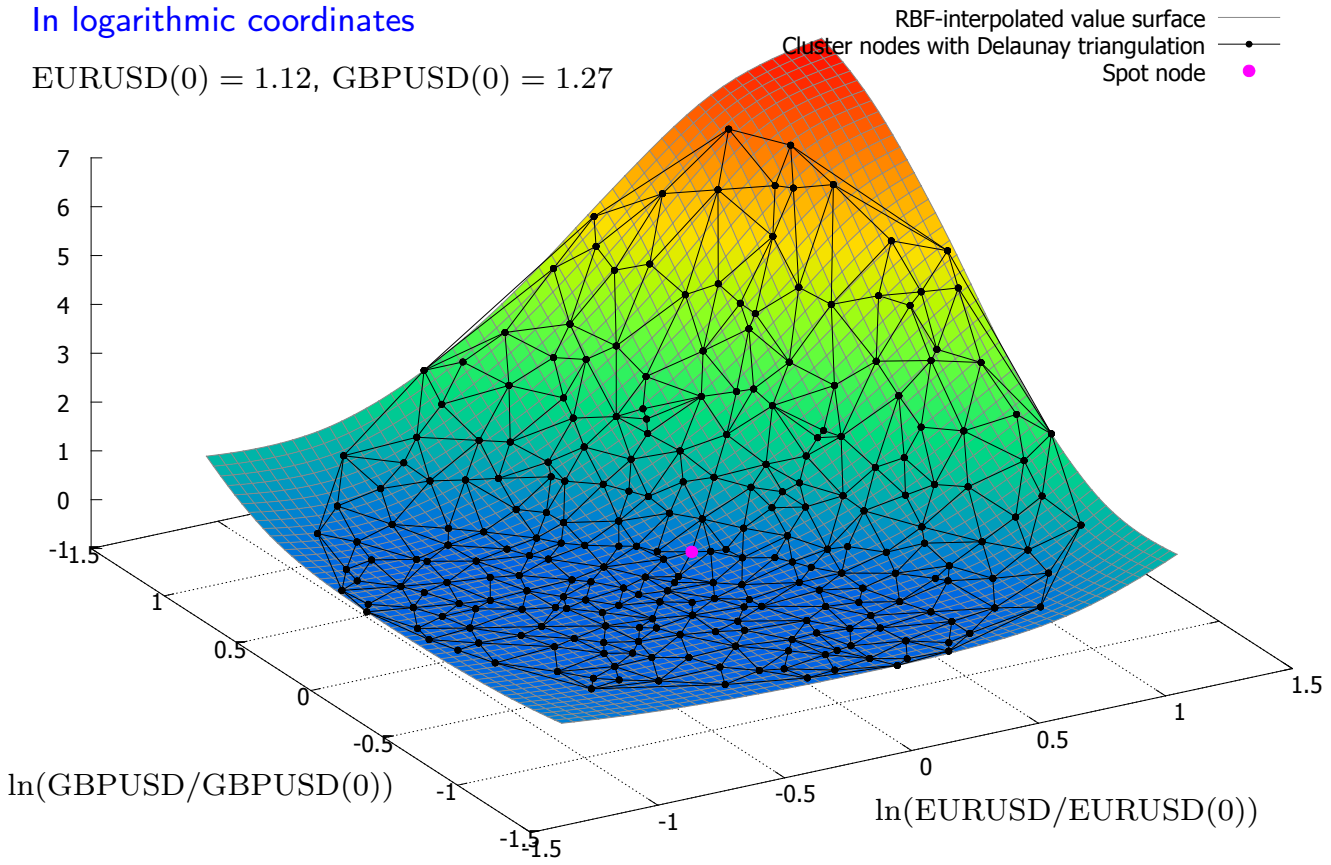
1Y ATM composite option with Black-Scholes and $n_{\text{cluster}} = 2^8 - 1 = 255$.



Composite option $\mathrm{E}[(\mathrm{EURUSD} \cdot \mathrm{GBPUSD} - 1.4224)_+]$ with Black-Scholes

In logarithmic coordinates

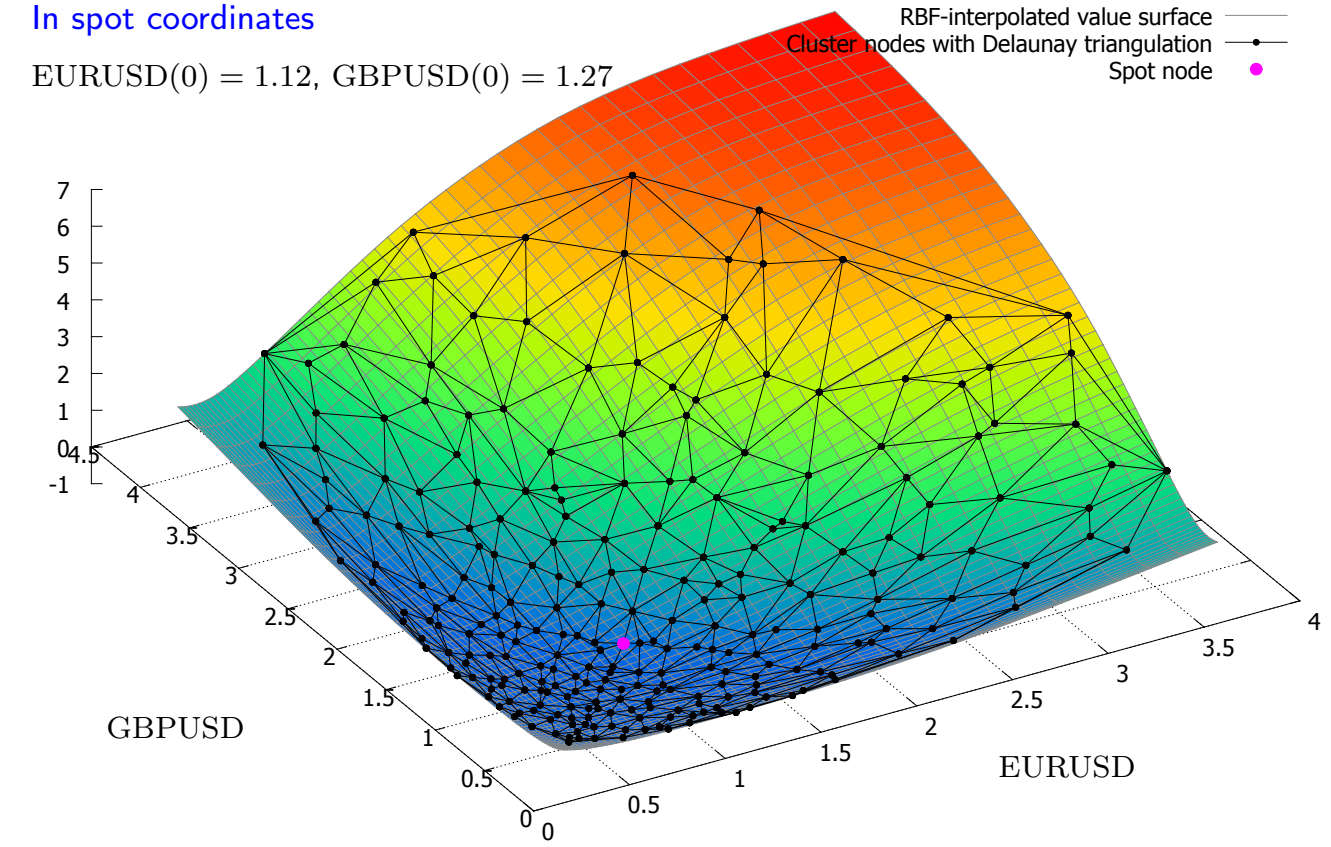
$\mathrm{EURUSD}(0) = 1.12,\ \mathrm{GBPUSD}(0) = 1.27$



Composite option $\mathrm{E}[(\mathrm{EURUSD} \cdot \mathrm{GBPUSD} - 1.4224)_+]$ with Black-Scholes

In spot coordinates

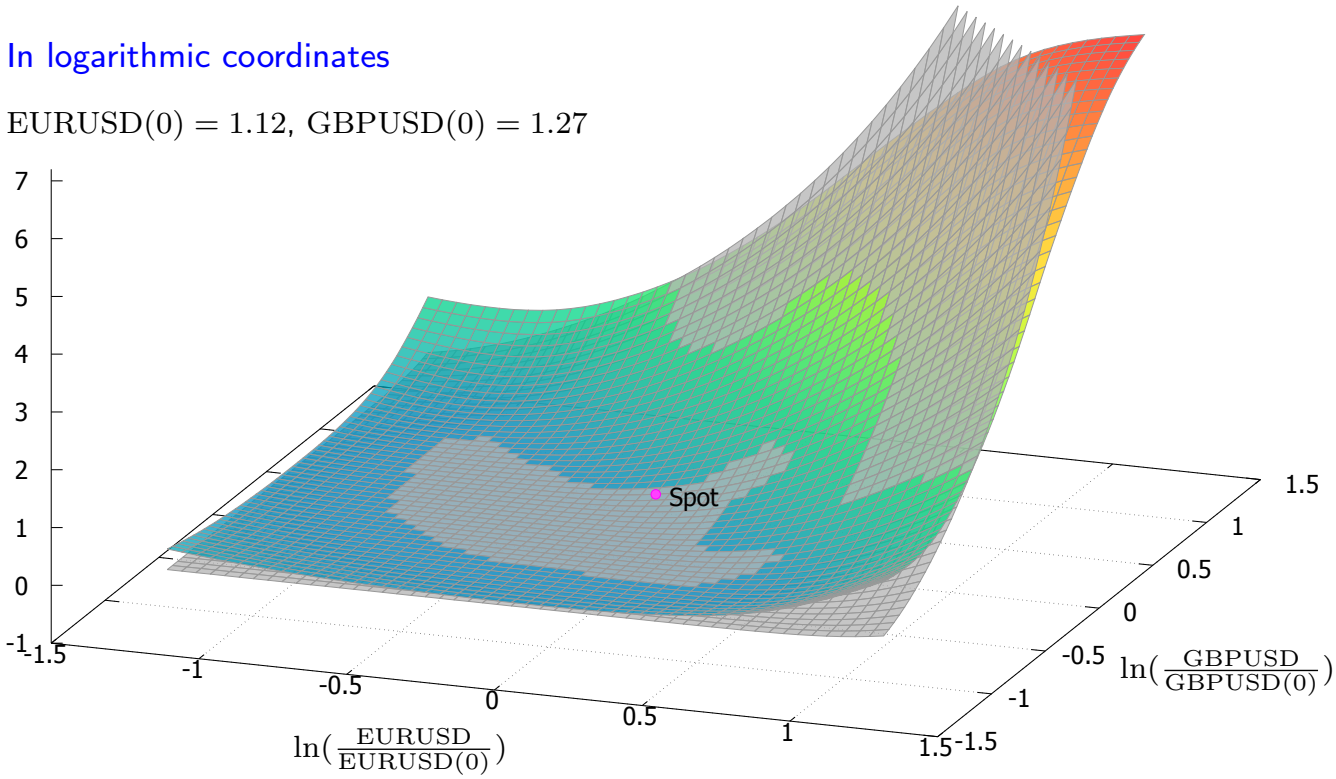
$\mathrm{EURUSD}(0) = 1.12,\ \mathrm{GBPUSD}(0) = 1.27$



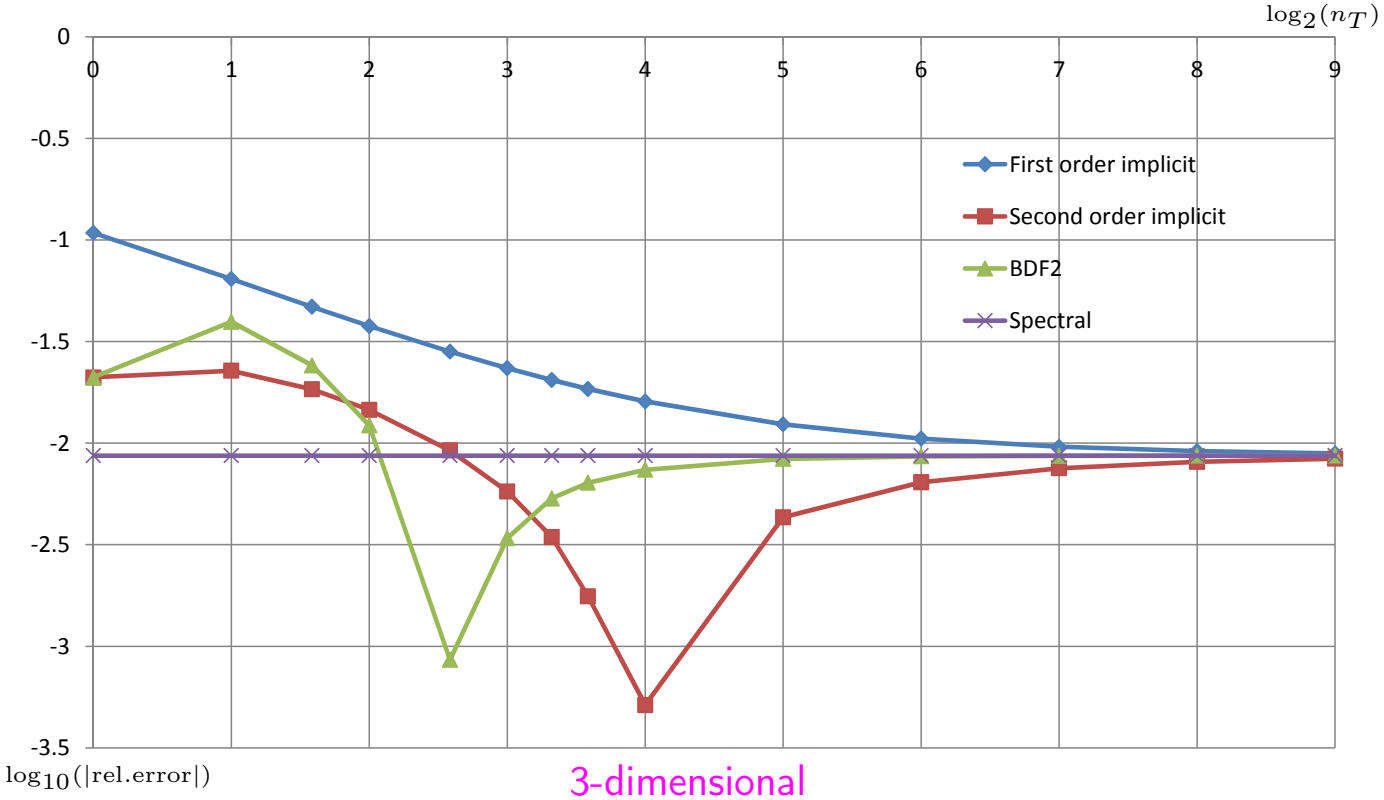
Comparison with exact analytical values (in gray):

In logarithmic coordinates

$\mathrm{EURUSD}(0) = 1.12$, $\mathrm{GBPUSD}(0) = 1.27$

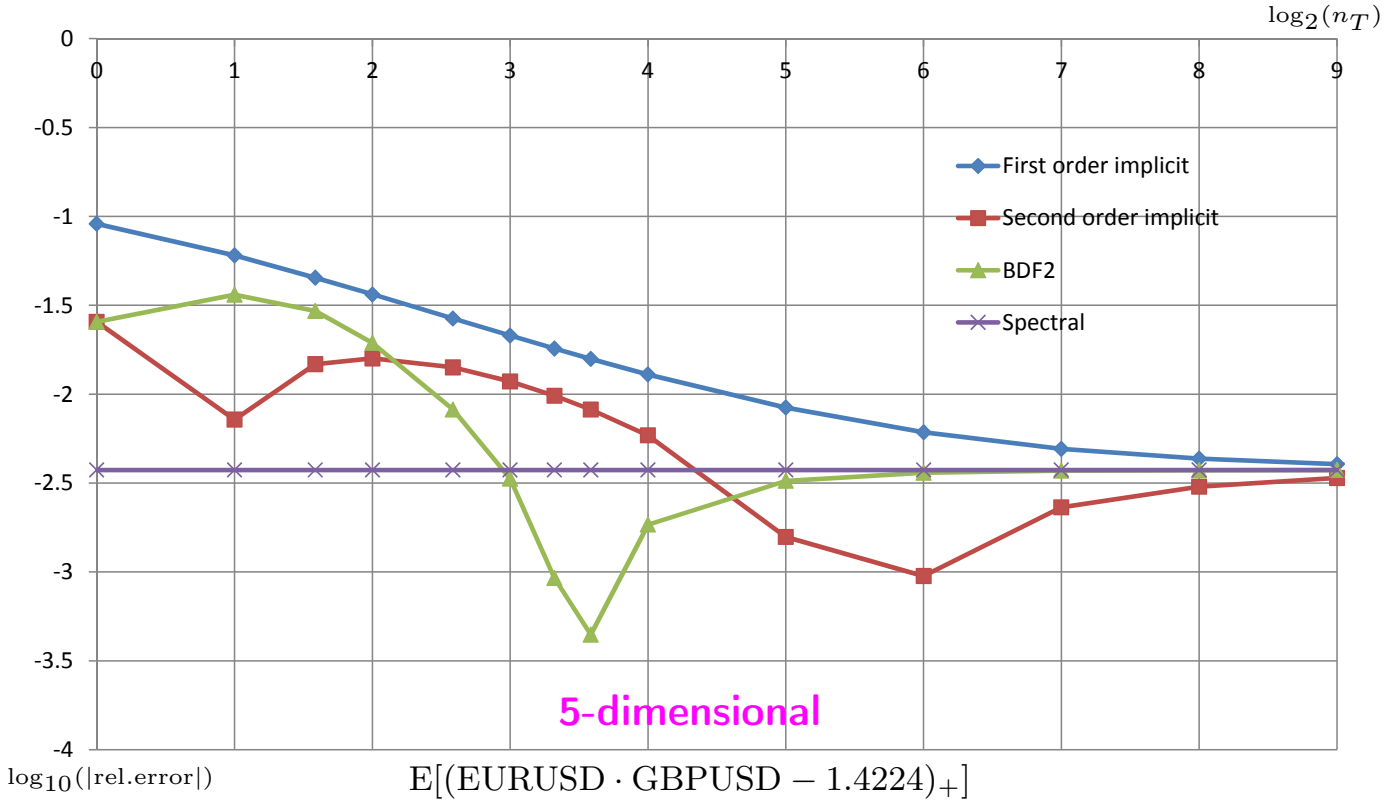


1Y ATM FX option with Linear Gaussian Markov model and $n_{\mathrm{cluster}} = 255$.

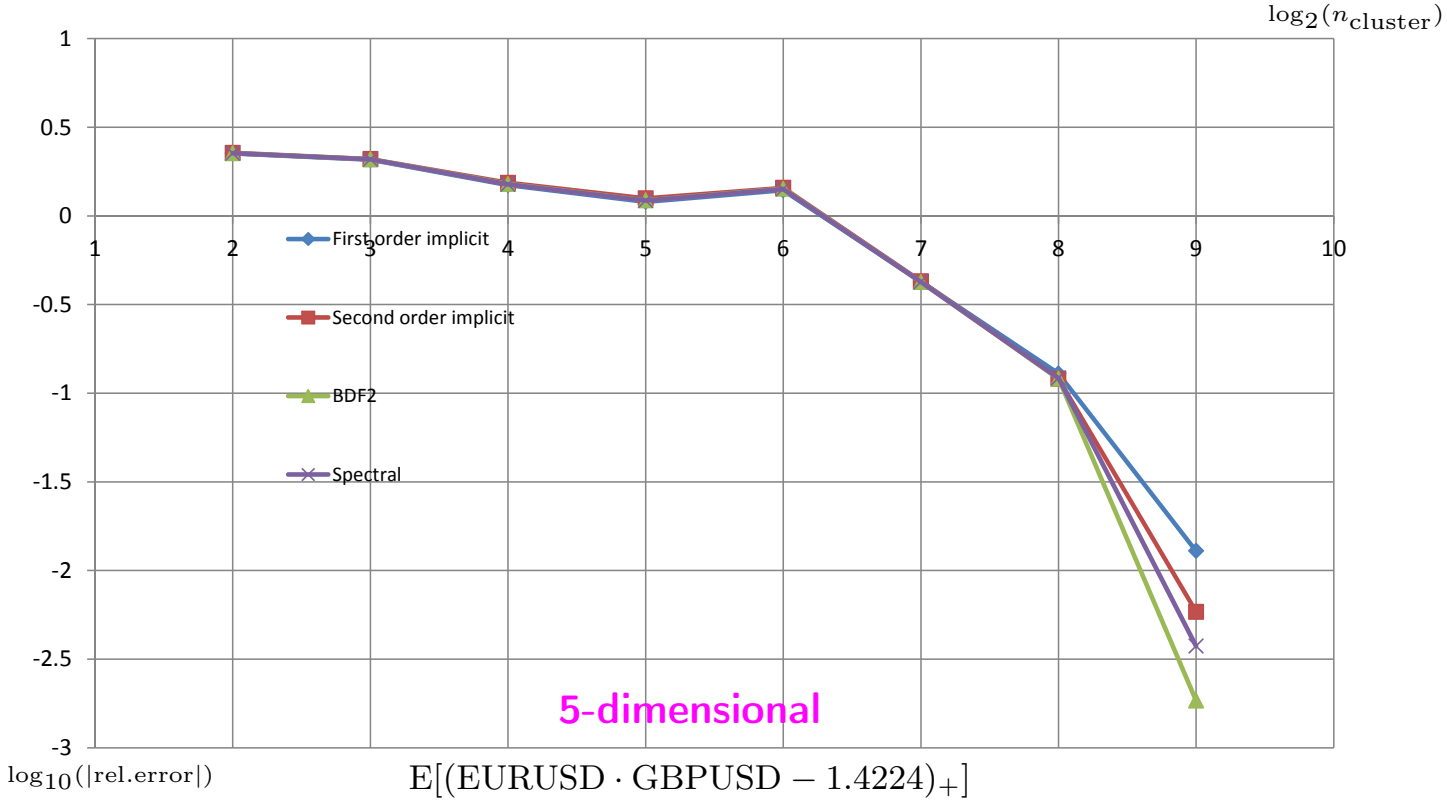


3-dimensional

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



1Y composite option with Linear Gaussian Markov model and $n_T = 16$.



Implementation and optimisations

Like with Libor market models, the *real* work starts with the implementation:-

- How do we choose the shape parameters ϵ_j?

  There is some research on this subject of variable practical applicability.

- How do we generate the cluster?

  There is little in the literature.

- Generic optimisations, e.g.:

  - Pre-factorise matrices where possible and multiple (non-simultaneous) solutions with different right-hand-sides are required.
  - Invoke Lapack's "solve-with-SVD" routine instead of "give-me-the-full-SVD" when you don't actually need the SVD itself.

- When ϵ is too large, the approximation diverges: the RBFs are then too localised to genuinely *interpolate* between cluster nodes.

- When ϵ is too small, the linear system starts to suffer from subtractive cancellation and becomes numerically unsolvable. The system's matrix condition number diverges.

- The sweet spot tends to be when the reciprocal matrix condition number *approaches* machine precision.

On the influence of the shape parameter on the accuracy, e.g., see:-

- figures 2 and 3 in [CS09].
- figures 11 and 13 in [Mil14].
- figures 2.3 and 7.4 in [FZ07].
- figures 5.1, 5.2, 5.3, 7.1, 7.2, and 7.3 in [FM12].
- figures 22, 26, and 27 in [Fas11].
- figure 3.1 b) and 4.2 in [FF15b].
- figure 2 in [SL16].
- figure 5.2 in [Wan14].
- figures 2, 4 and 6 in [LF03].
- figures 4–8, 10–12, and 14–17 in [Mon11] (my favourite).

Figures 12 and 14 from [Mon11]:

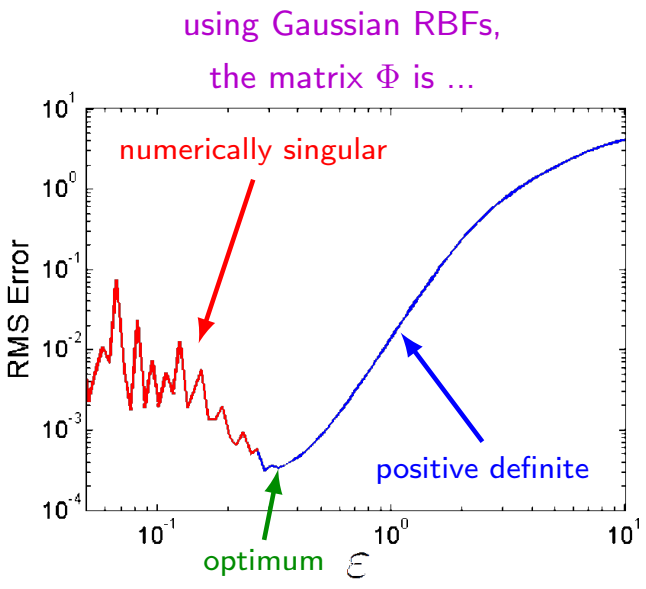


Figure 12 [Mon11].

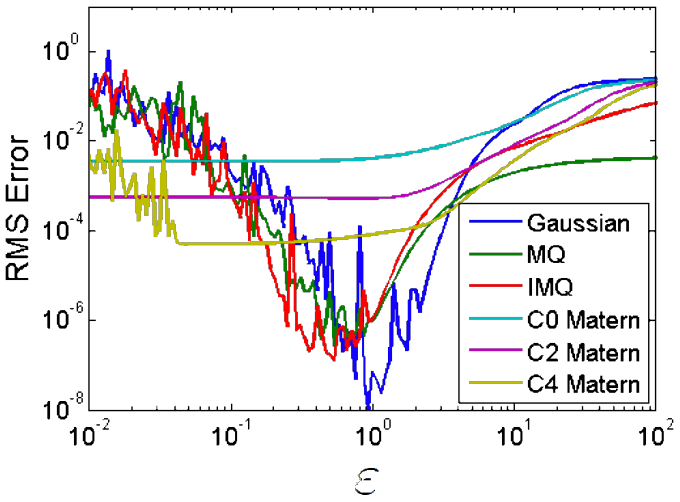


Figure 14 [Mon11].

The optimum shape parameter ϵ tends to be near the point where the kernel matrix Φ becomes singular, i.e., where $\kappa(\Phi)^{-1} \gtrsim$ DBL_EPSILON.

- Various methods have been suggested for (semi-)automatic choice algorithms for ϵ such as 'Leave-one-out-cross-validation' (LOOCV), but, for our industrial application purposes, none of them appear to be anywhere near robust enough for commercial use.
- I found that choosing each ϵ_j, to some extent, with a global cap and floor, based on the reciprocal distance to its near(est) neighbour(s) gives a significant improvement.
- Interestingly, there is disappointingly little discussion of this observation of mine in the literature, (arguably) the sole noteworthy exception being "*The Runge phenomenon and spatially variable shape parameters in RBF interpolation*" [FZ07]. Quote:

  > *The conditioning of the A-matrix is likely to be greatly improved when using spatially variable* ϵ_k.

- The conditioning of the A-matrix is not the only benefit, but I am glad I found *at least some literature* in support of my findings.

An unexplored idea:-

- Instead of merely demanding that the equation

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

  holds in all of the radial basis function centres, to arrive at the ODE

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

  with square matrix Φ,

- we ***oversample*** equation (4.5), i.e., ask for it to hold (as best as possible) *in an additional set of* (somehow, arbitrarily) *chosen locations*.

- Applying this to whichever chosen specific time-integration scheme, this results in an overdetermined system of the form

$$A(\epsilon) \cdot \boldsymbol{\lambda} = \boldsymbol{r} \ . \tag{6.1}$$

- Next, we demand that the RMS error of this overdetermined system is minimized over both $\boldsymbol{\lambda}$ *and* ϵ, which gives us the non-linear system

$$\begin{aligned} A^\top \cdot A \cdot \boldsymbol{\lambda} - A^\top \cdot \boldsymbol{r} &= 0 \\ \boldsymbol{\lambda}^\top \cdot A^\top \cdot A' \cdot \boldsymbol{\lambda} - \boldsymbol{r}^\top \cdot A' \cdot \boldsymbol{\lambda} &= 0 \end{aligned} \tag{6.2}$$

  where

$$A' = \frac{\mathrm{d}A(\epsilon)}{\mathrm{d}\epsilon} \ .$$

- Solving this non-linear system is obviously some numerical effort.
- And it may require safeguarding against its Jacobian becoming singular!
- One may not want to do this for each and every time step.
- One could attempt to solve it at every payment date, to capture the contribution of each payoff,
- and freeze ϵ over subsequent (regular) time steps where we'd only use the regular, i.e., square, matrix Φ.

To be tested...

How do we generate the cluster?

- Sadly, most research is done in just 1 dimension ! (!!!)
- 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.

A pragmatic approach:

- 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 standardised ξ-coordinate cluster nodes.
- If you wish, combine with further cluster improvement techniques...

In analogy to conventional (explicit) trees and (multivariate) quadrature methods [BW04, Jäc05], we may wish to avoid having any nodes at ξ-coordinates that in a one-dimensional standard normal distribution we would describe as lying in the tail beyond a certain number of standard deviations, say n standard deviations.

Since a one-dimensional normal distribution has two tails, this gives us that we demand

$$|\xi| \leq r^* \tag{6.3}$$

with

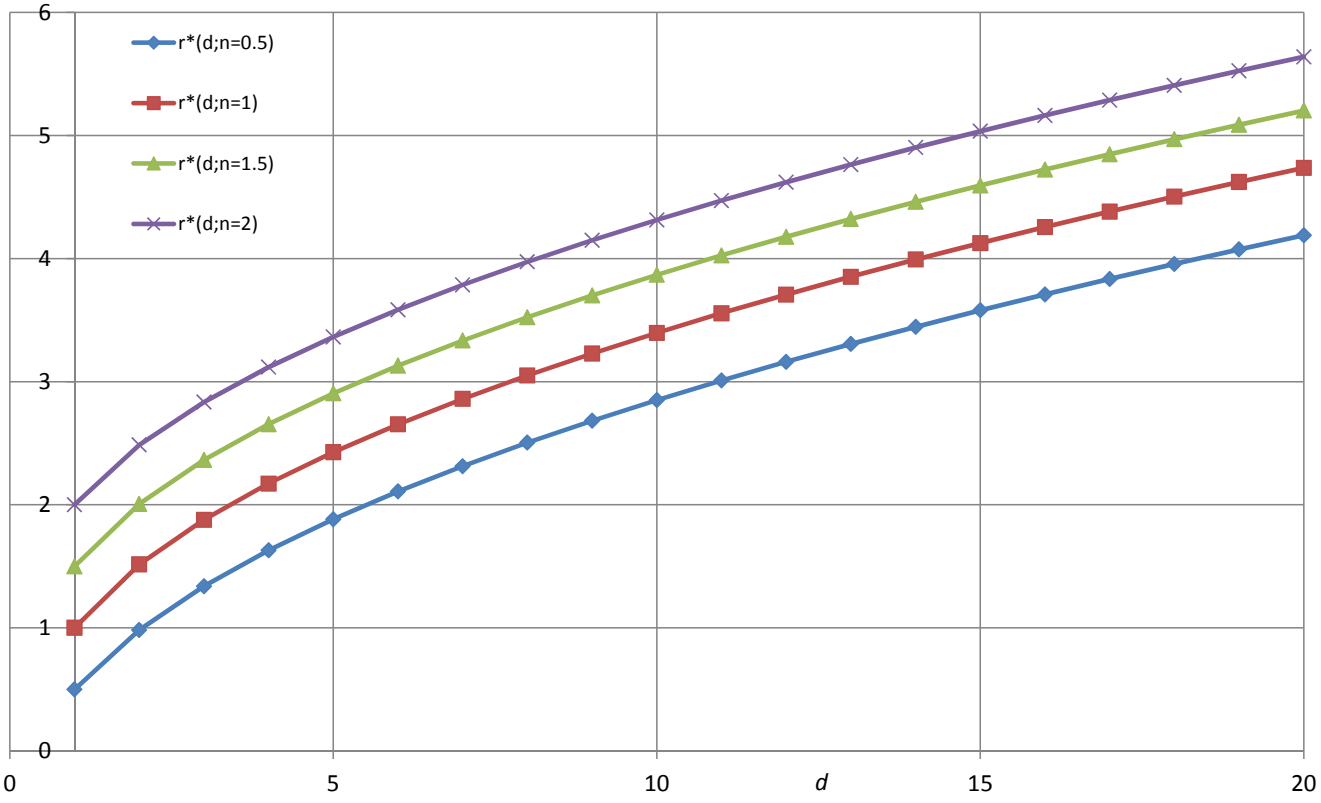
$$1 - \chi^2(r^{*\,2}; d) = 2 \cdot \Phi(-n) \tag{6.4}$$

where $\Phi(\cdot)$ is here the normal distribution and $\chi^2(\cdot; d)$ the distribution of the sum of the squares of d independent standard normal variates[1].

➡ We simply drop all drawn Gaussian Sobol' vectors that lie outside r^*.

[1] The inverse χ^2 function can be evaluated as a special case of the inverse Gamma distribution function for which there is an extremely efficient implementation [DM87].

The pruning radius r^* as a function of dimensionality d and Gaussian standard deviations n.

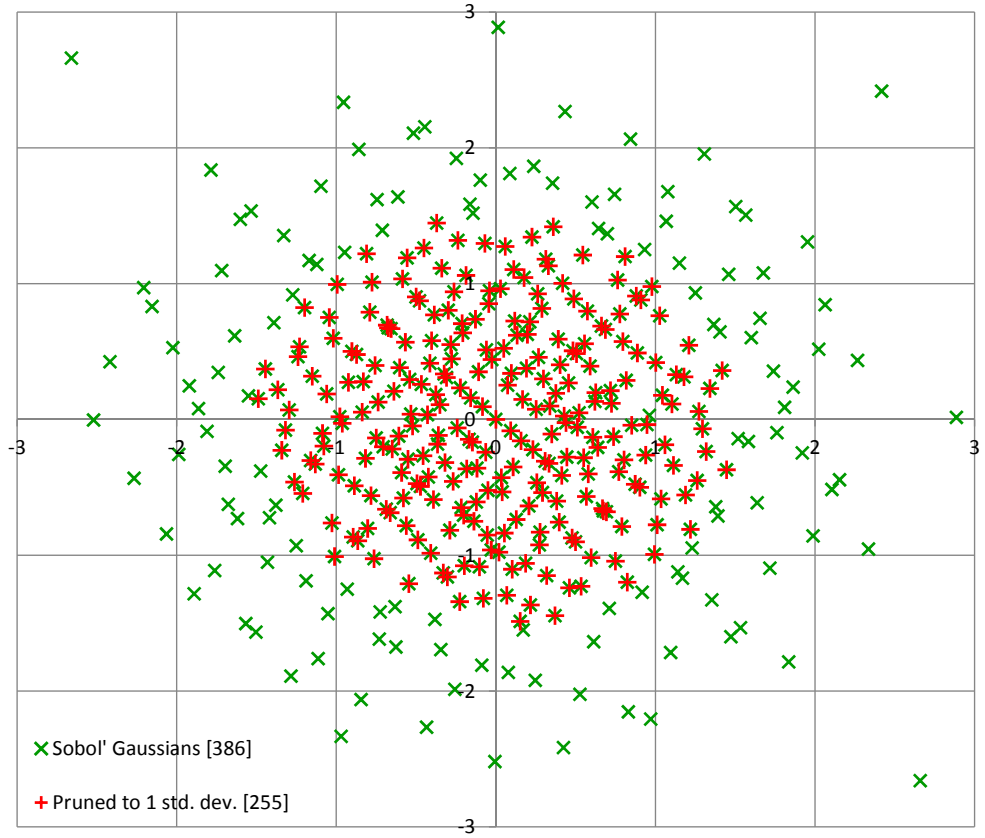


Latin-Hypercubing of the (perhaps pruned) cluster appears to help (in general):-

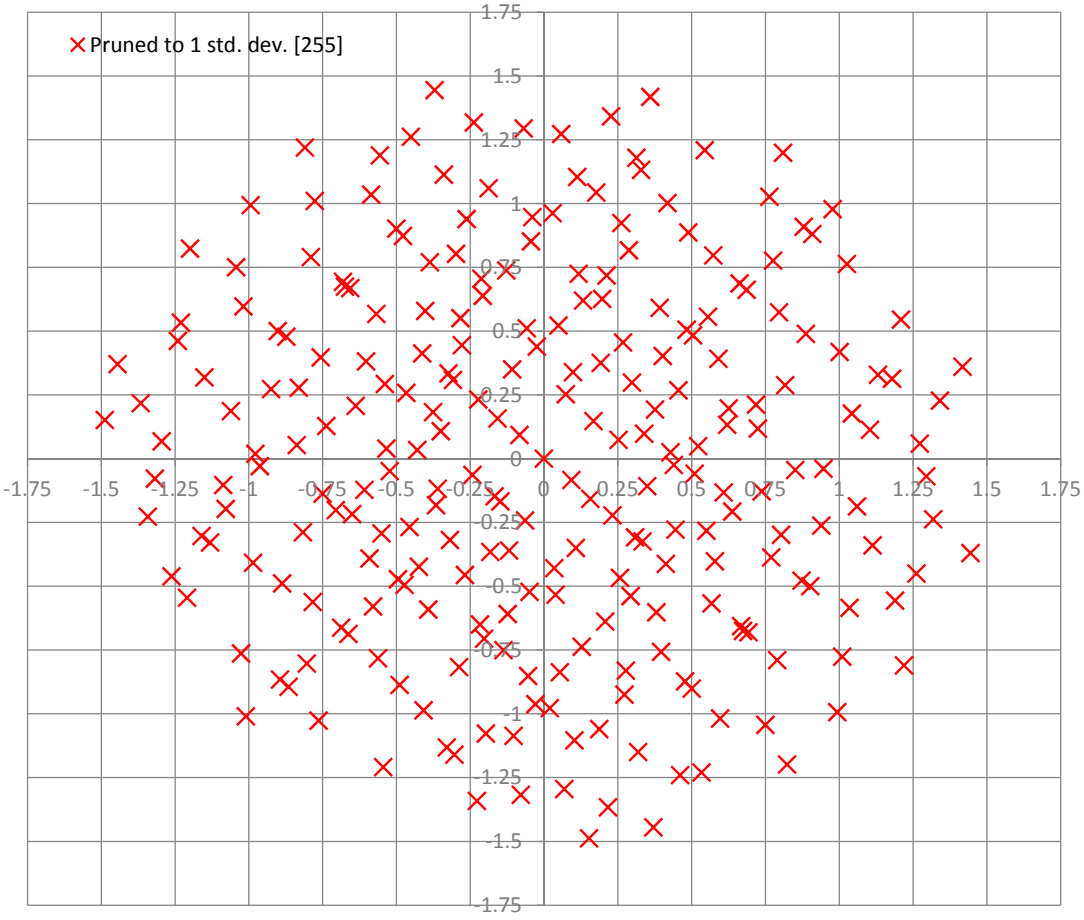
- For each dimension j from 1 to d, individually,
- sort the whole cluster by the j-th dimension (excluding the central node),
- and replace all the elements in the j-th dimension by standardised, (probability-)equidistant values (excluding the origin).

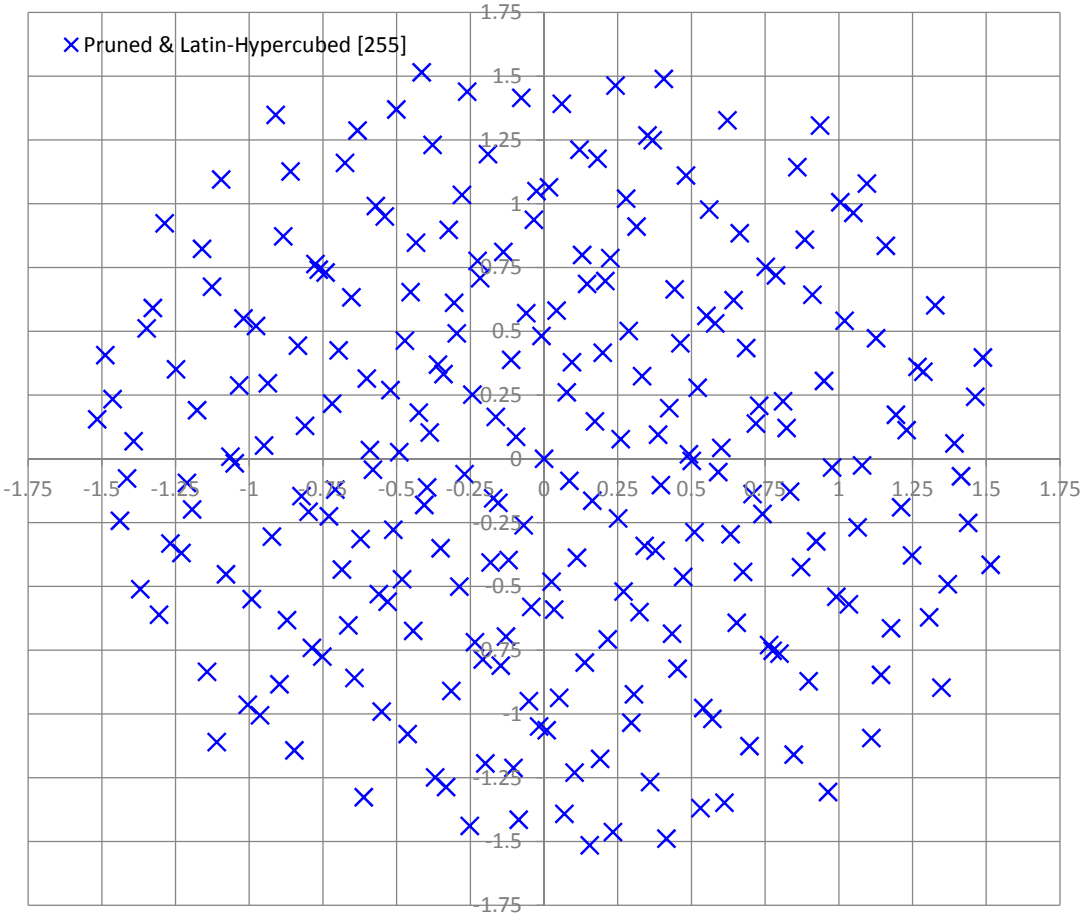
This used to be popular with pseudo-random Monte Carlo simulations before Sobol'-sampling superceded conventional Monte Carlo methods[2].

[2] Sobol' sampling has an approximate, asymptotic, version of the Latin hypercube built into it which obviates explicit Latin-Hypercubing.



$386/255 - 1 \approx 51\%$ fewer nodes when pruned!


Pruned to 1 std. dev. [255]
1.75
1.5
1.25
1
0.75
0.5
0.25
0
-0.25
-0.5
-0.75
-1
-1.25
-1.5
-1.75
-1.75 -1.5 -1.25 -1 -0.75 -0.5 -0.25 0 0.25 0.5 0.75 1 1.25 1.5 1.75


Pruned & Latin-Hypercubed [255]
1.75
1.5
1.25
1
0.75
0.5
0.25
0
-0.25
-0.5
-0.75
-1
-1.25
-1.5
-1.75
-1.75 -1.5 -1.25 -1 -0.75 -0.5 -0.25 0 0.25 0.5 0.75 1 1.25 1.5 1.75

As we saw, even with Latin-Hypercubing we may still have an unpleasantly irregular distribution of the cluster nodes.

One way to regularise the cluster further is *Elastic Cluster Relaxation*:-

1. Identify the *outer* nodes. These are the vertices of the convex hull. Consider these nodes, plus the origin (the *spot node*), as fixed.
2. Create a suitable point-to-point set of connections to form an (irregular) web over the cluster nodes.
3. Interpret the edges of the web as perfectly elastic springs, with all nodes (except those that were fixed) being only defined as vertices of the web by its topology.
4. *Relax the web!*

- In practice, the trickiest part of this procedure is step 2.
- A good candidate for the web generation is the *Delaunay triangulation*, which happens to be the "dual" of the *Voronoi tesselation*.
- The good news is that there is published software out there for the calculation of both the convex hull and the Delaunay triangulation in arbitrary dimensions (in principle) [BDH96].
- The bad news is that the computational effort for the Delaunay triangulation grows like
$$4^d \ .$$
- In a numerical experiment for 127 nodes, I was able to obtain the Delaunay triangulation up to 9 dimensions (though it took hours towards the end).
- In 10 dimensions, I gave up after one week.

The relaxation result is given by the minimisation of the total elastic spring energy

$$\mathcal{E} \;=\; \tfrac{1}{2} \sum_{(i,j)\in\mathcal{T}} \left|\boldsymbol{\xi}_i - \boldsymbol{\xi}_j\right|^2 \tag{6.5}$$

where $\mathcal{T}$ is the set of point index tuples that comprises the Delaunay triangulation over the variation of all the mobile vertices' coordinates.

The resulting linear system for the movable nodes separates into d identical individual positive definite[3] linear systems, one for each dimension.

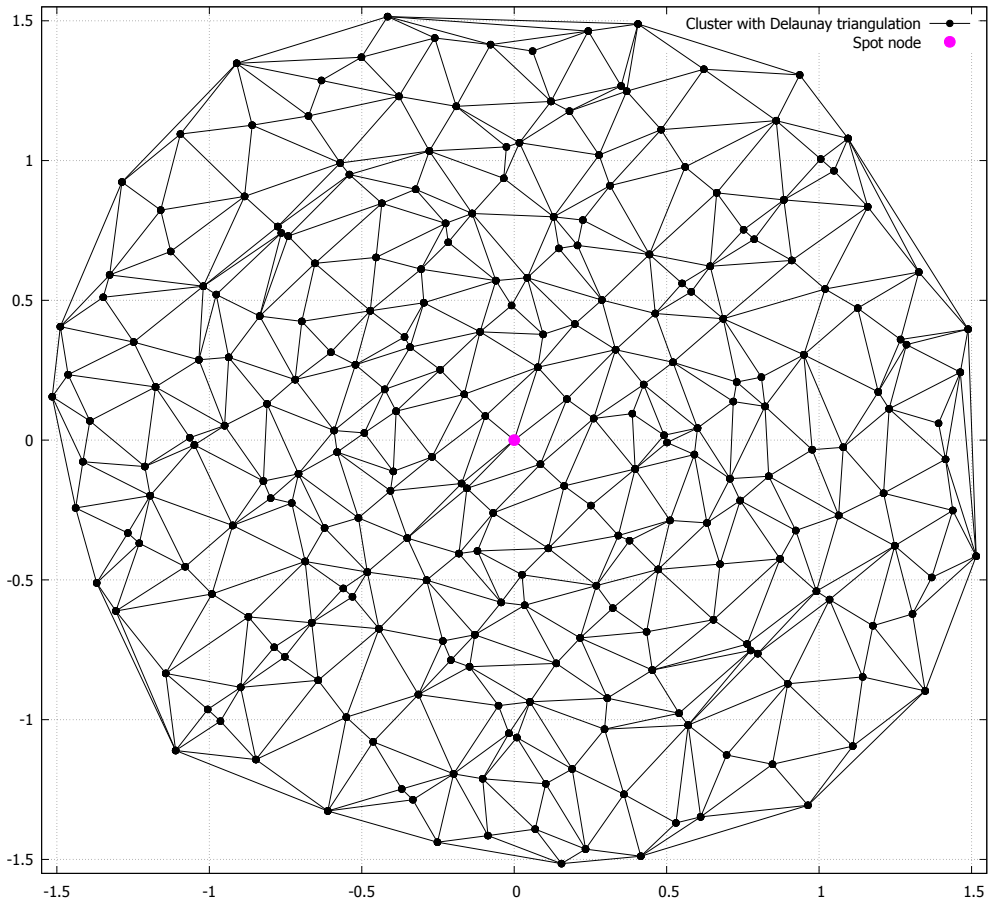
Let's see what this does for the *ugly* cluster shown on page 92 !

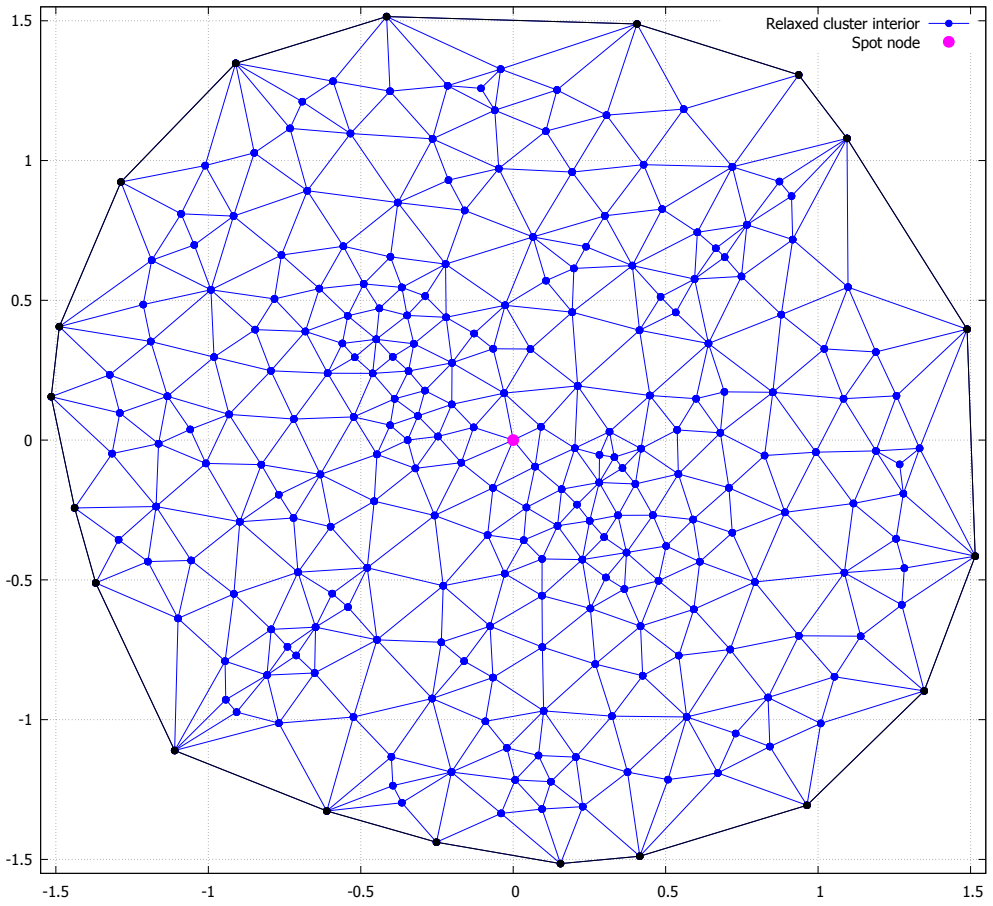
In fact, we'll do it twice since a repetition of the Delaunay triangulation after the first relaxation is not guaranteed to result in the same topology.

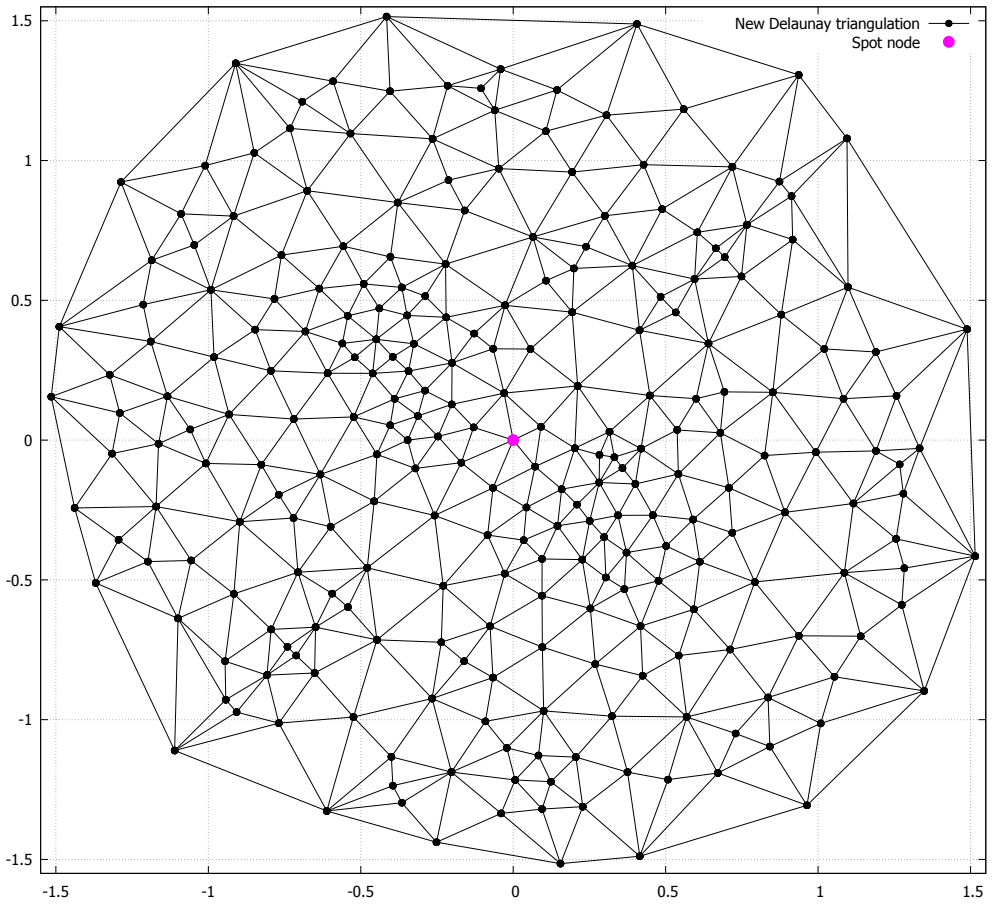
[3] This is easy to see since the energy can be written as

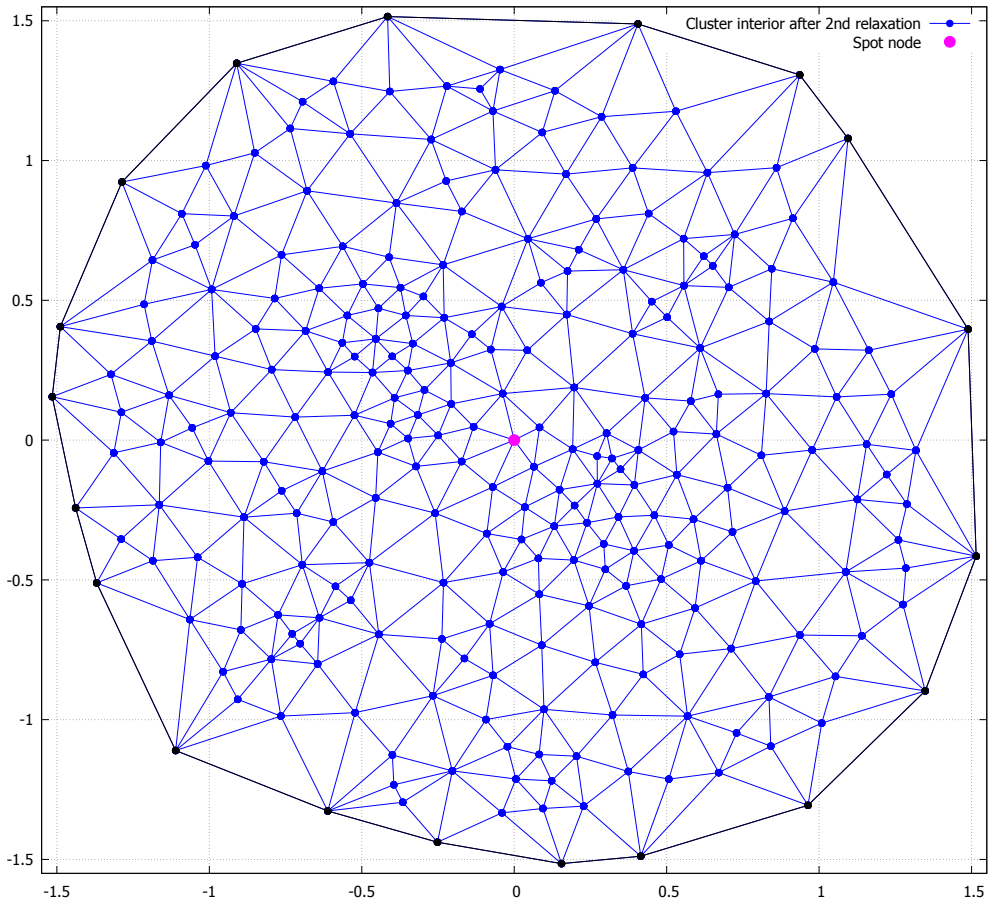
$$\tfrac{1}{2}\left|D\cdot\Xi\right|^2 \;=\; \tfrac{1}{2}\Xi^\top\cdot\left(D^\top\cdot D\right)\cdot\Xi \tag{6.6}$$

where D is a matrix whose rows each represent the distance calculation for one triangulation edge and Ξ is the matrix of the cluster coordinates.




Relaxed cluster interior
Spot node
1.5
1
0.5
0
-0.5
-1
-1.5
-1.5
-1
-0.5
0
0.5
1
1.5


New Delaunay triangulation
Spot node
1.5
1
0.5
0
-0.5
-1
-1.5
-1.5
-1
-0.5
0
0.5
1
1.5



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

$$\boldsymbol{x} \;\rightarrow\; \boldsymbol{\xi}_{\hat{k}(\boldsymbol{x};\mathcal{C})} \tag{6.7}$$

with

$$\hat{k}(\boldsymbol{x};\mathcal{C}) \;:=\; \arg\min_{k\in\{1,..,N\}} |\boldsymbol{x}-\boldsymbol{\xi}_k| \tag{6.8}$$

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

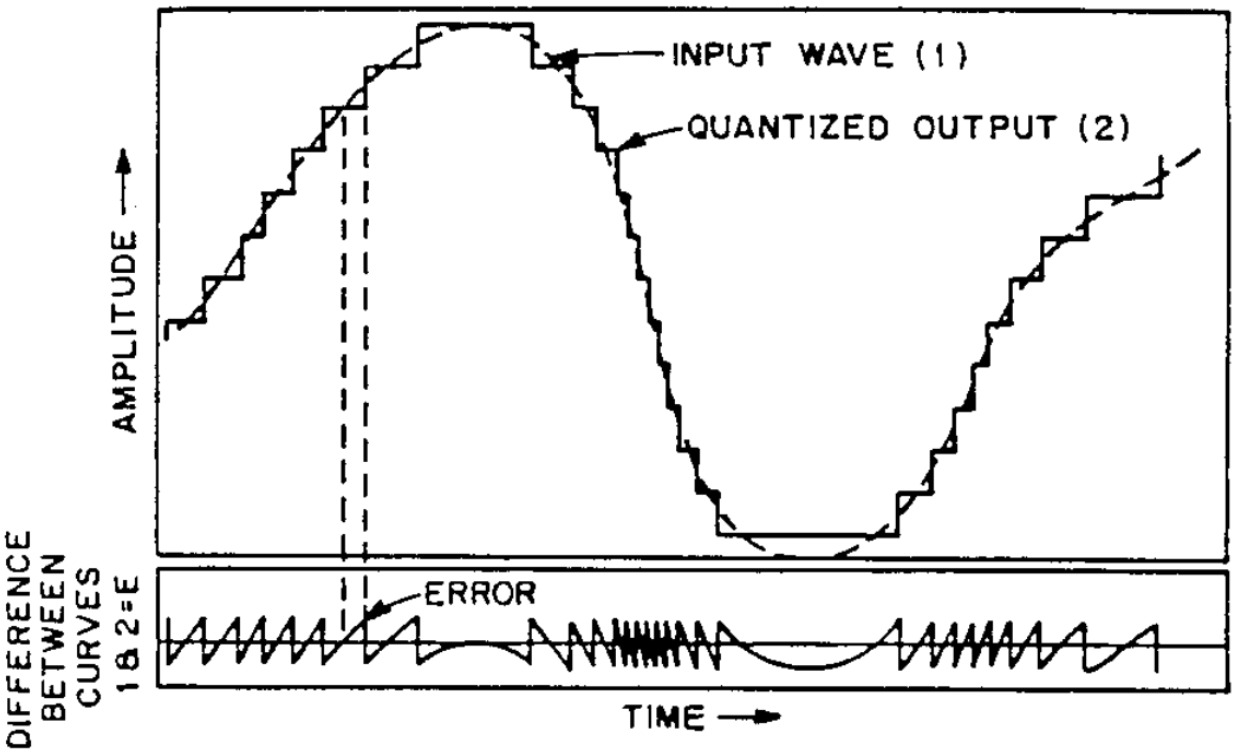
Then, we call the expectation

$$D(\mathcal{C}) \;:=\; \mathrm{E}\left[\left|\boldsymbol{x}-\boldsymbol{\xi}_{\hat{k}(\boldsymbol{x};\mathcal{C})}\right|^2\right] \tag{6.9}$$

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

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

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

$$\boldsymbol{\xi}_k \cdot \mathrm{E}\left[\mathbf{1}_{\left\{\hat{k}(\boldsymbol{x};\mathcal{C})=k\right\}}\right] \;=\; \mathrm{E}\left[\boldsymbol{x} \cdot \mathbf{1}_{\left\{\hat{k}(\boldsymbol{x};\mathcal{C})=k\right\}}\right] \tag{6.10}$$

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 $\mathcal{C}^*$ that satisfies (6.10) the *Minimum Distortion Cluster* though it is better known as the

Centroidal Voronoi Tesselation[4] .

[4]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 indicates the representation of a continuum by a subset of discrete values, without any statement about the rule that led to the choice of nodes.

Due to the considerable geometrical complications, there is in practice no competitive analytical method to compute $\mathcal{C}^*$ when d is more than two(-ish).

A simple algorithm attributed to Lloyd [Llo57] is to iterate

$$\boldsymbol{\xi}_k{}^{(n+1)} := \frac{\mathrm{E}\left[\boldsymbol{x}\cdot\mathbf{1}_{\left\{\hat{k}(\boldsymbol{x};\mathcal{C}^{(n)})=k\right\}}\right]}{\mathrm{E}\left[\mathbf{1}_{\left\{\hat{k}(\boldsymbol{x};\mathcal{C}^{(n)})=k\right\}}\right]} = \mathrm{E}\left[\boldsymbol{x}\,\middle|\,\hat{k}(\boldsymbol{x};\mathcal{C}^{(n)})=k\right]. \qquad (6.11)$$

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 (6.11) (semi-)analytically whence we resort to a **good old-fashioned Sobol'-Monte-Carlo evaluation**[5].

[5] Once more, 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 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 comprised by Gaussian Sobol'-Monte-Carlo draws, say, of size N =255.

Then, in each iteration (of, say, 127), zero-out a workspace of N vectors $\boldsymbol{\Xi}_k \in \mathbb{R}^d$, and zero-out N counter variables m_k for $k = 1, .., N$ and:-

- Loop over all of the M sampling points $\boldsymbol{x}_i$ for $i = 1, .., M$ to find the nearest (previous) cluster node $\boldsymbol{\xi}_{\hat{k}}{}^{(n)}$ and, having identified this nearest cluster node index $\hat{k} = \hat{k}(\boldsymbol{x}_i, \mathcal{C}^{(n)})$, set

$$\boldsymbol{\Xi}_{\hat{k}} \mathrel{+}= \boldsymbol{x}_i$$

$$m_{\hat{k}} \mathrel{+}= 1$$

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

$$\boldsymbol{\xi}_k{}^{(n+1)} := \boldsymbol{\Xi}_k / m_k$$

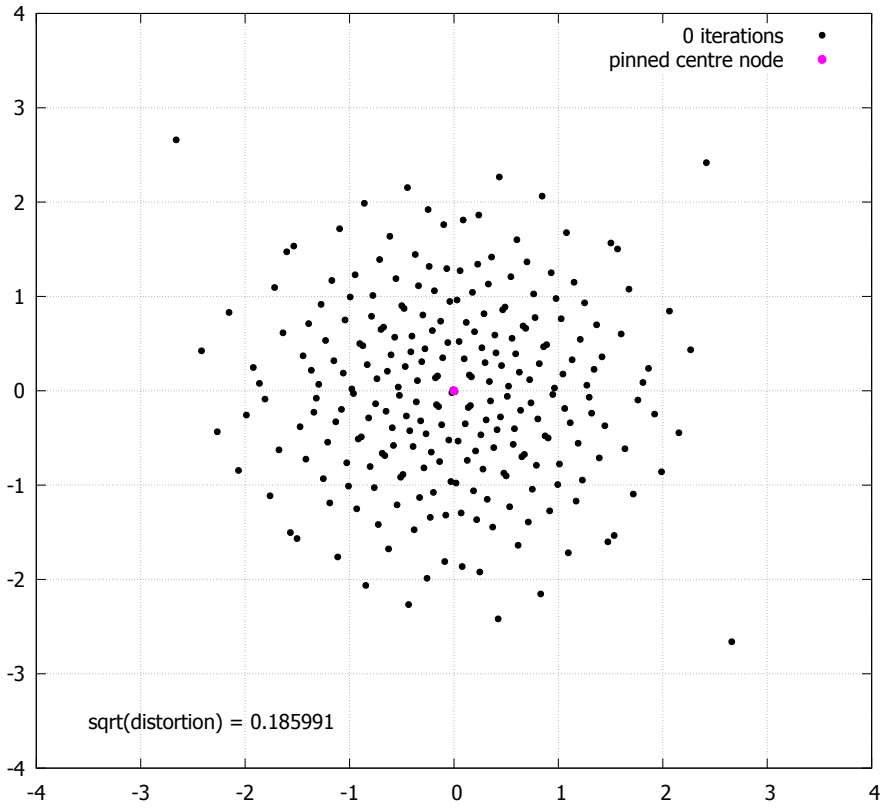
for $k = 1, .., N$.

Notes:-

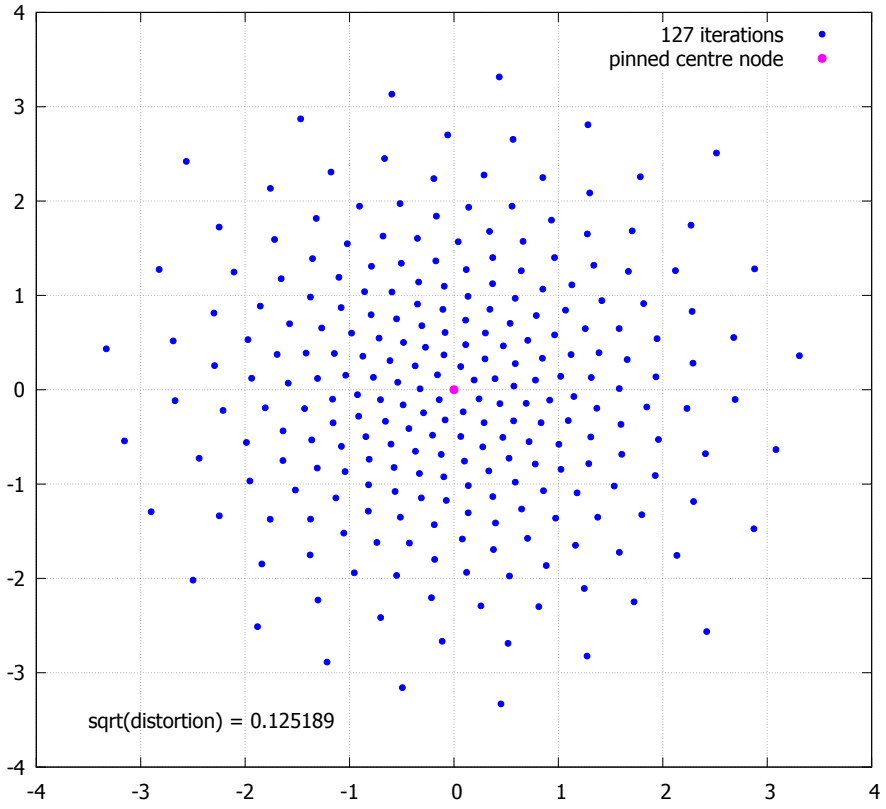
- The outer iteration (6.11) *may not converge (!)*. This is due to the discrete underlying sampling set used to evaluate the conditional expectation.
- **The above mentioned non-convergence is nothing to worry about!** It simply means that the algorithm eventually just cycles over a *discrete* set of equally good estimates for $\mathcal{C}^*$ for the given size N, dimensionality d, and Gaussian-Sobol' sampling set.
- The higher the dimensionality d, alas, the more outer iterations you will wish to use to attain a satisfactory cluster.
- 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.
- You may want to grow the cluster N size gently with increasing dimensionality d.
- In higher dimensions, such optimum-representation cluster computation techniques are part of a range of ***machine learning algorithms***, e.g., *"k-Means Clustering"*.
- You may find the oft-praised "Anderson acceleration" technique of little practical use.
- See Wikipedia [Llo18] for references to contemporary acceleration techniques and to links as to how these methods are also used in finite element calculations.

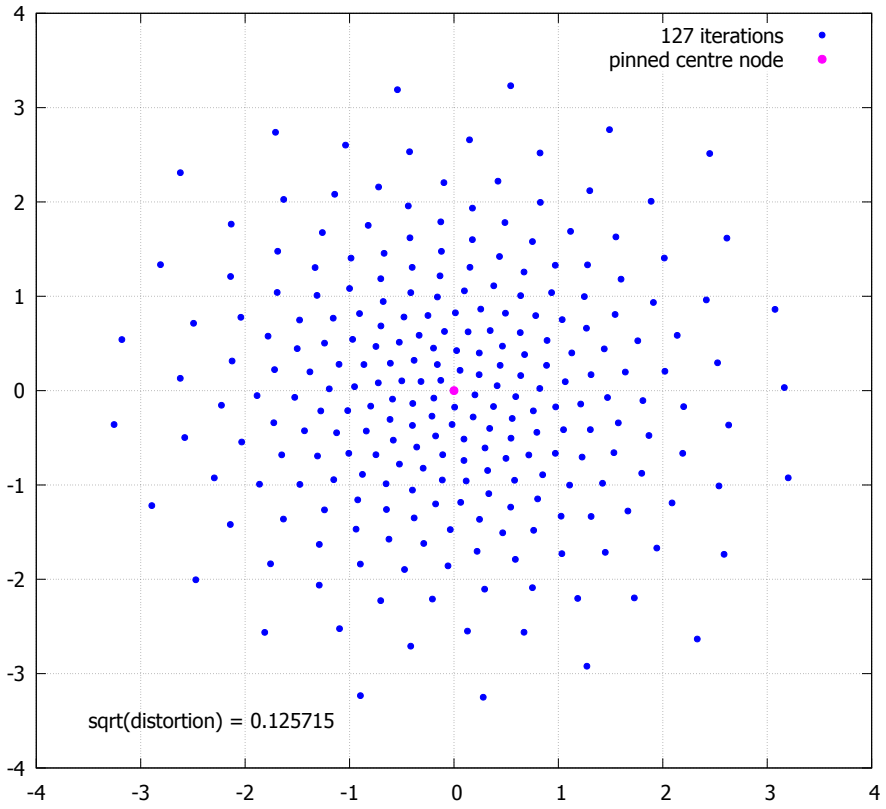
For $n_{\text{cluster}} = 255, n_{\text{sampling}} = 131071, n_{\text{dim}} = 2$, starting with:



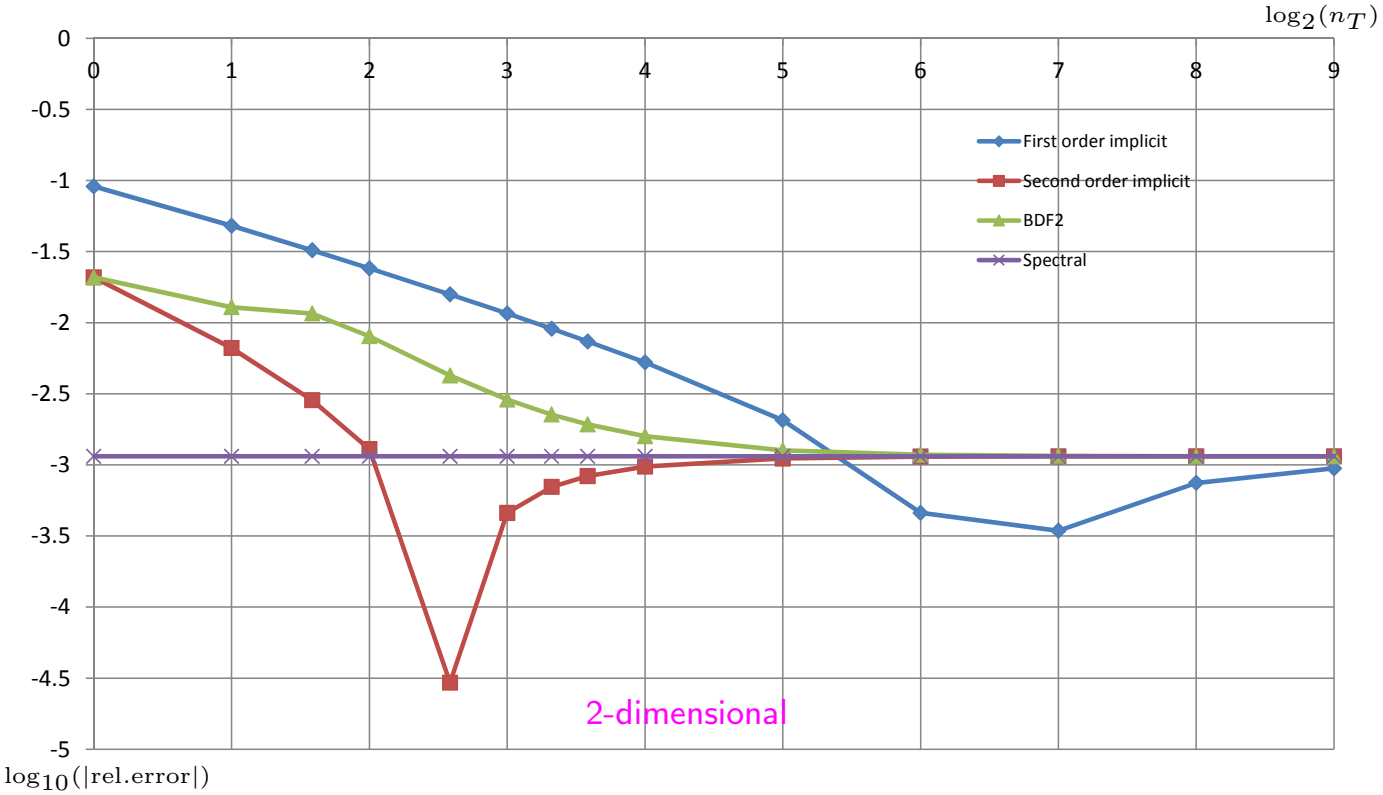
After 127 iterations, we arrive at:



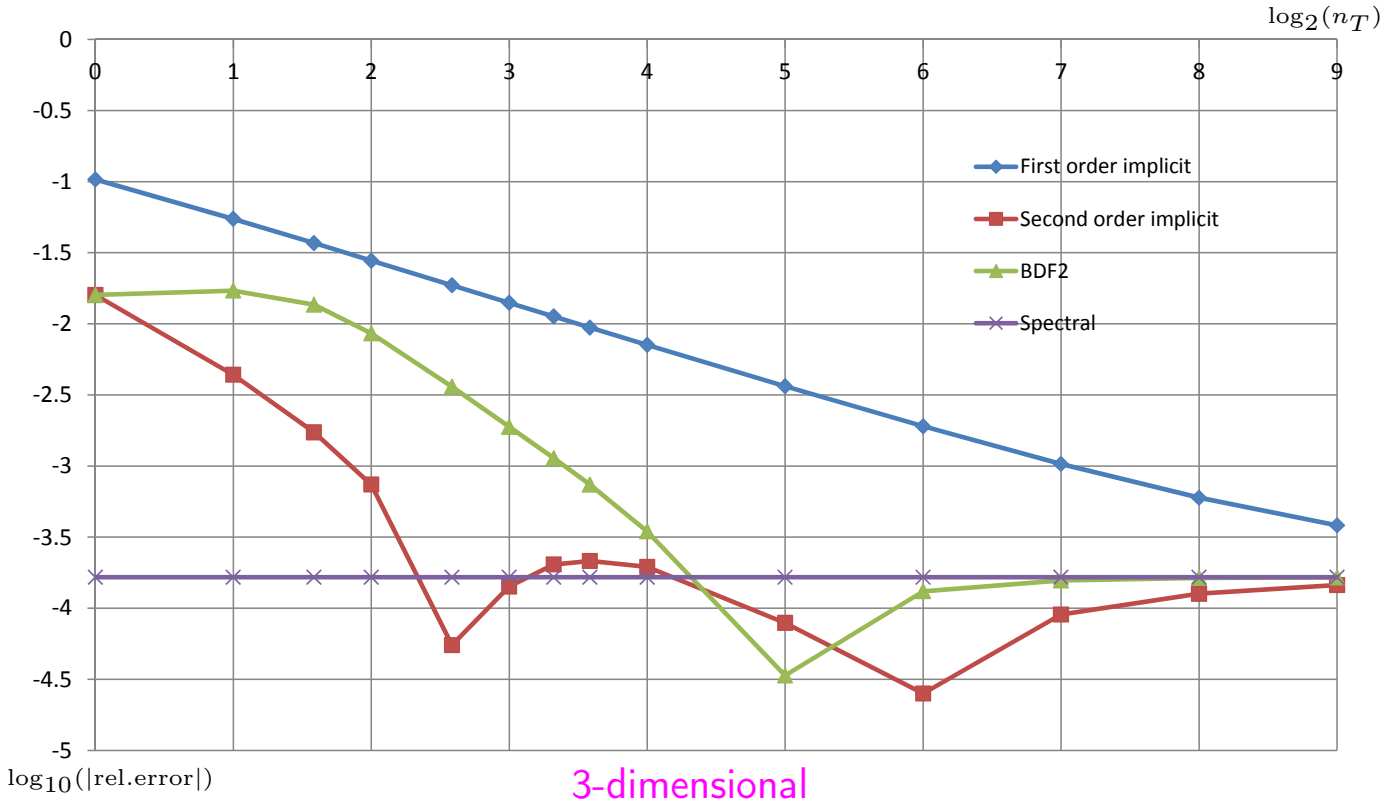
Here is an alternative (different algorithm, ∼3 times faster):



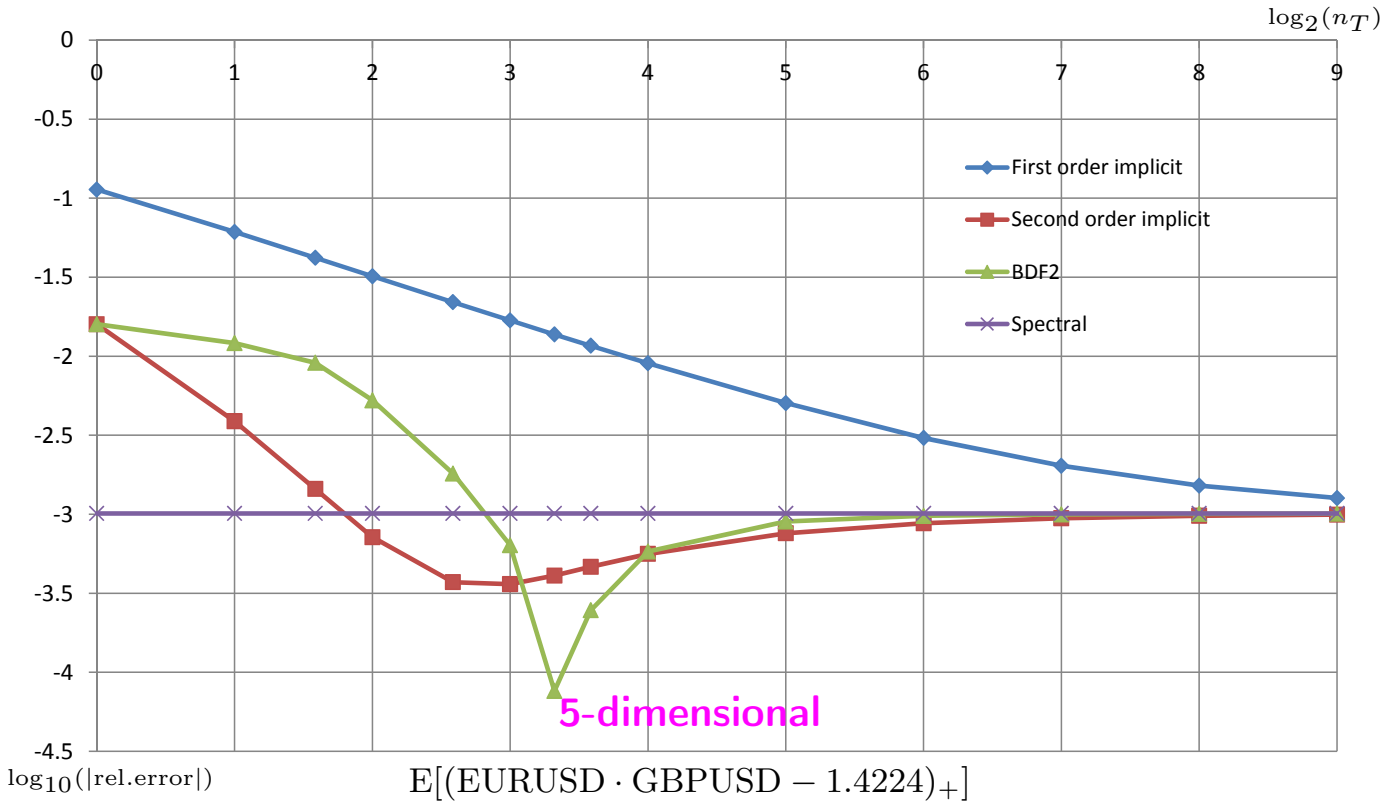
1Y ATM composite option, Black-Scholes, $n_{\text{cluster}} = 255$, $n_{\text{sampling}} = 32767$.



1Y ATM FX option, LGM model, $n_{\text{cluster}} = 255$, $n_{\text{sampling}} = 32767$.



1Y composite option, LGM model, $n_{\text{cluster}} = 511$, $n_{\text{sampling}} = 32767$.



A recent development is *to evaluate the spatial differential operator* $\mathcal{L}$ in

$$\partial_t f + \mathcal{L} f \;=\; 0 \tag{7.1}$$

on each cluster node *not directly*, i.e., analytically, on radial basis functions, but *as a weighted average of the function value over a judiciously chosen nearby subcluster*.

This was inspired by Bellman's *Differential Quadrature* [BKC72, BM96].

In a nutshell:

- A standard (spatial) finite-differencing method approximates, for instance, the term $\frac{\partial^2 f}{\partial x^2}$ by a weighted sum of the function value at the central node and its two nearby neighbours in the x-direction on a regular lattice:

$$\frac{\partial^2 f}{\partial x^2} \;\approx\; w_1 \cdot f(x - \Delta x) + w_2 \cdot f(x) + w_3 \cdot f(x + \Delta x) \tag{7.2}$$

- The weights are derived from a local Taylor expansion.
- With the Taylor expansion being a polynomial expansion, this is tantamount to demanding that the approximation is exact for the three test functions $f_1(x) = 1$, $f_2(x) = x$, and $f_3(x) = x^2$.
- This means the weights are derived from the linear system

$$\begin{pmatrix} f_1(x-\Delta x) & f_1(x) & f_1(x+\Delta x) \\ f_2(x-\Delta x) & f_2(x) & f_2(x+\Delta x) \\ f_3(x-\Delta x) & f_3(x) & f_3(x+\Delta x) \end{pmatrix} \cdot \begin{pmatrix} w_1 \\ w_2 \\ w_3 \end{pmatrix} \;=\; \begin{pmatrix} f_1''(x) \\ f_2''(x) \\ f_3''(x) \end{pmatrix} \tag{7.3}$$

- Bellman's Differential Quadrature is the extension of this concept to all the nodes in the lattice line using a complete set of polynomial test functions to the maximum order attainable.
- As usual, Chebyshev polynomials turn out to be very useful.

RBF-DQ [Tol00, WS02, TS03] is the translation of this idea to:-

- focus on the full differential operator $\mathcal{L}f$ monolithically without splitting it into individual derivative terms.
- make the test functions radial basis functions centered in the nodes.

Local RBF-DQ [SDY03] then:-

- evaluates the differential operator $\mathcal{L}f$ on any one node only by weighting the function values over judiciously selected nearby *supporting nodes*.
- use only the RBFs centered in the *supporting nodes* as test functions.

Alas,

- people have started calling this localised derivative collocation technique by the name "*RBF-FD*".
- This was meant to help conjure up the notion of a local stencil for the evaluation of $\mathcal{L}f$.
- But as we clearly see, this method has nothing to do with the calculation of "*finite differences*".
- "RBF-FD" is a misnomer just like the abhorrent term "quasi-random".

Caveat emptor:-

- All that was previously said about radial basis functions still applies!
- This includes the need for a suitably sized shape parameter,
- the need to choose among the many RBF types, and so on.
- In addition, one needs to choose the local stencil for each and every node!
- And even in 2 dimensions this may be as large as 37 nearby nodes [FF15a].

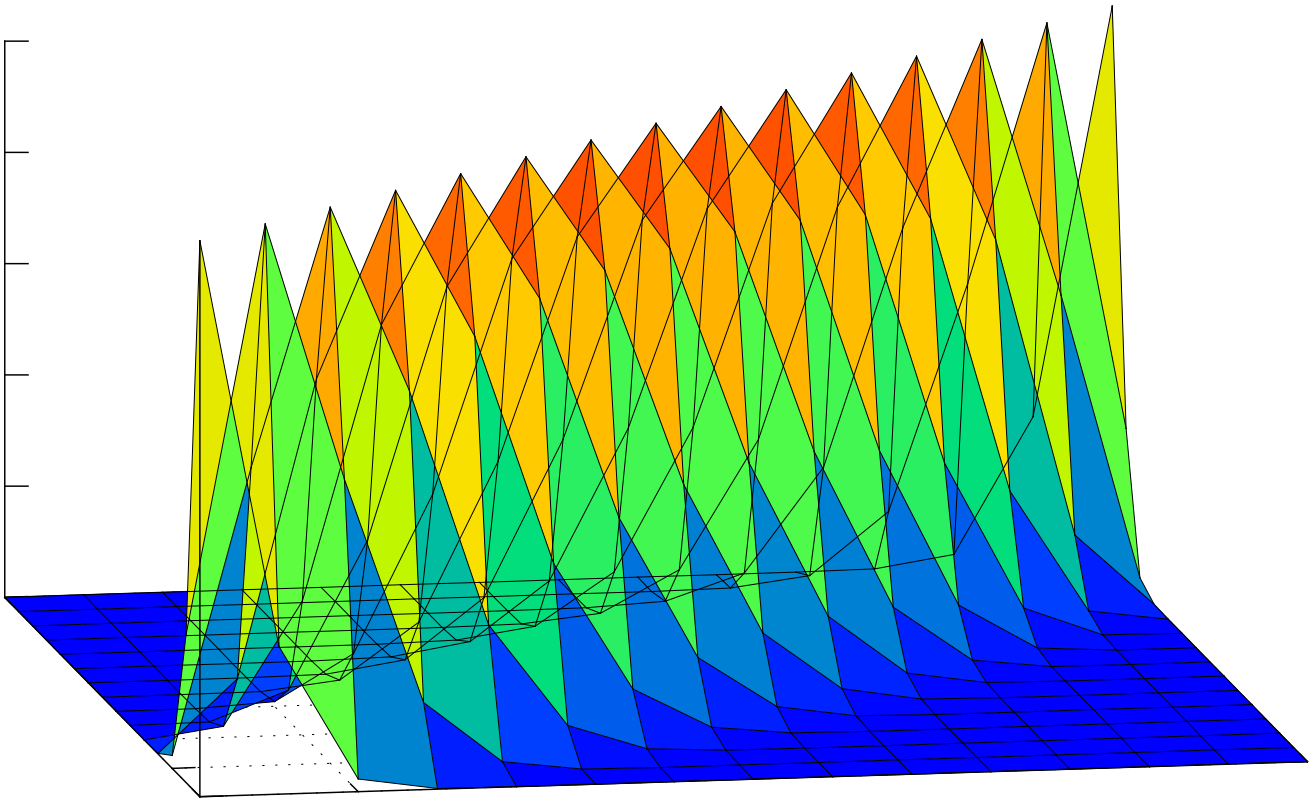
However, there is a lot of promising research on Local RBF-DQ methods!

➡ And by virtue of its *locally dispersed support nodes* for $\mathcal{L}f$, it *may* even *permit the use of **explicit** time integration schemes!*

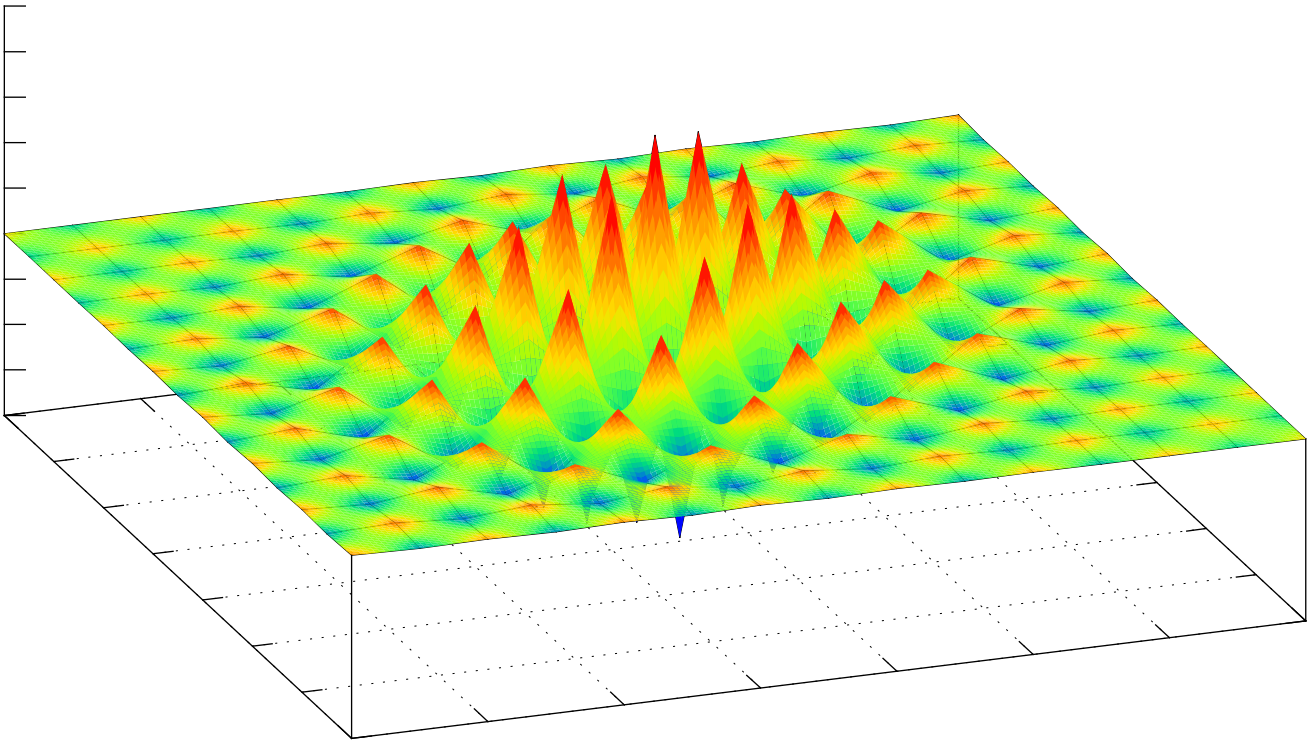
It remains to be seen...

THE END

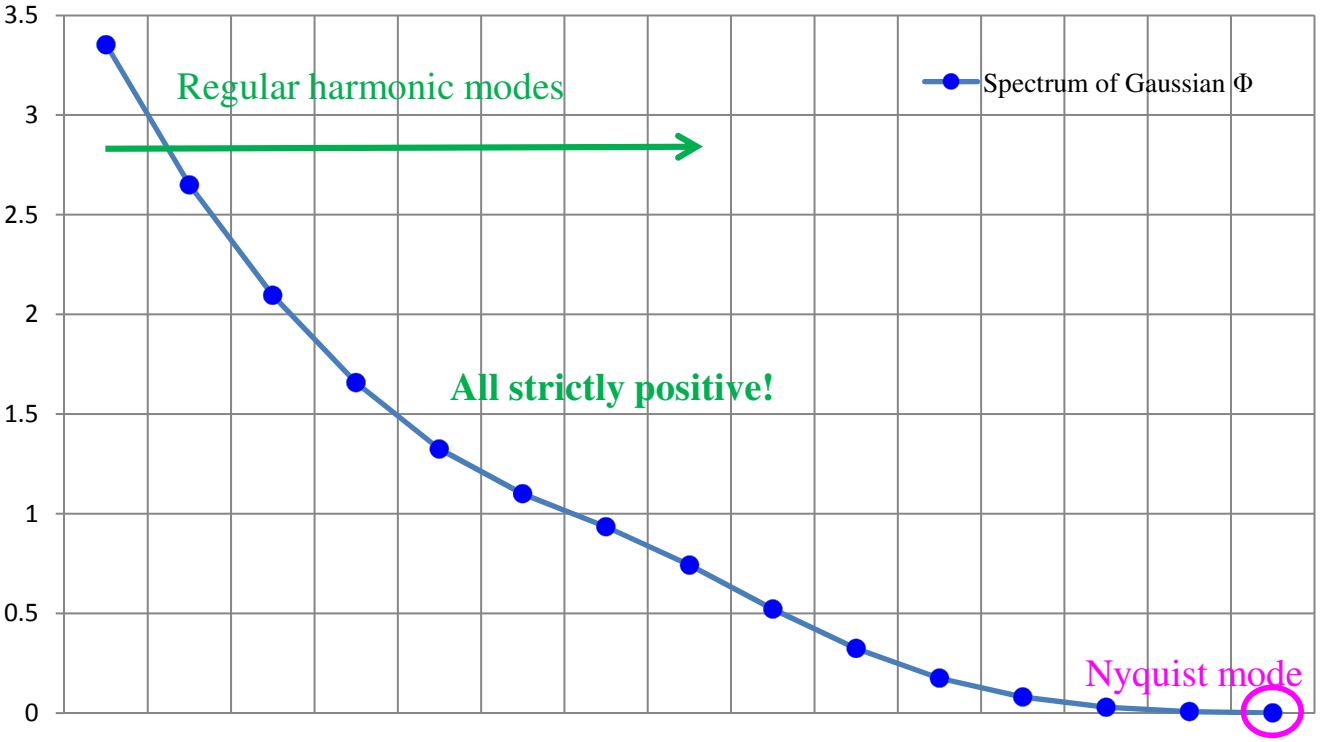
With $\phi(r,\epsilon) = \mathrm{e}^{-(\epsilon r)^2}$, we obtain for Φ (with $\epsilon = 3$):



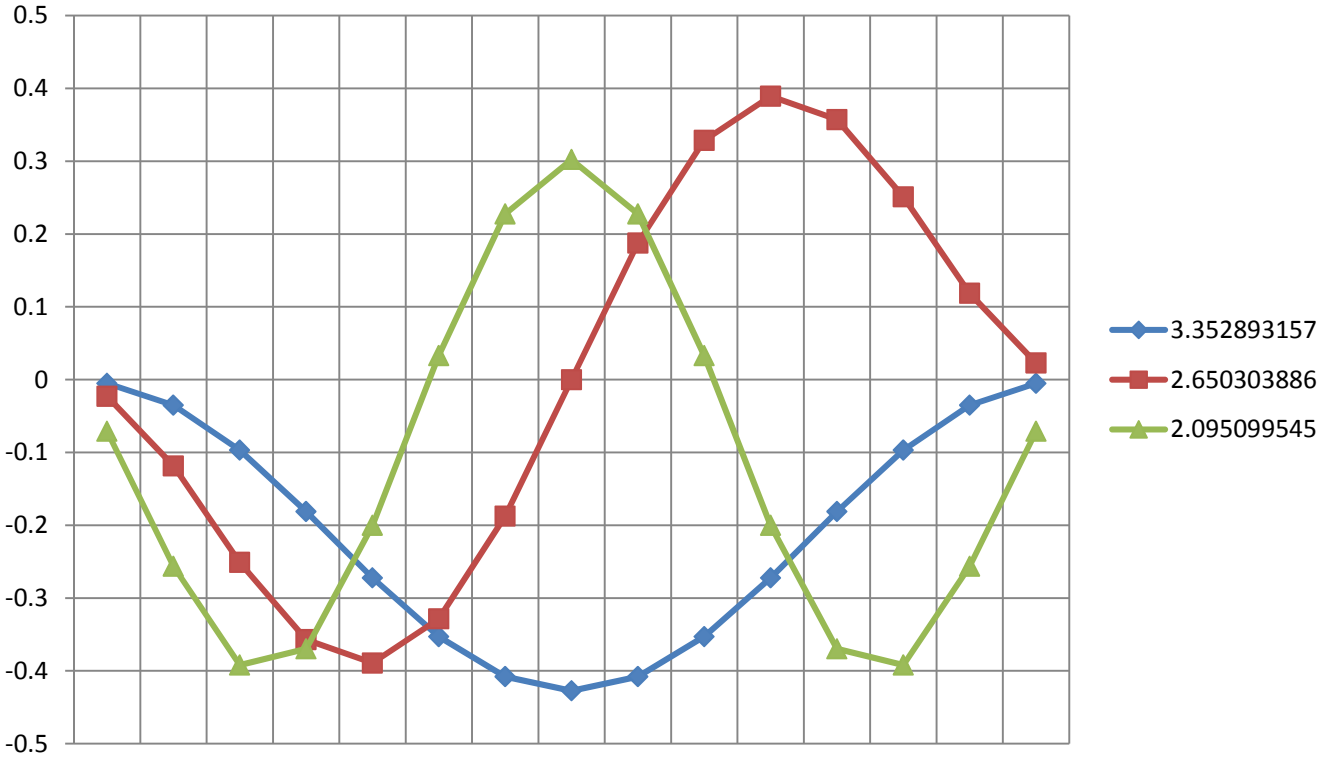
Gaussian Φ^{-1} matrix:



The spectrum of the Gaussian Φ matrix:

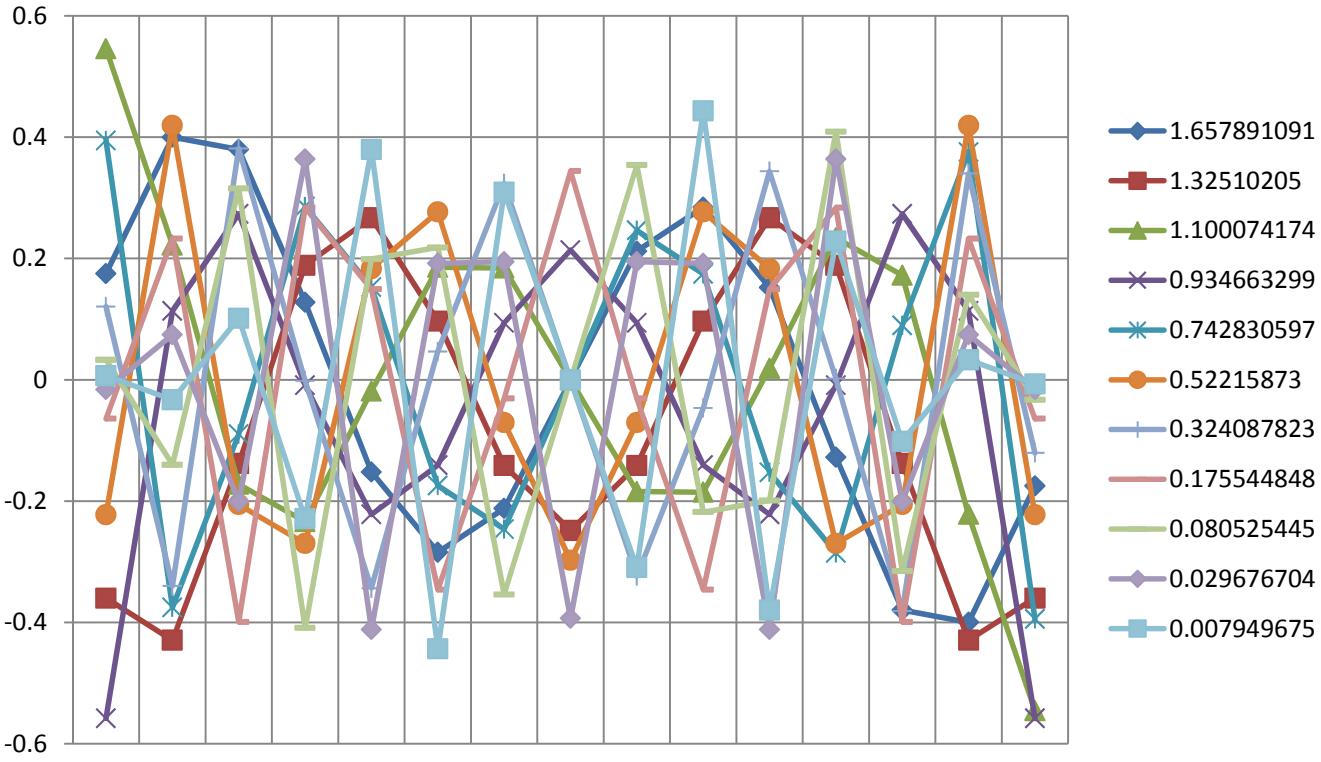


The first three harmonic modes of Φ and their eigenvalues:

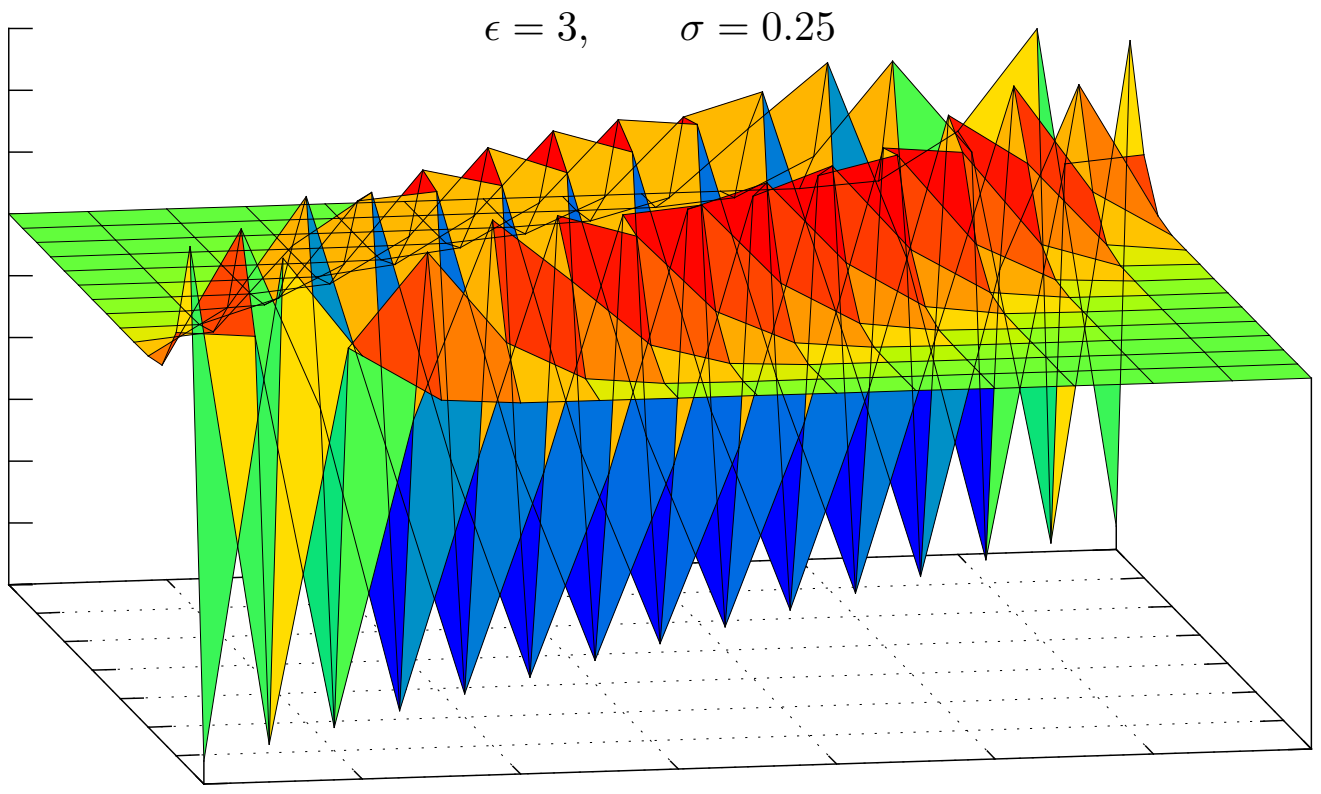


The Nyquist mode of Φ:

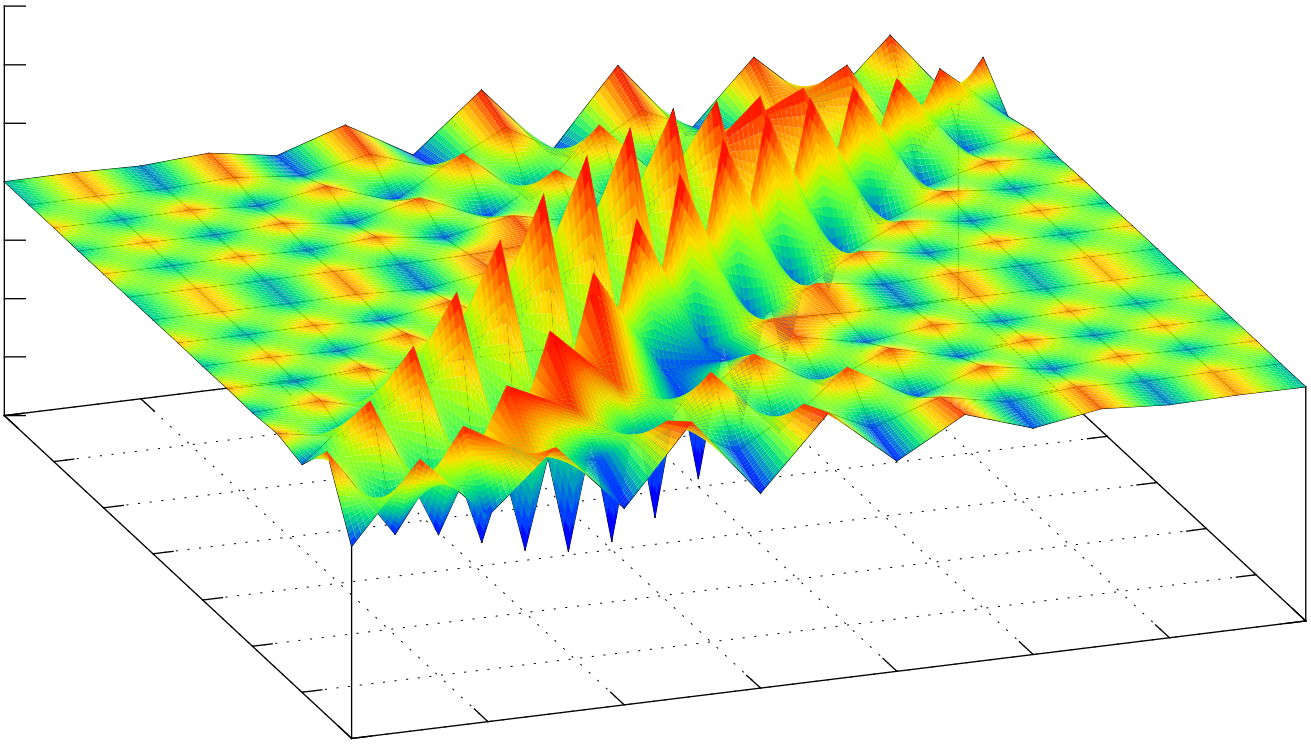
And all other modes of Φ in between:



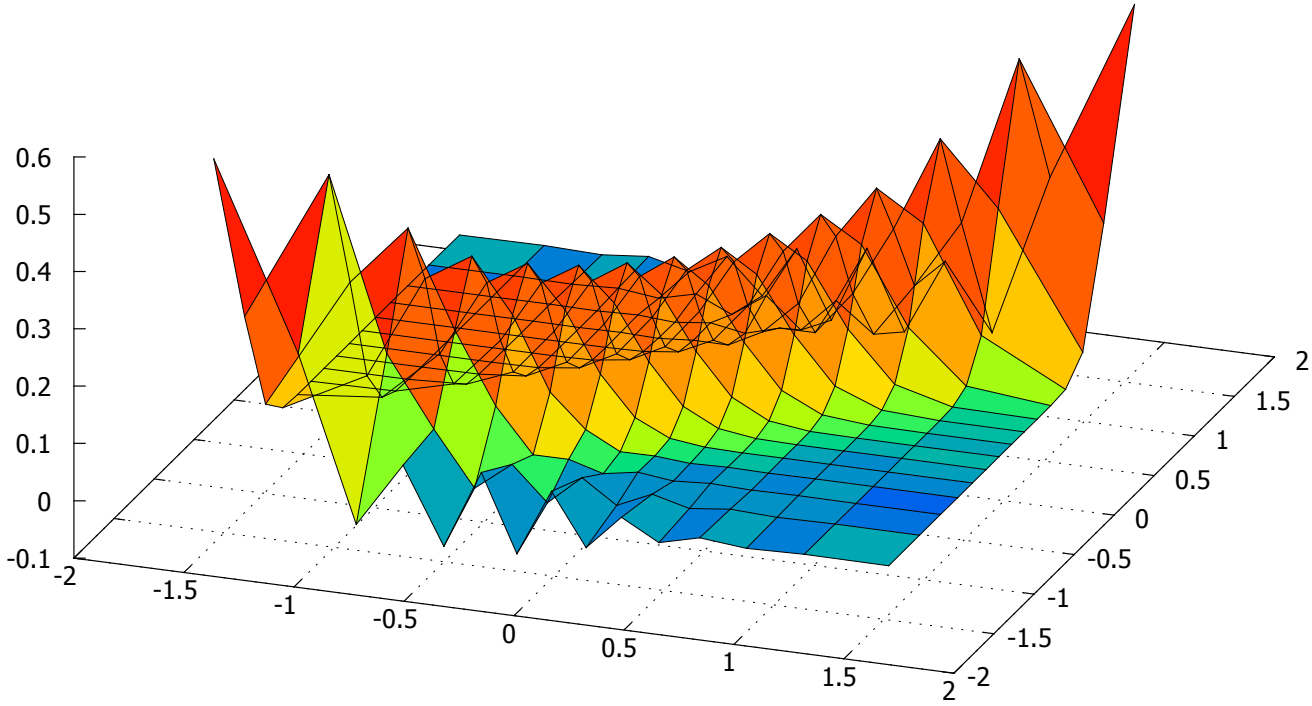
$\mathcal{L}\Phi$ for a standard diffusion:



The generator matrix $G = \mathcal{L}\Phi \cdot \Phi^{-1}$:

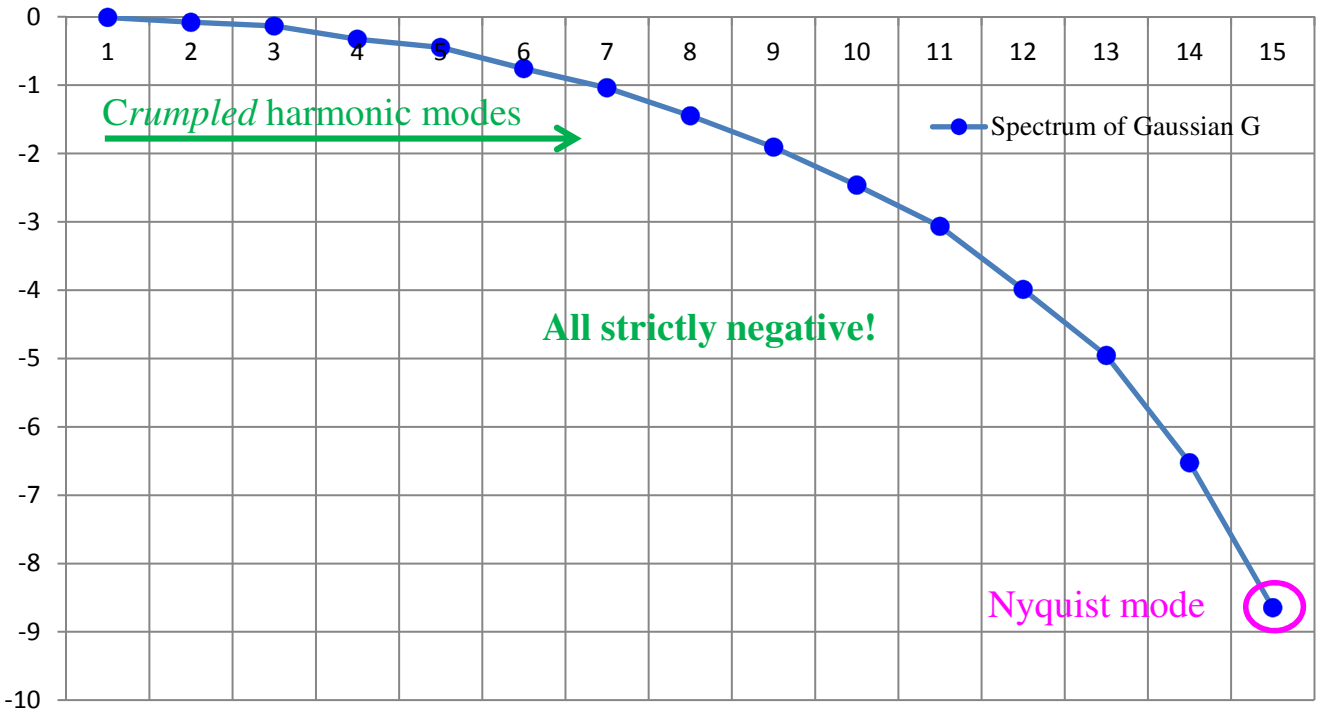


The (backward) propagator $\mathrm{e}^{\Delta t \cdot G}$ over actual $\boldsymbol{\xi}$ coordinates (with $\Delta t = 1$):

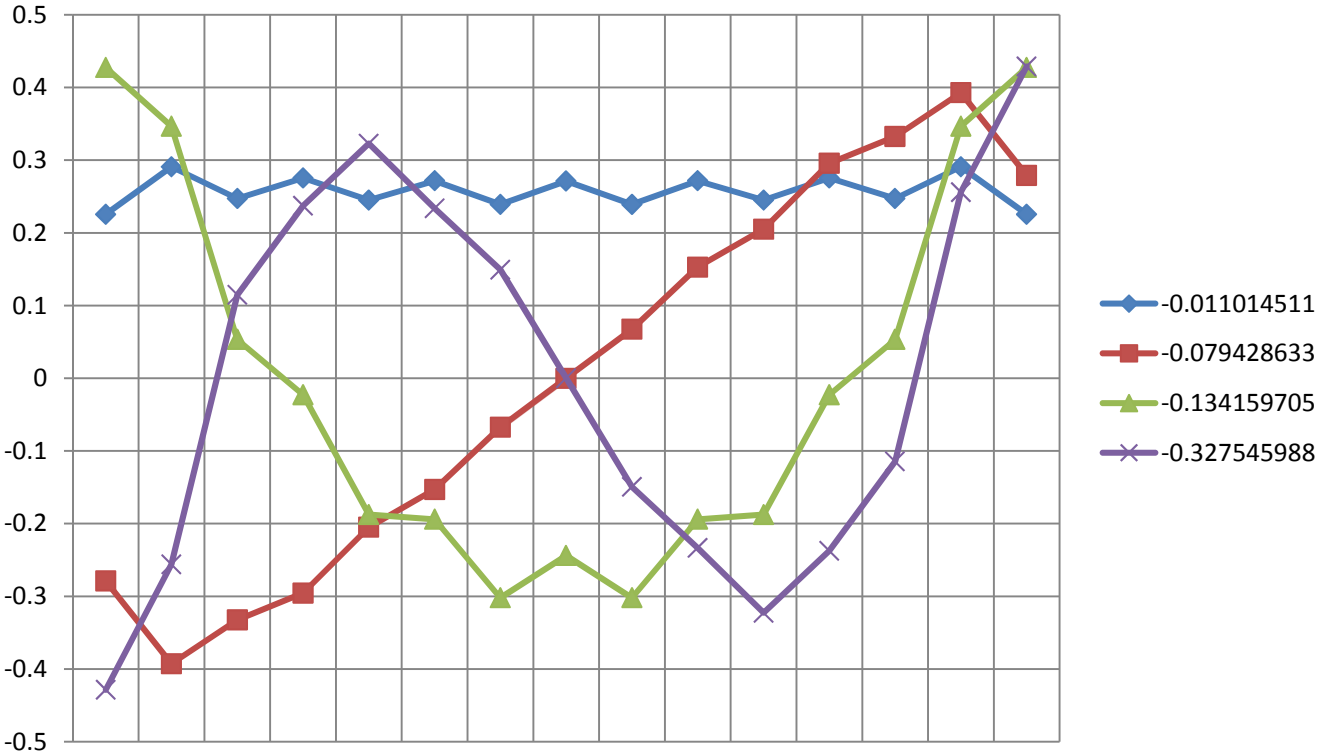


A Viking ship in rough water...

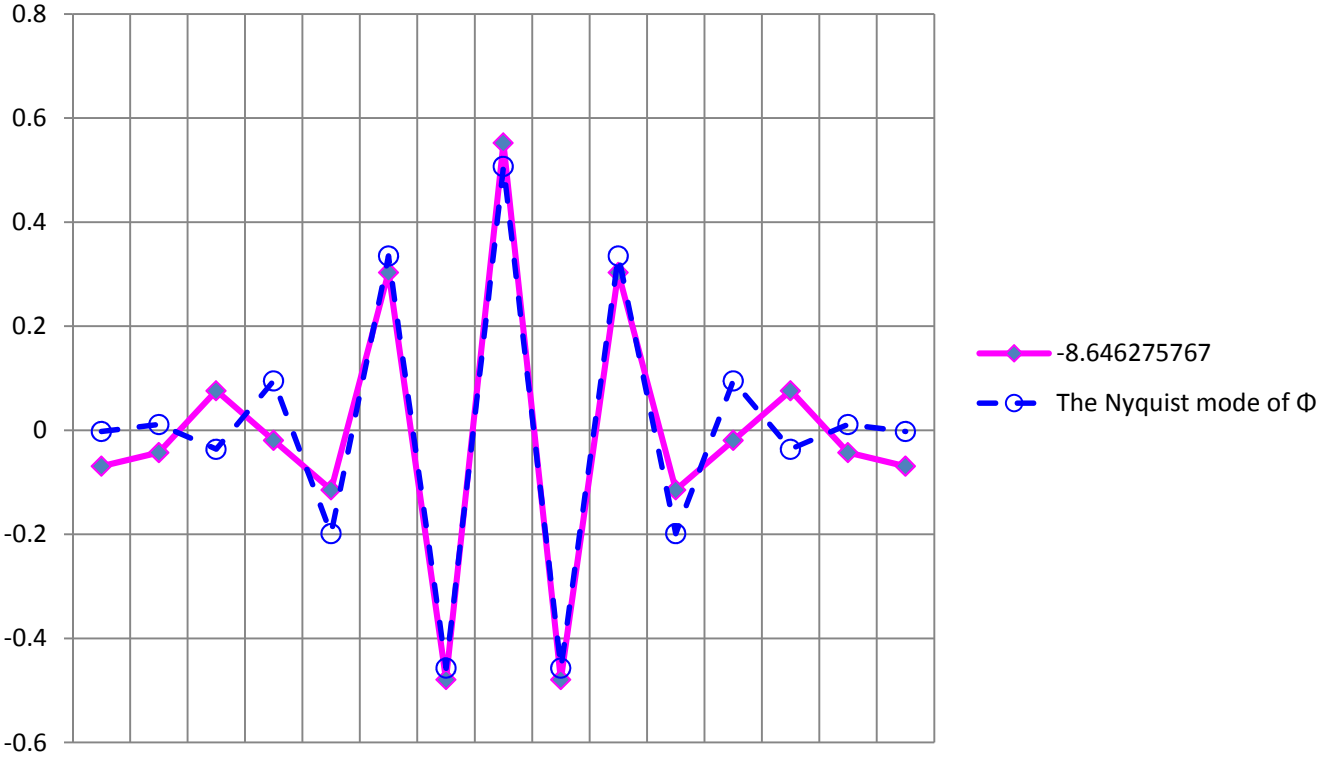
The spectrum of the Gaussian generator matrix $G = \mathcal{L}\Phi \cdot \Phi^{-1}$:



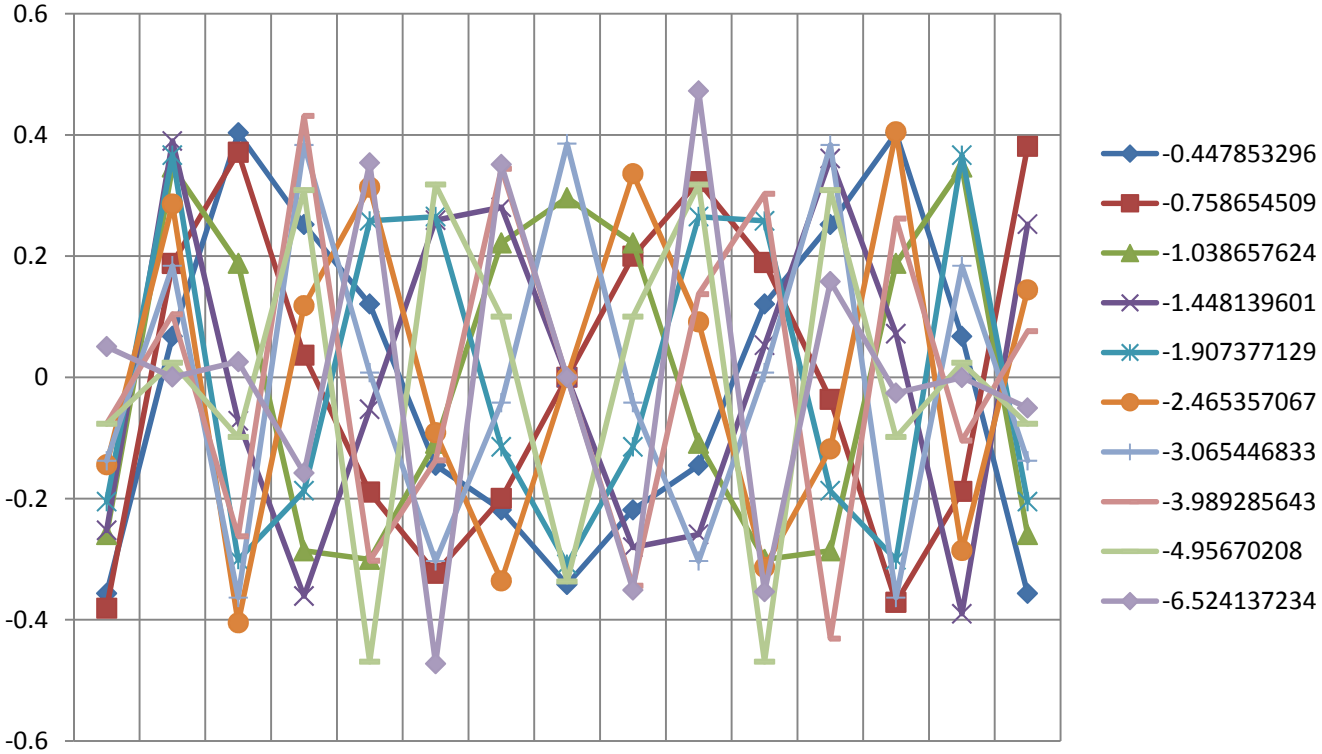
The first four modes of G and their eigenvalues:



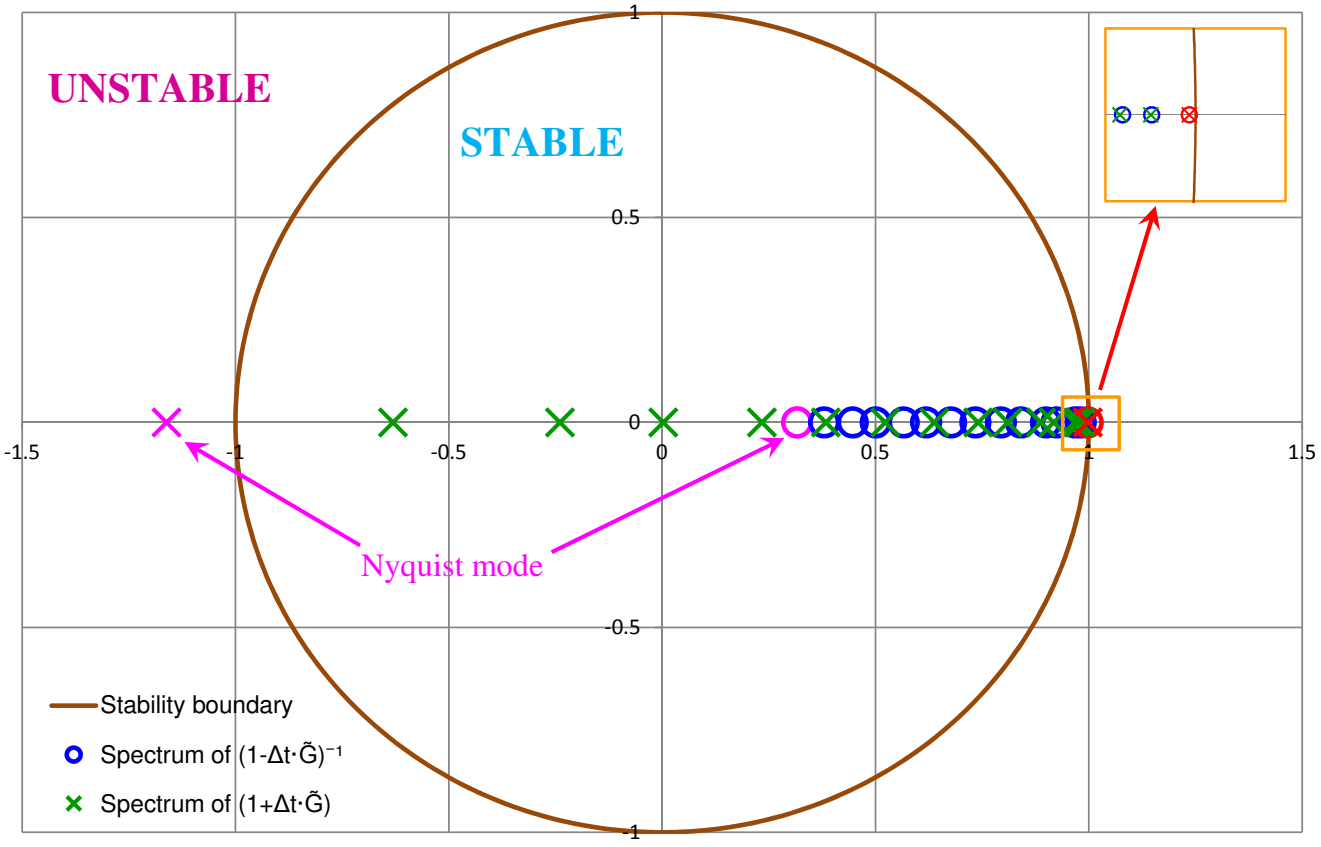
The Nyquist mode of G:



And all other modes of G in between:



And finally, the first order scheme stabilities with $\Delta t = 0.25$:



References

[BDH96] C.B. Barber, D.P. Dobkin, and H. Huhdanpaa.
The Quickhull algorithm for convex hulls.
ACM Transactions on Mathematical Software, 22(4):469–483, 1996.
www.qhull.org/.

[BKC72] R. Bellman, B.G. Kashef, and J. Casti.
Differential quadrature: A technique for the rapid solution of nonlinear partial differential equations.
Journal of Computational Physics, 10(1):40–52, 1972.

[BL88] D.S. Broomhead and D. Lowe.
Multivariable Functional Interpolation and Adaptive Networks.
Complex Systems, 2(1):321–355, 1988.

[BM96] C.W. Bert and M. Malik.
Differential quadrature method in computational mechanics: a review.
Applied Mechanics Reviews, 49:1–28, 1996.

[BW04] R. Baule and M. Wilkens.
Lean Trees — A General Approach for Improving Performance of Lattice Models for Option Pricing.
Review of Derivatives Research, 7(1):53–72, 2004.
www.fernuniversität-hagen.de/bwlbuf/uploads/allgemein/sonstiges/lean_trees.pdf.

[CH52] C.F. Curtiss and J.O. Hirschfelder.
Integration of Stiff Equations.
Proceedings of the National Academy of Sciences of the United States of America, 38(3):235–243, 1952.

[CS09] M.E. Chenoweth and S.A. Sarra.
A numerical study of generalized multiquadric radial basis function interpolation.
SIAM Undergraduate Research Online, 2:58–70, 2009.

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

[DM87] A. R. DiDonato and A. H. Morris.
Incomplete gamma function ratios and their inverse.
ACM TOMS, 13:318–319, 1987.
http://www.netlib.org/toms/654.

[Fas11] G.E. Fasshauer.
Positive definite kernels: past, present and future.
Dolomite Research Notes on Approximation, 4:21–63, 2011.
num.math.uni-goettingen.de/meshless/abstracts/fasshauer.pdf.

[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.

[FM12] G.E. Fasshauer and M.J. McCourt.
Stable Evaluation of Gaussian Radial Basis Function Interpolants.
SIAM J. Sci. Comput., 34(2):737–762, March 2012.
math.iit.edu/~fass/StableGaussianRBFs.pdf.

[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.

[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.

[Gea71] C.W. Gear.
Numerical initial value problems in Ordinary Differential Equations.
Prentice Hall, 1971.

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

[Jäc05] P. Jäckel.
A note on multivariate Gauss-Hermite quadrature.
www.jaeckel.org/ANoteOnMultivariateGaussHermiteQuadrature.pdf, May 2005.

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

[LF03] E. Larsson and B. Fornberg.
A numerical study of some radial basis function based solution methods for elliptic PDEs.
Computers & Mathematics with Applications, 46(5):891–902, 2003.
amath.colorado.edu/faculty/fornberg/Docs/el_bf.pdf.

[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.

[Mic86] C.A. Micchelli.
Interpolation of scattered data: Distance matrices and conditionally positive definite functions.
Constructive Approximation, 2(1):11–22, 1986.

[Mil14] S. Milovanović.
Radial Basis Functions generated Finite Differences (RBF-FD) for Solving High-Dimensional PDEs in Finance.

Uppsala University, June 2014.
people.kth.se/~ejhall/cirkus/talks/milovanovic.pdf.

[Mon11] M. Mongillo.
Choosing basis functions and shape parameters for radial basis function methods.
SIAM Undergraduate Research Online, 4:190–209, 2011.
72.32.205.185/students/siuro/vol4/S01084.pdf.

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

[SDY03] C. Shu, H. Ding, and K.S. Yeo.
Local radial basis funcion-based differential quadrature method and its application to solve two-dimensional incompressible Navier-Stokes equations.
Computer Methods in Applied Mechanics and Engineering, 192:941–954, 02 2003.

[SL16] V. Shcherbakov and E. Larsson.
Radial basis function partition of unity methods for pricing vanilla basket options.
Computers & Mathematics with Applications, 71(1):185 – 200, 2016.
www.it.uu.se/research/publications/reports/2015-001/2015-001-nc.pdf.

[Smi86] G. Smith.
Numerical Solution of Partial Differential Equations: Finite Difference Methods.
Oxford University Press, January 1986.

[Tol00] A.I. Tolstykh.
On using RBF-based differencing formulas for unstructured and mixed structured–unstructured grid calculations.
In *Proc. 16th IMACS World Congress*, pages 4606—-4624, 2000.

[TS03] A.I. Tolstykh and D.A. Shirobokov.
On using radial basis functions in a "finite difference mode" with applications to elasticity problems.
Computational Mechanics, 33(1):68–79, 2003.

[Wan14] Xi Wang.
Finite differences based on radial basis functions to price options, 2014.
www.diva-portal.org/smash/get/diva2:787451/FULLTEXT01.pdf.

[WS02] Y.L. Wu and C. Shu.
Development of RBF-DQ method for derivative approximation and its application to simulate natural convection in concentric annuli.
Computational Mechanics, 29(6):477–485, 2002.