Periodic kernel-based high-dimensional approximation

Ian H. Sloan

UNSW Sydney, Australia

MCM 2023

Joint work with Vesa Kaarnioja and Frances Kuo

June 30, 2023

The story

- Parametric PDE, with many parameters $\Longrightarrow$ high dimensionality
- The standard approaches use multivariate polynomials
- But in high dimensions multivariate polnomials are very expensive, unless sparsity is used
- The talk presents a case an efficient kernel method, needing no sparsity, and allowing high dimensionality

The Parametric PDE Problem

Kaarnioja, Kazashi, Kuo, Nobile, IHS 2022 (KKKNS)
Kaarnioja, Kuo, IHS 2023 (KKS)

$$-\nabla \cdot (\tilde{a}(\mathbf{x}, \mathbf{z}) \nabla \tilde{u}(\mathbf{x}, \mathbf{z})) = q(\mathbf{x}) \quad \mathbf{x} \in D$$

$$\tilde{u}(\mathbf{x}, \mathbf{z}) = 0 \quad \text{on} \quad \partial D, \quad \mathbf{z} \in [-1, 1]^s,$$

with D a bounded Lipschitz domain in $\mathbb{R}^d$, $d = 2, 3$, and

$$\tilde{a}(\mathbf{x}, \mathbf{z}) = \bar{a} + \sum_{j=1}^{s} z_j \psi_j(\mathbf{x}) \geq a_{\min} > 0, \quad \mathbf{x} \in D, \quad \mathbf{z} \in [-1, 1]^s,$$

where $z_1, \ldots, z_s$ are parameters representing independent random variables distributed on $[-1, 1]$ with probability density $\rho(z)$.

Here we assume

$$\rho(z) = \tfrac{1}{\pi}(1 - z^2)^{-1/2} \quad \text{(Chebyshev of 1st kind)}$$

A periodic reformulation

Following Kaarnioja, Kuo, IHS (2020), we transform the field

$$\tilde{a}(\mathbf{x}, \mathbf{z}) = \bar{a} + \sum_{j=1}^{s} z_j \, \psi_j(\mathbf{x}), \quad \mathbf{x} \in D, \quad \mathbf{z} = (z_1, \ldots, z_s) \in [-1, 1]^s,$$

with density $\prod_j 1/(\pi\sqrt{1 - z_j^2})$ by sustituting $z_j = \sin(2\pi y_j)$, to get

$$a(\mathbf{x}, \mathbf{y}) := \tilde{a}(\mathbf{x}, \sin(2\pi\ \mathbf{y})) := \bar{a} + \sum_{j=1}^{s} \sin(2\pi y_j) \, \psi_j(\mathbf{x}), \quad \mathbf{x} \in D,$$

with $\mathbf{y} = (y_1, \ldots, y_s) \in [0, 1]^s$, and the y_j **iid uniformly distributed** on $[0, 1]$.

The probabilty model is unchanged by the substitution!

The space H_α

We take H_α to be an "unanchored" weighted Sobolev space of dominating mixed smoothness of order $\alpha \in \mathbb{N}$:

$$\|f\|^2_{H_\alpha} = \sum_{\mathbf{u} \subseteq \{1:s\}} \frac{1}{\gamma_{\mathbf{u}} (2\pi)^{2\alpha|\mathbf{u}|}} \int_{[0,1]^{|\mathbf{u}|}} \times \left| \int_{[0,1]^{s-|\mathbf{u}|}} \left(\prod_{j \in \mathbf{u}} \frac{\partial^\alpha}{\partial y_j^\alpha} \right) f(\mathbf{y}_{\mathbf{u}}, \mathbf{y}_{-\mathbf{u}}) \mathrm{d}\mathbf{y}_{-\mathbf{u}} \right|^2 \mathrm{d}\mathbf{y}_{\mathbf{u}},$$

- $\mathbf{y}_{\mathbf{u}}$ denotes the components of $\mathbf{y}$ that belong to the subset $\mathbf{u}$,
- $\mathbf{y}_{-\mathbf{u}}$ denotes the components that do not belong to $\mathbf{u}$.
- The $\gamma_{\mathbf{u}}$ are "weights" – one for each subset $\mathbf{u} \subseteq \{1:s\}$.

H_α is a RKHS with kernel K_α:

$$K_\alpha(\mathbf{y}, \mathbf{y}') = \sum_{\mathbf{u} \subseteq \{1:s\}} \gamma_{\mathbf{u}} \prod_{j \in \mathbf{u}} \eta_\alpha(y_j - y_j'),$$

where $\{1:s\} = \{1, 2, \ldots, s\}$, η_α is a known periodic function of a single variable, given by

$$\eta_\alpha(y) = \sum_{h \neq 0} \frac{e^{2\pi i h y}}{|h|^{2\alpha}},$$

H_α is a RKHS with kernel K_α:

$$K_\alpha(\mathbf{y}, \mathbf{y}') = \sum_{\mathbf{u} \subseteq \{1:s\}} \gamma_{\mathbf{u}} \prod_{j \in \mathbf{u}} \eta_\alpha(y_j - y_j'),$$

where $\{1:s\} = \{1, 2, \ldots, s\}$, η_α is a known periodic function of a single variable, given by

$$\eta_\alpha(y) = \sum_{h \neq 0} \frac{e^{2\pi i h y}}{|h|^{2\alpha}},$$

For α an integer, η_α is a piecewise polynomial:

$$\eta_\alpha(y) = \frac{(2\pi)^{2\alpha}}{(-1)^{\alpha-1}(2\alpha)!} B_{2\alpha}(\{y\})$$

with B_m a Bernouilli polynomial of even degree m,

$$B_2(y) = y^2 - y + \frac{1}{6}, \qquad B_4(y) = y^4 - 2y^3 + y^2 - \frac{1}{30}, \qquad \ldots$$

Given a real-valued function f defined on $[0,1]^s$, and points $\mathbf{t}_1, \mathbf{t}_2, \ldots, \mathbf{t}_n$, the **kernel interpolant** based on these points takes the form

$$f_n(\mathbf{y}) = \sum_{k=1}^{n} a_k K_\alpha(\mathbf{t}_k, \mathbf{y}), \qquad \mathbf{y} \in [0,1]^s,$$

where the coefficients are determined by

$$f_n(\mathbf{t}_k) = f(\mathbf{t}_k), \text{ for } k = 1, \cdots, n.$$

Given a real-valued function f defined on $[0,1]^s$, and points $\mathbf{t}_1, \mathbf{t}_2, \ldots, \mathbf{t}_n$, the **kernel interpolant** based on these points takes the form

$$f_n(\mathbf{y}) = \sum_{k=1}^{n} a_k K_\alpha(\mathbf{t}_k, \mathbf{y}), \qquad \mathbf{y} \in [0,1]^s,$$

where the coefficients are determined by

$$f_n(\mathbf{t}_k) = f(\mathbf{t}_k), \text{ for } k = 1, \cdots, n.$$

$$\Longrightarrow \sum_{k=1}^{n} K_\alpha(\mathbf{t}_k, \mathbf{t}_{k'})\, a_k = f\left(\mathbf{t}_{k'}\right), \qquad k' = 1, \ldots, n.$$

Good choice for $\{\mathbf{t}_1, \ldots, \mathbf{t}_s\}$

Advocated by Zeng, Leung, Hickernell 2004, and Zeng, Kritzer, Hickernell 2009.

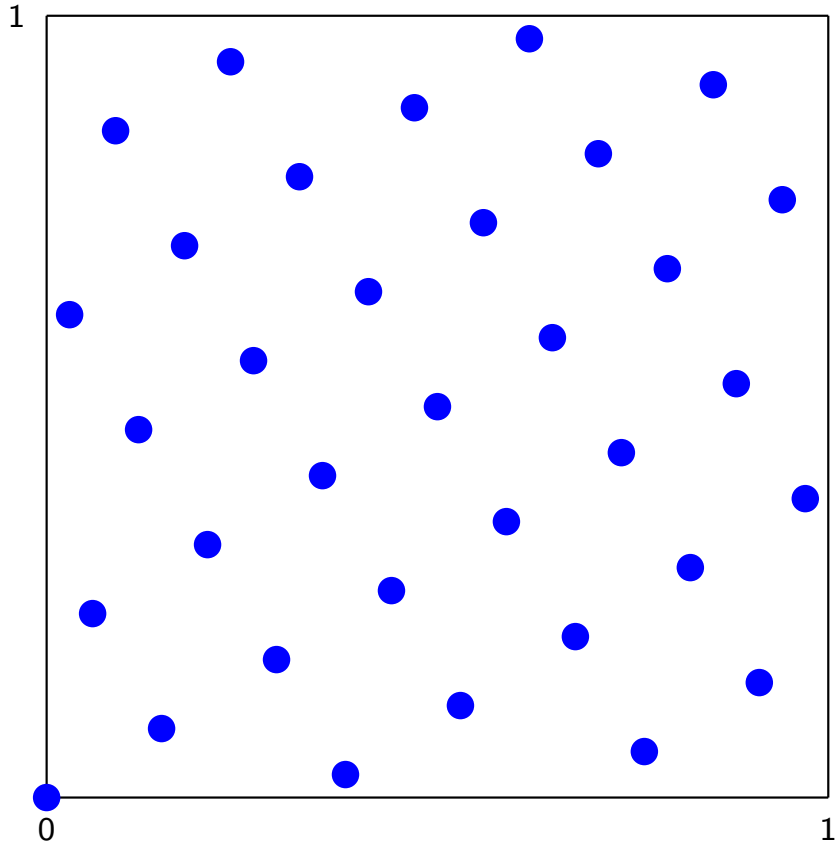
Take the interpolation points to be **lattice points**:

$$\mathbf{t}_k = \left\{ \frac{k\mathbf{z}^{\mathrm{gen}}}{n} \right\}, \quad k = 1, \ldots, n,$$

with $\mathbf{z}^{\mathrm{gen}} = (z_1, \ldots, z_s) \in \{1, 2, \ldots, n-1\}^s$.

Example of a (good) lattice

$$s = 2,\ n = 34,\ \boldsymbol{z}^{\mathrm{gen}} = (1, 21)$$



$$\boldsymbol{t}_k := \left\{ \left(\frac{k}{34}, \frac{21k}{34} \right) \right\}, \qquad k = 1, \ldots, 34$$

Lattice points give a major simplification:

The matrix elements become

$$K_\alpha(\mathbf{y}_k, \mathbf{y}_{k'}) = K_\alpha\left(\left\{\frac{(k-k')\mathbf{z}^{\mathrm{gen}}}{n}\right\}, 0\right).$$

Lattice points give a major simplification:

The matrix elements become

$$K_\alpha(\mathbf{y}_k, \mathbf{y}_{k'}) = K_\alpha\left(\left\{\frac{(k-k')\mathbf{z}^{\mathrm{gen}}}{n}\right\}, 0\right).$$

So the interpolation matrix is a **circulant** matrix, making it easy to invert by FFT – needs only $\boldsymbol{O(n \ln n)}$ flops.

With product weights the method is fast

Product weights take the form

$$\gamma_{\mathbf{u}} = \prod_{j \in \mathbf{u}} \gamma_j.$$

$$\implies K_\alpha(\mathbf{y}, \mathbf{y}') = \prod_{j=1}^{s} \left(1 + \gamma_j \eta(y_j - y_j')\right).$$

For product weights the method can compute $f_n(\mathbf{y})$ at m arbitrary points $\mathbf{y}$ in a time of order

$$O(\underbrace{Tn}_{f(\boldsymbol{t}_k)\forall k} + \underbrace{n \ln n}_{\text{lin. sys.}} + \underbrace{snm}_{\text{kernel evals.}}),$$

Here T is the time for a single function evaluation.

This is rather fast!

How to choose $\mathbf{z}^{\mathrm{gen}}$?

We choose $\mathbf{z}^{\mathrm{gen}}$ to give a small Worst-Case L_2 error for $f \in H_\alpha$.

The **worst-case L_2 error** for an algorithm $A : H \to L_2$ is

$$e(A, L_2, H) := \sup\{\|f - A(f)\|_{L_2} : f \in H, \quad \|f\|_H \leq 1\}.$$

For $A = A^*$. where $A^* f := f_n$, and $H = H_\alpha$, how to estimate the worst-case L_2 error?
We get computable upper bound on the worst-case error for kernel interpolation algorithm A^* (for an average choice of $\mathbf{z}^{\mathrm{gen}}$), by using the **o**ptimal property of reproducing kernel interpolation.

$\mathbf{z}^{\mathrm{gen}}$ is then chosen by a "component by component" (CBC) construction: fix $z_1 = 1$, and then in turn fix $z_2, \ldots, z_s$ by minimising the upper bound for dimensions $2, \ldots, s$ repsectively , keeping all earlier components fixed.

The L_2 error

Recall: The **worst-case L_2 error** for an algorithm $A : H_\alpha \to L_2$ is

$$e(A, L_2, H_\alpha) := \sup\{\|f - A(f)\|_{L_2} : f \in H_\alpha, \quad \|f\|_{H_\alpha} \leq 1\}.$$

Since $A^* f := f_n$ is linear this implies

$$\|f - A^*(f)\|_{L_2} \leq e(A^*, L_2, H_\alpha) \times \|f\|_{H_\alpha} \quad \text{for } f \in H_\alpha.$$

The KKKNS L_2 error bound is: for all $\lambda \in (1/2\alpha, 1]$,

$$\|f - A^*(f)\|_{L_2} \le \frac{\kappa}{n^{1/4\lambda}} \left(1 + \sum_{\emptyset \ne \mathbf{u} \subseteq \{1:s\}} |\mathbf{u}| \gamma_{\mathbf{u}}^{\lambda} [2\zeta(2\alpha\lambda)]^{|\mathbf{u}|} \right)^{1/2\lambda} \times \|f\|_{H_\alpha}.$$

- No matter how we choose the weights, the predicted ultimate convergence rate is arbitrarily close to $\alpha/2$. (Take λ arbitrarily close to $1/2\alpha$.)
- A convergence rate of $n^{-\alpha/2}$ for the worst-case L_2 error in H_α is known to be best possible, given that we use lattice points (Byrenheid, Kämmerer, Ulrich and Volkmer, 2017).
- (It is suboptimal if we are free to choose any points – the optimal rate for L_2 approximation in H_α is $n^{-\alpha}$.)

Parametric PDE example

V Kaarnioja, Y Kazashi, F Kuo, F Nobile, IHS, Numer. Math. '22

$$-\nabla\Big(a(\mathbf{x},\mathbf{y})\nabla u(\mathbf{x},\mathbf{y})\Big) = x_2, \quad \mathbf{x}\in D,\ \mathbf{y}\in U_s := [0,1]^s,$$
$$u(\mathbf{x},\mathbf{y}) = 0, \quad \mathbf{x}\in \partial D,\ \mathbf{y}\in U_s,$$

in the domain $D = (0,1)^2$, with

$$a(\mathbf{x},\mathbf{y}) := 1 + \sum_{j=1}^{s} \sin(2\pi y_j)\psi_j(\mathbf{x}), \quad \mathbf{x}\in D,\ \mathbf{y}\in U_s,$$

where

$$\psi_j(\mathbf{x}) = c\, j^{-\theta} \sin\left(j\pi x_1\right) \sin\left(j\pi x_2\right),$$

with $\theta > 1$ and $c > 0$ variable,

and with $s = 10$ or 100 or .....

The KKKNS error bound:

For all $\lambda \in (\frac{1}{2\alpha}, 1]$,

$$\sqrt{\int_D \int_{[0,1]^s} |u(\mathbf{x},\mathbf{y}) - u_n(\mathbf{x},\mathbf{y})|^2 \mathrm{d}\mathbf{y}\mathrm{d}\mathbf{x}} \leq \kappa \|q\|_{H^{-1}(D)} n^{1/4\lambda} C_s(\lambda)$$

$$C_s(\lambda) := \left(1 + \sum_{\emptyset \neq \mathbf{u} \subseteq \{1:s\}} |\mathbf{u}| \gamma_{\mathbf{u}}{}^{\lambda} [2\zeta(2\alpha\lambda)]^{|\mathbf{u}|}\right)^{1/2\lambda}$$
$$\times \left(\sum_{\mathbf{u} \subseteq \{1:s\}} \frac{1}{\gamma_{\mathbf{u}}} \left(\sum_{\mathbf{m}_{\mathbf{u}} \in \{1:\alpha\}^{|\mathbf{u}|}} |\mathbf{m}_{\mathbf{u}}|! \prod_{j \in \mathbf{u}} (b_j^{m_j} S(\alpha, m_j))\right)^2\right)^{1/2}.$$

Note that both factors depend on the weights $\gamma_{\mathbf{u}}$.

The KKKNS weights

For $\mathbf{u} \neq \emptyset$, and $\alpha = 1$ or 2,

$$\gamma_{\mathbf{u}} := \sum_{\mathbf{m}_{\mathbf{u}} \in \{1:\alpha\}^{|\mathbf{u}|}} (|\mathbf{m}_{\mathbf{u}}|!)^{\frac{2}{1+\lambda}} \prod_{j \in \mathbf{u}} \left(\frac{(c\, j^{-\theta}/a_{\min}))^{m_j}}{\sqrt{2e^{1/e}\zeta(2\alpha\lambda)}} \right)^{\frac{2}{1+\lambda}}.$$

The KKKNS weights

For $\mathbf{u} \neq \emptyset$, and $\alpha = 1$ or 2,

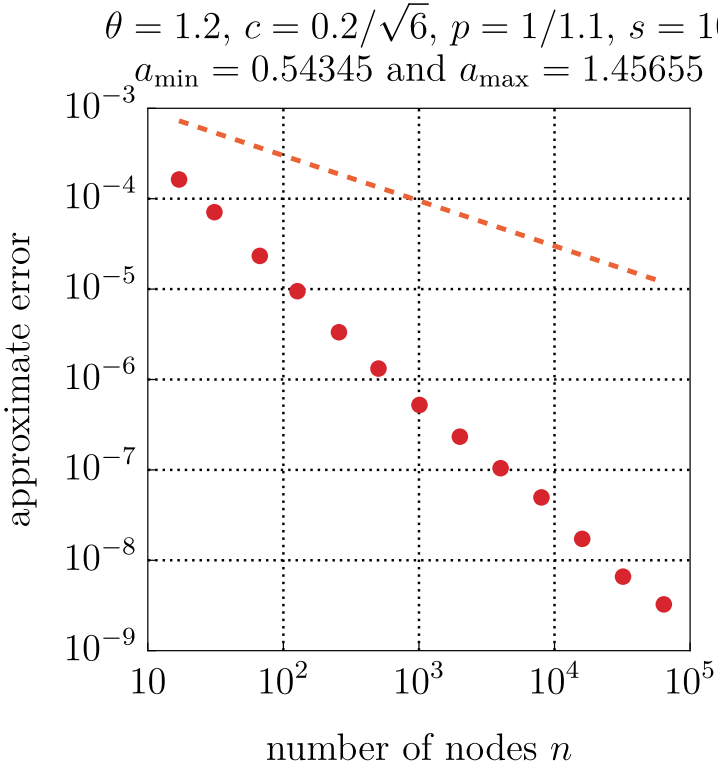
$$\gamma_{\mathbf{u}} := \sum_{\mathbf{m_u} \in \{1:\alpha\}^{|\mathbf{u}|}} (|\mathbf{m_u}|!)^{\frac{2}{1+\lambda}} \prod_{j \in \mathbf{u}} \left(\frac{(c\, j^{-\theta}/a_{\min}))^{m_j}}{\sqrt{2e^{1/e}\zeta(2\alpha\lambda)}} \right)^{\frac{2}{1+\lambda}}.$$

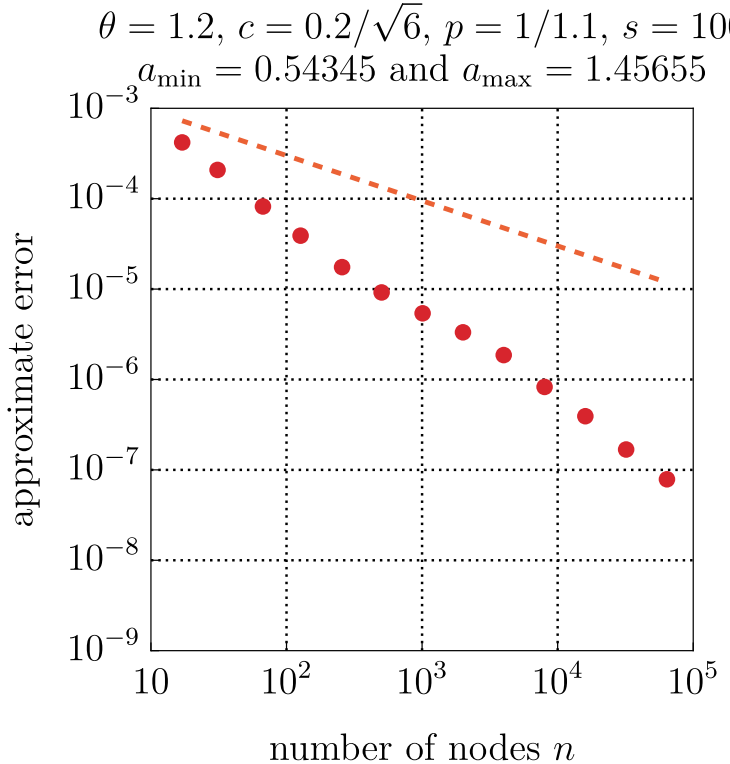
For these weights the error bound was shown in KKKNS to be

$$\boxed{\|u - u_n\|_{L_2(D\times U)} \leq C_\delta n^{-\alpha/2+\delta}},$$

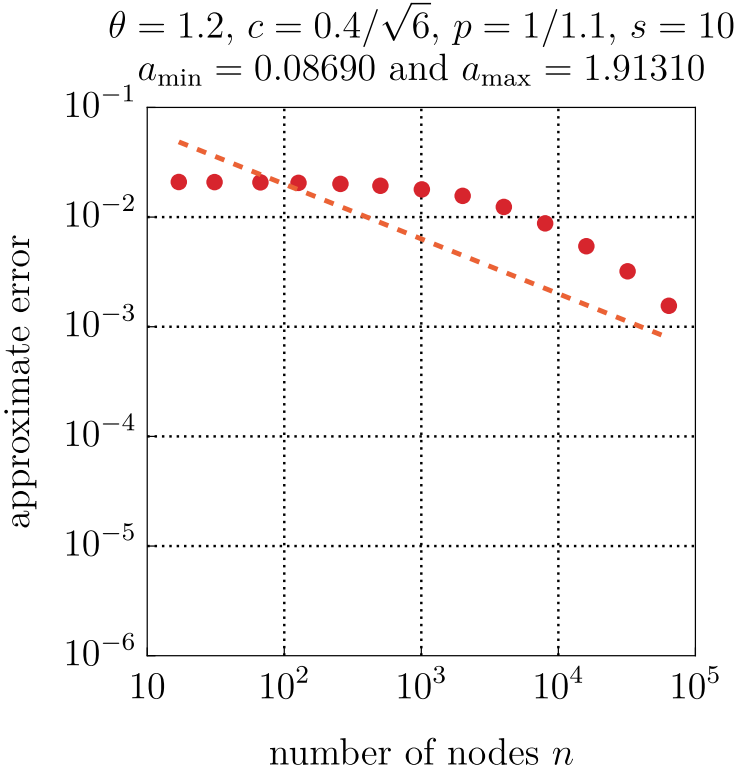
where C_δ (which is known explicitly!) is independent of s.

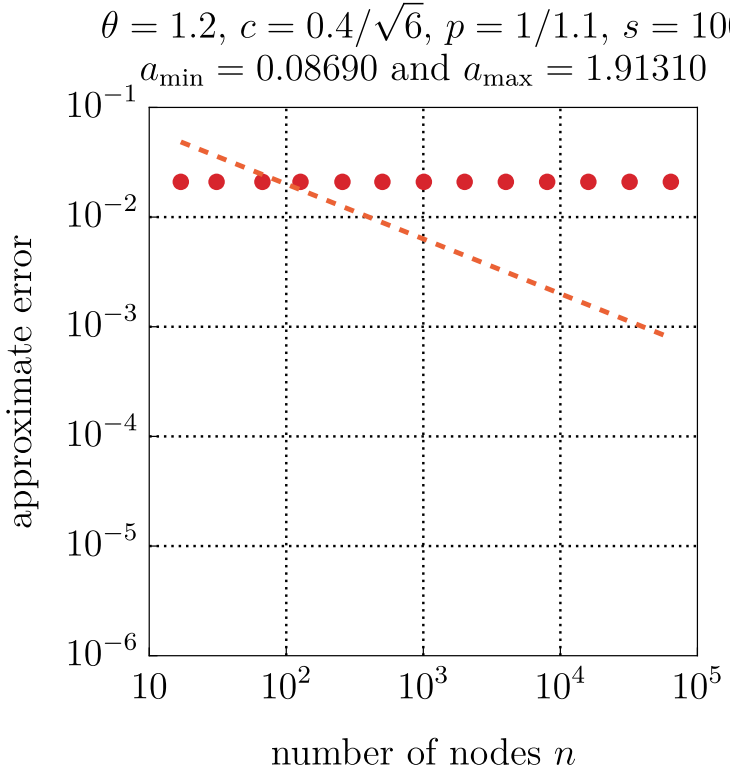
Numerical results for $\theta = 1.2, c = 0.2/\sqrt{6}, \alpha = 1$





Numerical results for $\theta = 1.2, c = 0.4/\sqrt{6}, \alpha = 1$





The weights used above were not product weights!

They were more complicated ("SPOD") weights:

$$\gamma_{\mathbf{u}} := \sum_{\mathbf{m}_{\mathbf{u}} \in \{1:\alpha\}^{|\mathbf{u}|}} (|\mathbf{m}_{\mathbf{u}}|!)^{\frac{2}{1+\lambda}} \prod_{j \in \mathbf{u}} \left(\frac{(c\, j^{-\theta}/a_{\min}))^{m_j}}{\sqrt{2e^{1/e}\zeta(2\alpha\lambda)}} \right)^{\frac{2}{1+\lambda}}.$$

The weights used above were not product weights!

They were more complicated ("SPOD") weights:

$$\gamma_{\mathbf{u}} := \sum_{\mathbf{m}_{\mathbf{u}} \in \{1:\alpha\}^{|\mathbf{u}|}} (|\mathbf{m}_{\mathbf{u}}|!)^{\frac{2}{1+\lambda}} \prod_{j \in \mathbf{u}} \left(\frac{(c\, j^{-\theta}/a_{\min}))^{m_j}}{\sqrt{2e^{1/e}\zeta(2\alpha\lambda)}} \right)^{\frac{2}{1+\lambda}}.$$

For these weights the cost (Cools, Kuo, Nuyens, IHS 2019(2)) is

$$O(Tn + n \ln n + s^2 nm);$$

and the cost for computing $\mathbf{z}$ is $O(sn \ln n + s^3 n)$.

Recall: product weights

$$O(\underbrace{Tn}_{f(t_k)\forall k} + \underbrace{n \ln n}_{\text{lin. sys.}} + \underbrace{snm}_{\text{kernel evals.}}),$$

and the cost for computing $\mathbf{z}$ is $O(sn \ln n)$.

Can we get as good results with product weights?

YES WE CAN (and even better)! We use **serendipitous weights**.
V Kaarnioja, F Kuo, IHS, in progress

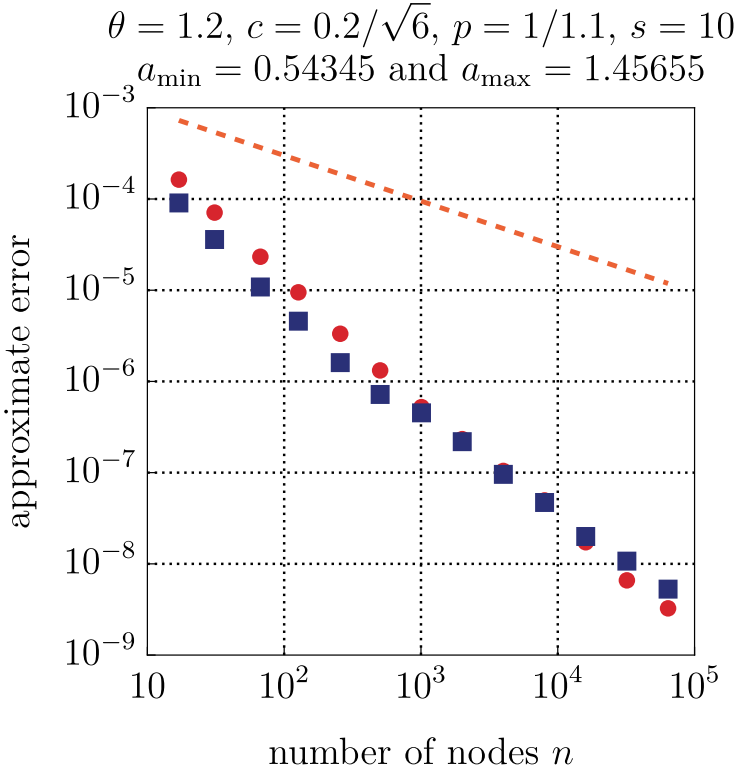
["Serendipity" – happy discovery by accident] Because the SPOD weights are much too large, we tried dropping the factorials:

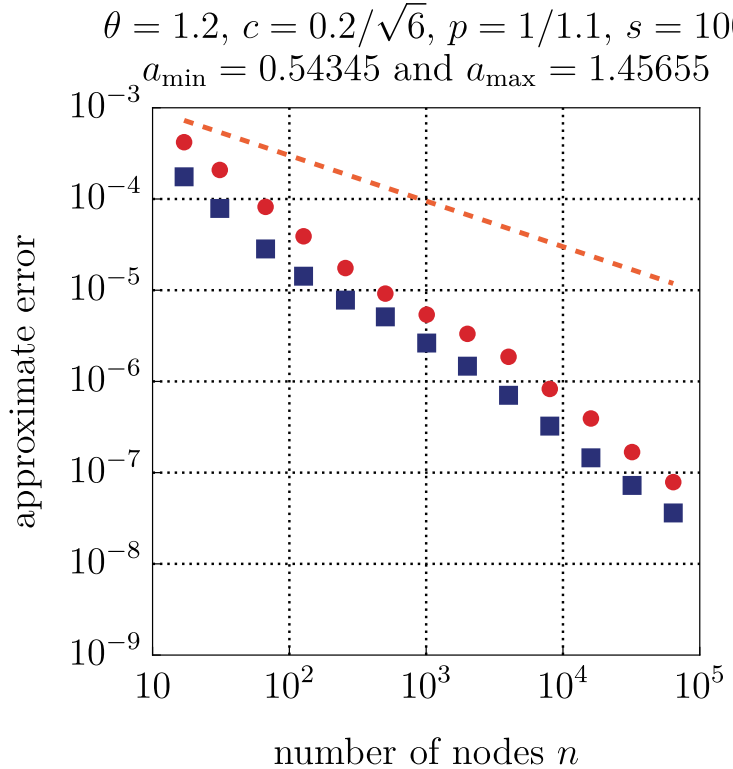
$$\text{Recall } \gamma_{\mathbf{u}}^{\mathrm{KKKNS}} := \sum_{\mathbf{m}_{\mathbf{u}} \in \{1:\alpha\}^{|\mathbf{u}|}} \underbrace{(|\mathbf{m}_{\mathbf{u}}|!)^{\frac{2}{1+\lambda}}}_{\text{OMIT!}} \prod_{j \in \mathbf{u}} \left(\frac{(c\, j^{-\theta}/a_{\min})^{m_j}}{\sqrt{2e^{1/e}\zeta(2\alpha\lambda)}} \right)^{\frac{2}{1+\lambda}}.$$

$$\implies \gamma_{\mathbf{u}}^{\mathrm{ser}} = \prod_{j \in \mathbf{u}} \sum_{m_j=1}^{\alpha} \left(\frac{(c\, j^{-\theta}/a_{\min})^{m_j}}{\sqrt{2e^{1/e}\zeta(2\alpha\lambda)}} \right)^{\frac{2}{1+\lambda}},$$

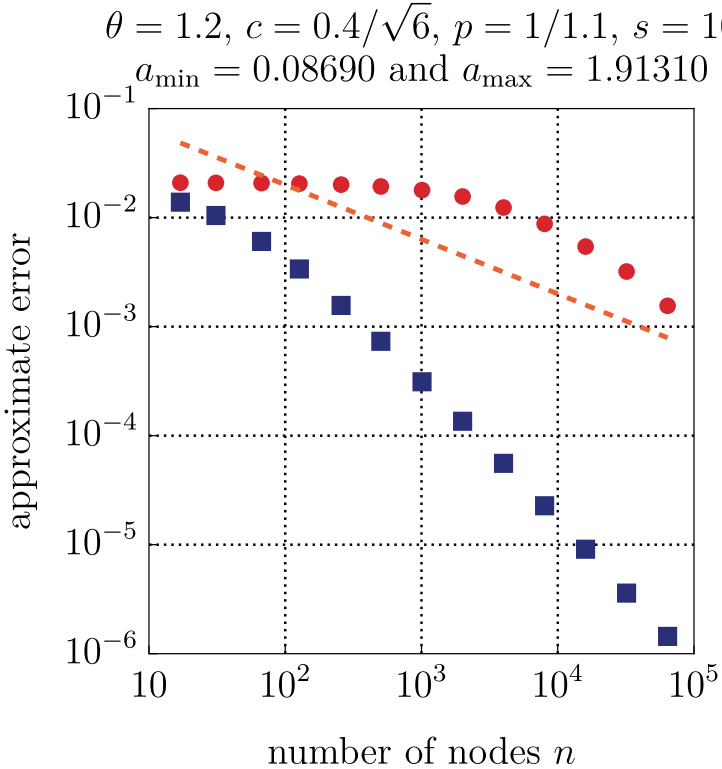
leading to the same error bound, with a dufferent constant, no longer independent of s.

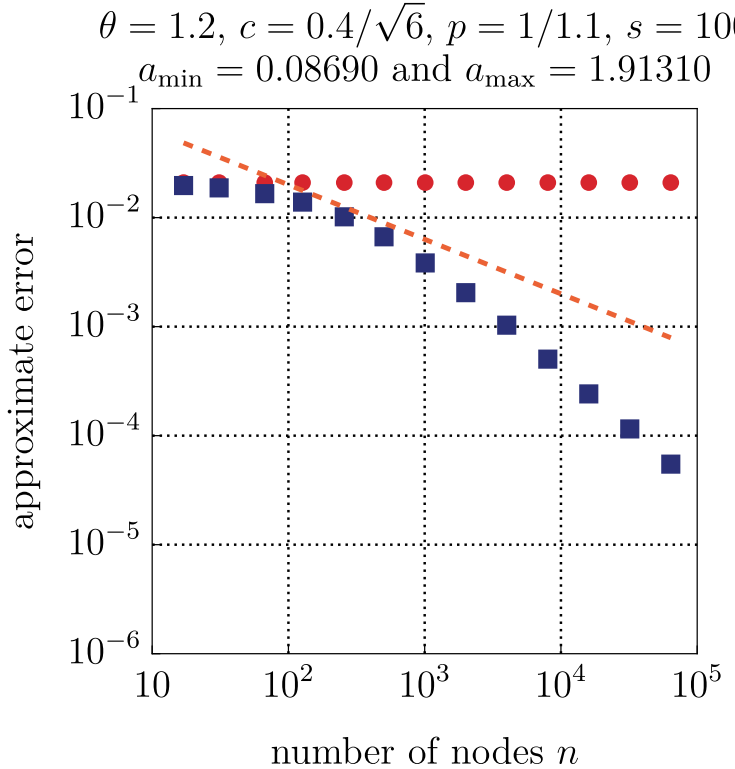
Numerical results for $\theta = 1.2, c = 0.2/\sqrt{6}, \alpha = 1$





Numerical results for $\theta = 1.2, c = 0.4/\sqrt{6}, \alpha = 1$

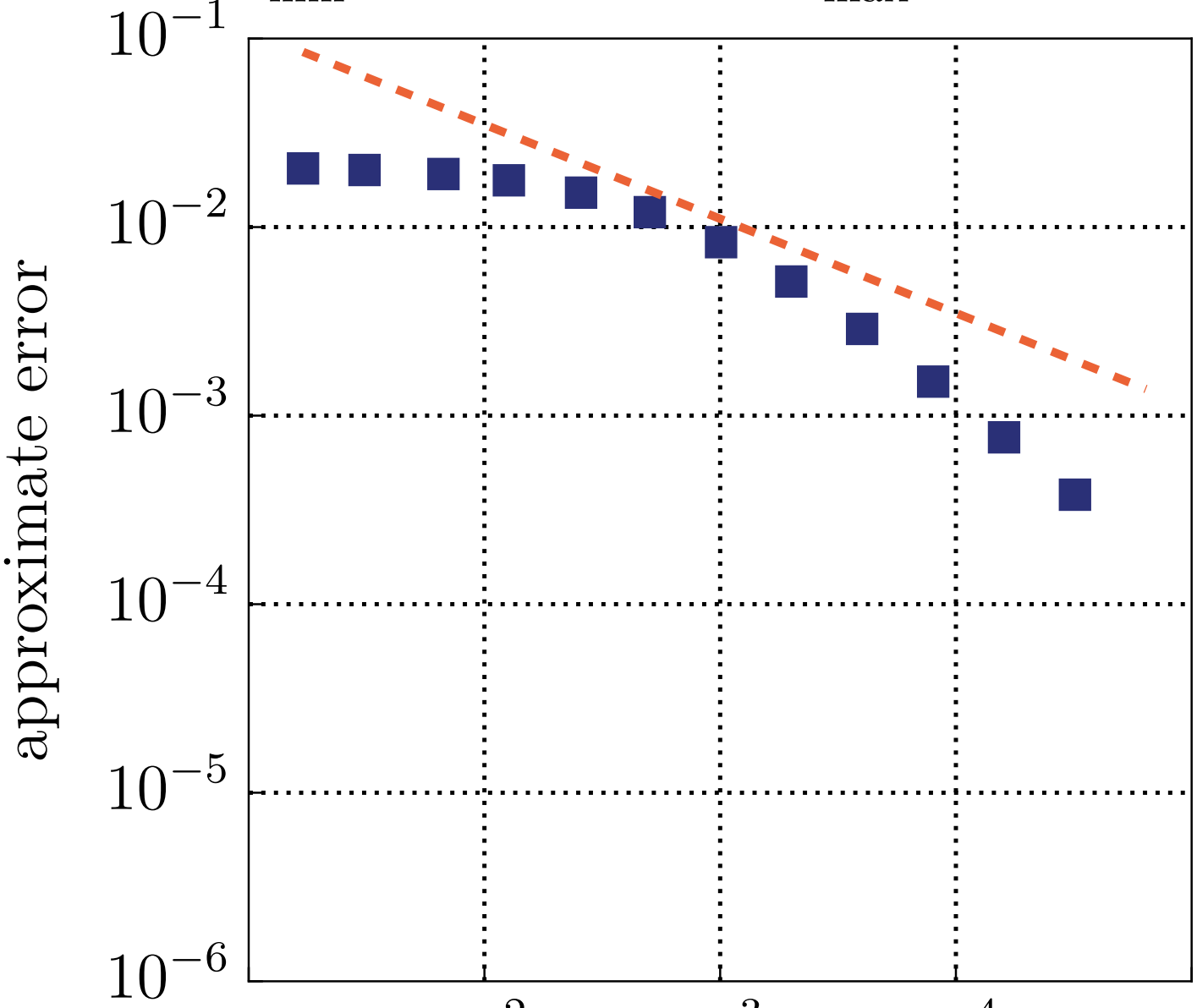




That hardest problem, but with $s = 1000$

$$\theta = 1.2,\ c = 0.4/\sqrt{6},\ p = 1/1.1,\ s = 1000$$

$$a_{\min} = 0.08690 \text{ and } a_{\max} = 1.91310$$



Summary

High-dimensional **approximation** in a realistic application

- Approximation can be both mathematically rigorous and feasible for large dimensionality and large number of points.
- The cost is merely linear in s IF we use periodic random variables and serendipitous weights.
- The probability model is the same as for GPC with Chebyshev of 1st kind as weight function.
- Serendipitous weights are seen to give excellent results even for **hard** high-dimensional problems.

The rate is often much better than predicted!

In all these cases (all with $\alpha = 1$) the rate is much bettter than predicted above. Why? Because:

The solution of the parameteric PDE is really smooth, and in particular lies in $H_{2\alpha}$.

Kaarnioja, Kuo and Sloan 2023 show that in this case the asymptotic L_2 error convergence rate is doubled, to $n^{-\alpha}$.