Statistical Numerical Approximation

(Glorified linear interpolation)

Houman Owhadi

Oberwolfach March 14, 2019

DARPA EQUiPS / AFOSR award no FA9550-16-1-0054
Computational Information Games, 2015-2018

AFOSR. Grant number FA9550-18-1-0271.
Games for Computation and Learning, 2018-2021.

Collaborators

Clint Scovel

Tim Sullivan

Lei Zhang

- Kernel Flows: from learning kernels from data into the abyss. H. Owhadi and G. R. Yoo, arXiv:1808.04475, 2018.
- Operator adapted wavelets, fast solvers, and numerical homogenization from a game theoretic approach to numerical approximation and algorithm design, H. Owhadi and C. Scovel, Cambridge University Press, 2019.
- Universal Scalable Robust Solvers from Computational Information Games and fast eigenspace adapted Multiresolution Analysis, 2017. arXiv:1703.10761. H. Owhadi and C. Scovel.
- Compression, inversion, and approximate PCA of dense kernel matrices at near-linear computational complexity, arXiv:1706.02205, 2017. Schäfer, Sullivan, Owhadi.
- De-noising by thresholding operator adapted wavelets. G. R. Yoo and H. Owhadi, 2018, Statistics and Computing, [arXiv:1805.10736]
- Gamblets for opening the complexity-bottleneck of implicit schemes for hyperbolic and parabolic ODEs/PDEs with rough coefficients, H. Owhadi and L. Zhang, Journal of Computational Physics, Volume 347, pages 99-128, 2017. arXiv:1606.07686.
- Multigrid with rough coefficients and Multiresolution operator decomposition from Hierarchical Information Games. H. Owhadi. SIAM Review, 59(1), 99149, 2017. arXiv:1503.03467
- Bayesian Numerical Homogenization. H. Owhadi. SIAM Multiscale Modeling & Simulation, 13(3), 812828, 2015. arXiv:1406.6668

Florian Schäfer

Yoo Gene Ryan

Connections between numerical approximation and Gaussian process regression

Pioneering work


Poincaré (1896). Sul'din (1959). Sard (1963).
Kimeldorf and Wahba (1970). Larkin (1972)

Bayesian Numerical Analysis

Diaconis (1988). Shaw (1988).
O'Hagan (1991). Skilling (1992).

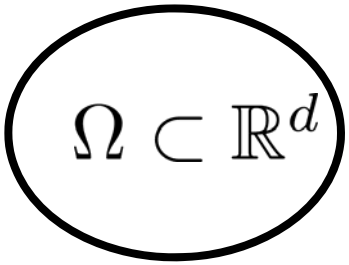
Information based complexity

Woźniakowski (1986). Wasilkowski and Woźniakowski (1986).
Packel (1987). Traub, Wasilkowski and Woźniakowski (1988).
Novak and Woźniakowski (2008-2010).

Probabilistic Numerics

Briol, Chkrebtii, Campbell, Calderhead, **Cockayne**, Conrad, Duvenaud, Girolami, Griebel, Hennig, Karniadakis, Raissi, Oates, Osborne, O., Paris, Sejdinovic, Särkä, Schäfer, Schober, Scovel, Sullivan, Stuart, Venturi, Zabaras, Zhang, Zygalakis... (2014-now)

The operator



$\mathcal{L}$: linear, symmetric, positive, invertible, local

$$(H_0^s(\Omega), \|\cdot\|_{H_0^s(\Omega)}) \xrightarrow{\mathcal{L}} (H^{-s}(\Omega), \|\cdot\|_{H^{-s}(\Omega)})$$

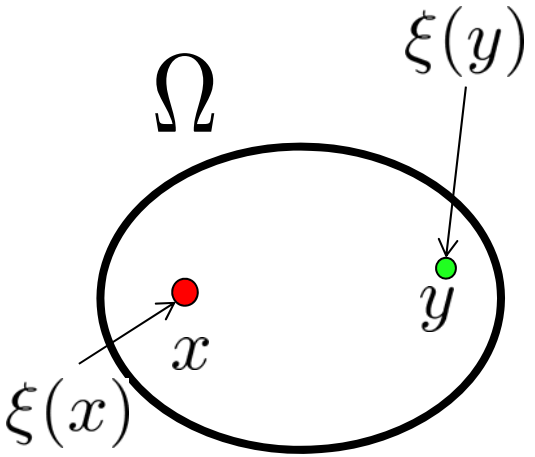
$\mathcal{L}$ local: $\int_\Omega u\mathcal{L}v = 0$ if u and v have disjoint supports

G : Green's function

$$(1)\quad \boxed{\mathcal{L}u = f}$$

The solution of (1) is $u(x) = \int_\Omega G(x,y)f(y)\,dy$

The Gaussian field



G : symmetric positive definite kernel

$$\xi \sim \mathcal{N}(0, G)$$

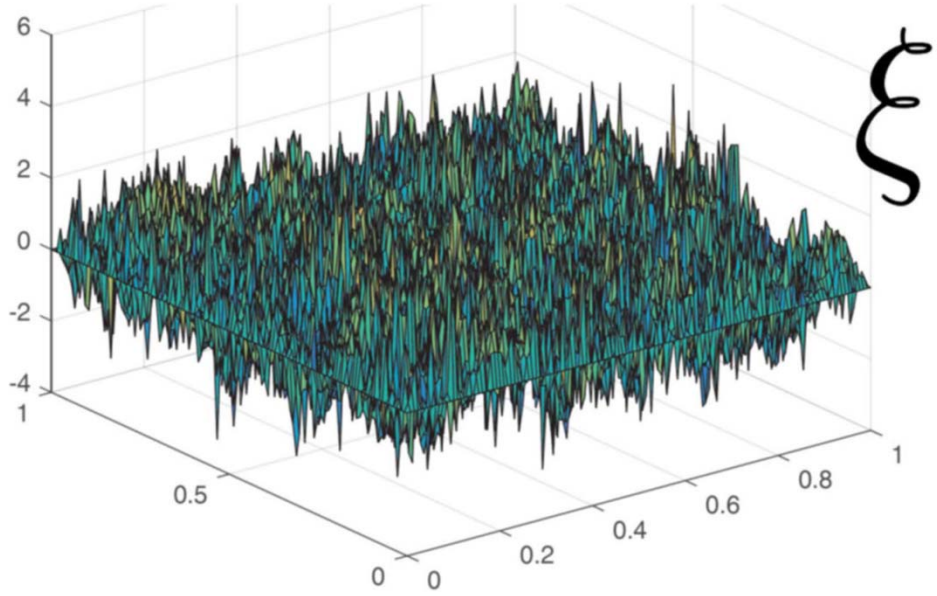
$$s > d/2$$

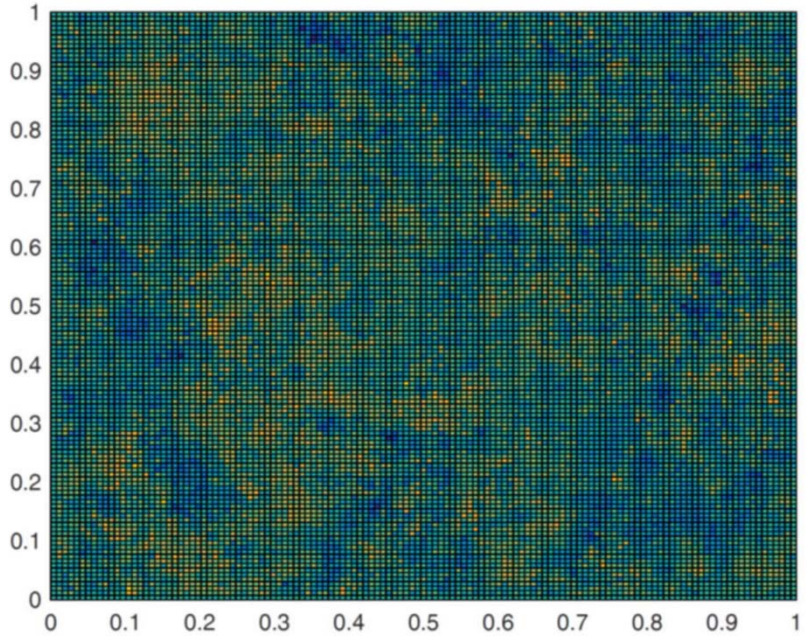
$\xi(x)$ is a centered Gaussian process

$$\boxed{\mathrm{Cov}\left(\xi(x), \xi(y)\right) = G(x, y)}$$

For $\phi \in H^{-s}(\Omega)$,

$$\boxed{\int_\Omega \xi(x)\phi(x)\,dx \sim \mathcal{N}\left(0, \int_{\Omega^2} \phi(x)G(x,y)\phi(y)\,dx\,dy\right)}$$



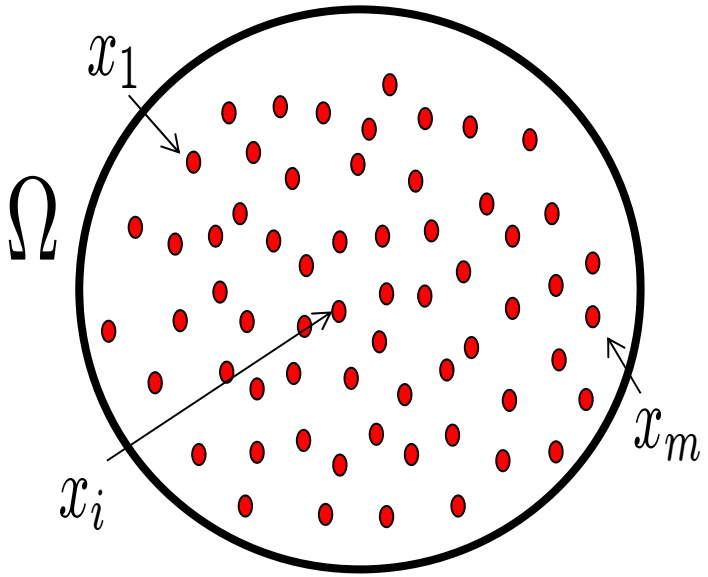


Problem $s > d/2$

$u \in H_0^s(\Omega)$ unknown

Given $u(x_1), \ldots, u(x_m)$
what is your best approximation of u?
Best: $\min_v \max_u \frac{\|u-v\|^2}{\|u\|^2}$ as small as possible

$$\|u\|^2 = \int_\Omega u \mathcal{L} u$$



Answer

$$v^\dagger(x) = \mathbb{E}\left[\xi(x) \middle| \xi(x_i) = u(x_i) \text{ for all } i\right]$$

Numerical approximation/Optimal recovery game

$\|\cdot\|$: Operator/Energy norm defined by $\mathcal{L}$

$\|u\|^2 = \int_\Omega u\mathcal{L}u = [\mathcal{L}u, u]$

$\phi_1, \ldots, \phi_m \in H^{-s}(\Omega)$

$[\phi, u] := \int_\Omega \phi u$

Player I

Chooses $u \in H_0^s(\Omega)$

Player II

Sees $\big([\phi_1, u], \ldots, [\phi_m, u]\big)$

Chooses $v\big([\phi_1, u], \ldots, [\phi_m, u]\big) \in H_0^s(\Omega)$

$\mathcal{M}ax$

$\mathcal{M}in$

$$\frac{\Big\|u - v\big([\phi_1,u],\ldots,[\phi_m,u]\big)\Big\|^2}{\|u\|^2}$$

Theorem

The optimal strategy of Player II is

$$v^\dagger = \mathbb{E}\Big[\xi \Big| [\phi_i, \xi] = [\phi_i, u] \text{ for } i \in \{1, \ldots, m\}\Big]$$

$v^\dagger$ is also the minimizer of

$$\inf_v \sup_{u \in H_0^s(\Omega)} \frac{\Big\| u - v([\phi_1, u], \ldots, [\phi_m, u]) \Big\|^2}{\|u\|^2}$$

C. A. Micchelli. Orthogonal projections are optimal algorithms. *Journal of Approximation Theory*, 40(2):101–110, 1984.

C. A. Micchelli and T. J. Rivlin. A survey of optimal recovery. In *Optimal Estimation in Approximation Theory*, pages 1–54. Springer, 1977.

Representation theorem

$$v^\dagger(x) = \sum_{i=1}^{m} [\phi_i, u]\psi_i(x)$$

Optimal recovery splines

$$\psi_i = \sum_{j=1}^{m} \Theta_{i,j}^{-1} \mathcal{L}^{-1}\phi_j \qquad \Theta_{i,j} := \int \phi_i \mathcal{L}^{-1}\phi_j$$

Elementary gambles/bets

$$\psi_i = \mathbb{E}\Big[\xi \Big| [\phi_j, \xi] = \delta_{i,j} \text{ for } j \in \{1, \ldots, m\}\Big]$$

Numerical Homogenization

$$(1)\quad \boxed{\mathcal{L}u = f}$$

Given m, to find $\psi_1, \ldots, \psi_m$ s.t:

1. *Accuracy.*

   $$\sup_{f \in L^2(\Omega)} \inf_{c \in \mathbb{R}^m} \frac{\|\mathcal{L}^{-1} f - \sum_{i=1}^m c_i \psi_i\|}{\|f\|_{L^2(\Omega)}}$$

   is as small as possible.

2. *Localization.* The ψ_i are as localized as possible.

Numerical Homogenization

Classical homogenization

Papanicolaou, Bensoussan, Lions, Murat, Tartar, Varadhan, Zhikov, Kozlov, Oleinik, Allaire, Nguetseng,... (and many others)

MsFEM Hou, Wu, Efendiev, Fish, Wagiman, Chung...

Harmonic coordinates Babuska, Caloz, Osborn, Allaire, Brizzi, Zhang, O. ...

HMM Engquist, E, Abdulle, Runborg, Schwab,...

Stochastic Homogenization

Papanicolaou, Varadhan, Zhikov, O., Bourgeat, Piatnitsky, Lebris, Legoll, Blanc, Jing, Bal, Sougadinis, E,...
Gloria, Otto (quantitative CLT estimates)

Projection based methods

Nolen, Papanicolaou, Pironneau.

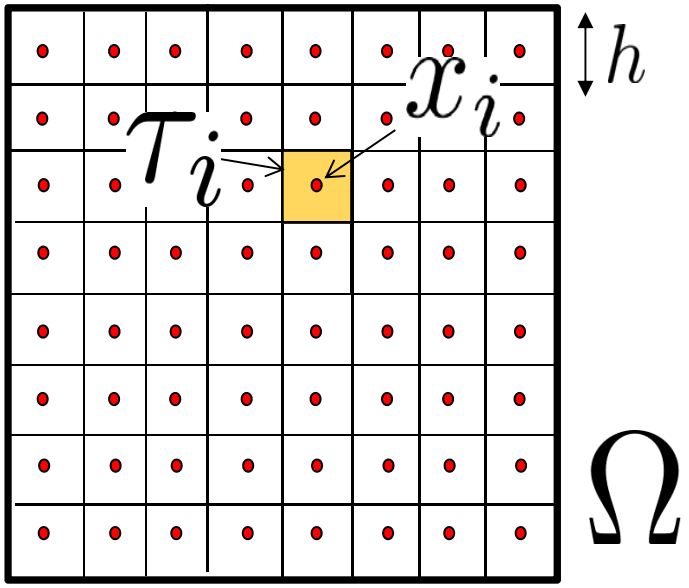
Variational Multiscale Method. Orthogonal Decomposition

Hughes, Feijóo, Mazzei, Quincy, Malqvist, Peterseim,...

Flux Norm. Rough Polyharmonic splines

O., Berlyand, Zhang, Symes, Bebendorf,...

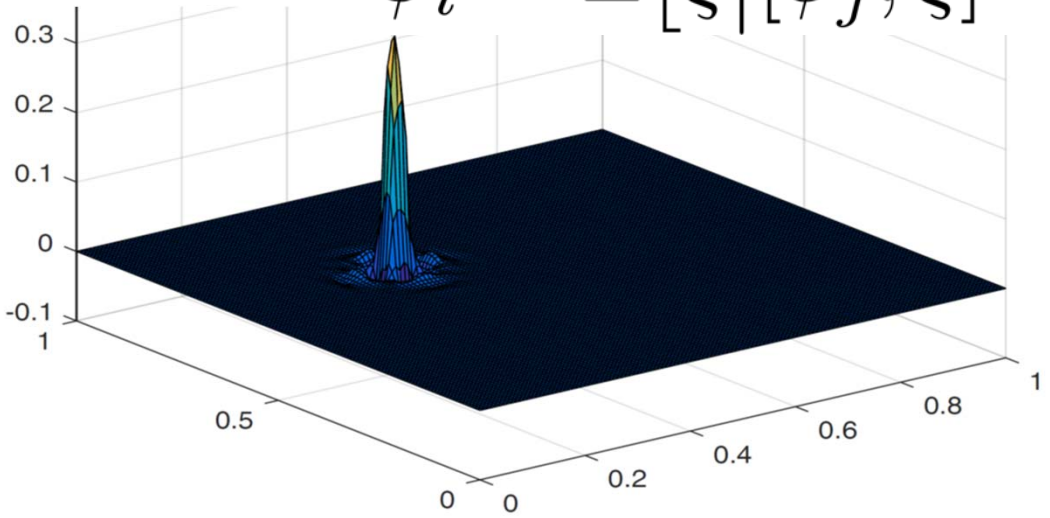
Measurement functions



- $\phi_i = \dfrac{1_{\tau_i}}{\sqrt{|\tau_i|}}.$
- $\phi_i = \delta(\cdot - x_i),$ $(s > \dfrac{d}{2})$

$$\psi_i = \mathbb{E}\left[\xi \middle| [\phi_j, \xi] = \delta_{i,j} \ \forall j\right]$$

Gamblets



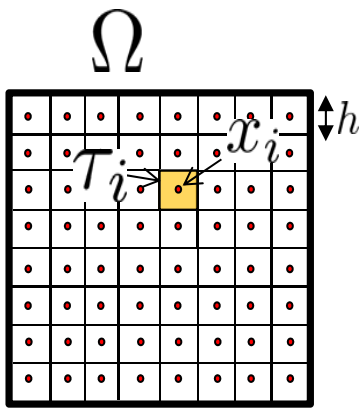
Accuracy

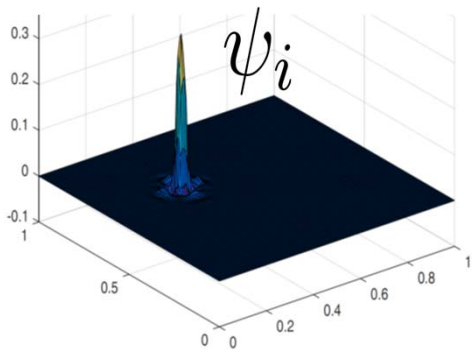
$$\inf_{v \in \mathrm{span}\{\psi_1,\ldots,\psi_m\}} \|u - v\| \leq C h^s \|\mathcal{L}u\|_{L^2(\Omega)}$$

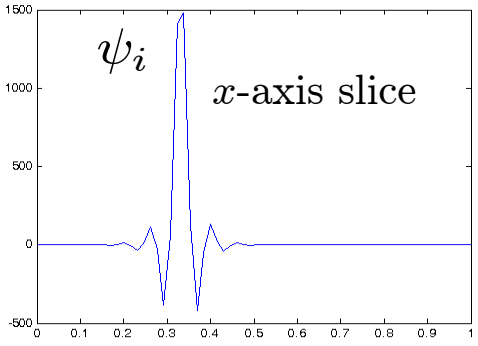
Achieves Kolmogorov n-width accuracy
up to multiplicative constant

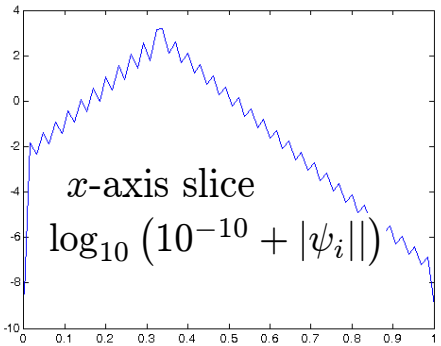
Localization

$$\|\psi_i\|_{H^s(\Omega/B(x_i,nh))} \leq C e^{-n/C}$$









Localization problem

Localization problem in Numerical Homogenization

[Chu-Graham-Hou-2010] (limited inclusions).
[Efendiev-Galvis-Wu-2010] (limited inclusions or mask).
[Babuska-Lipton 2010] (local boundary eigenvectors).
[Owhadi-Zhang 2011] (localized transfer property) based on Green's function estimates of [Gloria, Neukamm, Otto, 2015] (quantification of ergodicity).
[Owhadi-Zhang-Berlyand 2013] (Rough Polyharmonic Splines).

Local Orthogonal Decomposition - Subspace iteration

[Malqvist-Peterseim 2012] Local Orthogonal Decomposition.
[Kornhuber, Peterseim, Yserentant, 2016] Subspace correction.

Non-conforming measurements, higher order PDEs.

[O. 2015]. [O. - Scovel, 2017], [Hou-Zhang, 2017]

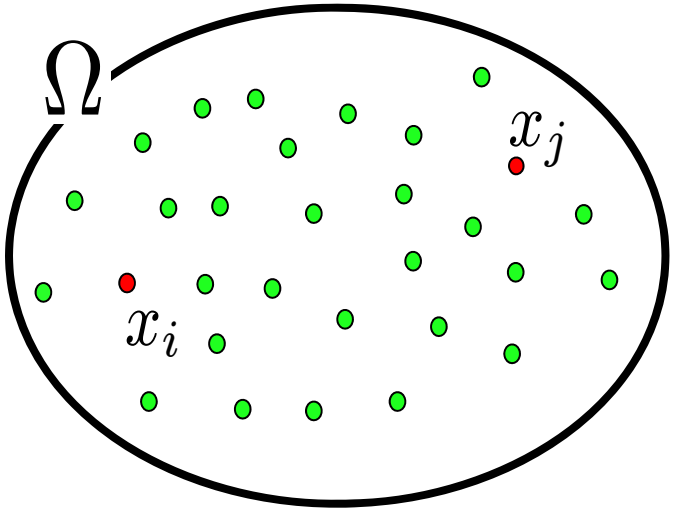
Wannier basis functions

[Wannier 1962] [Kohn 1959] [Marzari, Vanderbilt, 1997]

Screening effect

"The screening effect is the geostatistical term for the phenomenon of nearby observations tending to reduce the influence of more distant observations when using kriging for spatial interpolation." (Stein, 2011)

[Stein 2002]: asymptotic results.
[Stein 2011]: formulating a general result is hard.



$$\mathrm{Cor}\left(\xi(x_i), \xi(x_j) \middle| \xi(x_l), l \neq i, j\right) = -\frac{\langle \psi_i, \psi_j \rangle}{\|\psi_i\| \|\psi_j\|}$$

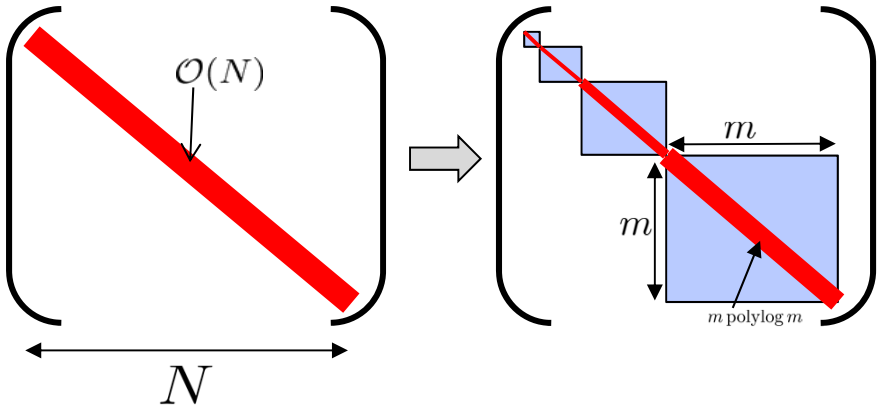
$$\mathrm{Cor}\left(\xi(x_i), \xi(x_j) \middle| \xi(x_l), l \neq i, j\right) \leq C e^{-\mathrm{dist}(x_i, x_j)/h}$$

Operator adapted wavelets

$$(H_0^s(\Omega), \|\cdot\|_{H_0^s(\Omega)}) \xrightarrow{\mathcal{L}} (H^{-s}(\Omega), \|\cdot\|_{H^{-s}(\Omega)})$$

How to construct operator adapted wavelets for $\mathcal{L}$?

1. Scale-orthogonal wavelets with respect to operator scalar product (leads to block-diagonalization)
2. Operator needs to be well conditioned within each subband
3. Wavelets need to be localized (compact support or exp. decay)



Operator adapted wavelets

First Generation Wavelets: Signal and imaging processing

Morlet, Grossmann, Mallat, Daubechies, Coifman, Meyer, Wickerhauser,...

First Generation Operator Adapted Wavelets (shift and scale invariant)

Cohen, Daubechies, Feauveau (Biorthogonal bases of compactly supported wavelets), Beylkin, Coifman, Rokhlin, Engquist, Osher, Zhong, Alpert, Jawerth, Sweldens, Dahlke, Weinreich, Bacry, Mallat, Papanicolaou, Bertoluzza, Maday, Ravel, Vasilyev, Paolucci, Dahmen, Kunoth, Stevenson, Candes...

Lazy wavelets (Multiresolution decomposition of solution space)

Yserentant (Multilevel splitting), Bank, Dupont, Yserentant (Hierarchical basis multigrid method),...

Second Generation Operator Adapted Wavelets

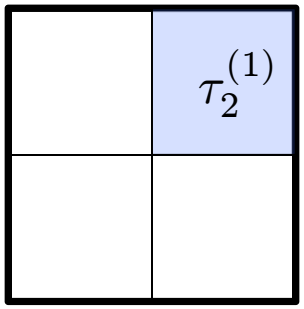
Sweldens (the lifting scheme); Dorobantu, Engquist; Vassilevski, Wang (stabilizing the hierarchical basis); Carnicer, Dahmen, Peña, Lounsbery, DeRose, Warren, Barinka, Barsch, Charton, Cohen, Dahlke, Dahmen, Urban, Cohen, Dahmen, DeVore, Chiavassa, Liandrat, Dahmen, Kunoth, Schwab, Stevenson, Sudarshan, Engquist, Runborg, Yin, Liandrat,...

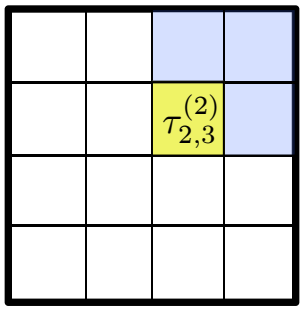
Hierarchy of measurement functions

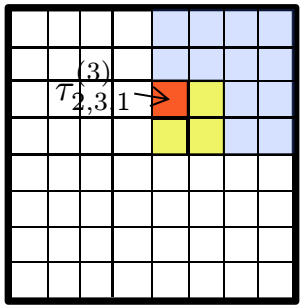
$$\phi_i^{(k)} \in H^{-s}(\Omega) \text{ with } k \in \{1, \ldots, q\}$$

$$\phi_i^{(k)} = \sum_j \pi_{i,j}^{(k,k+1)} \phi_j^{(k+1)}$$

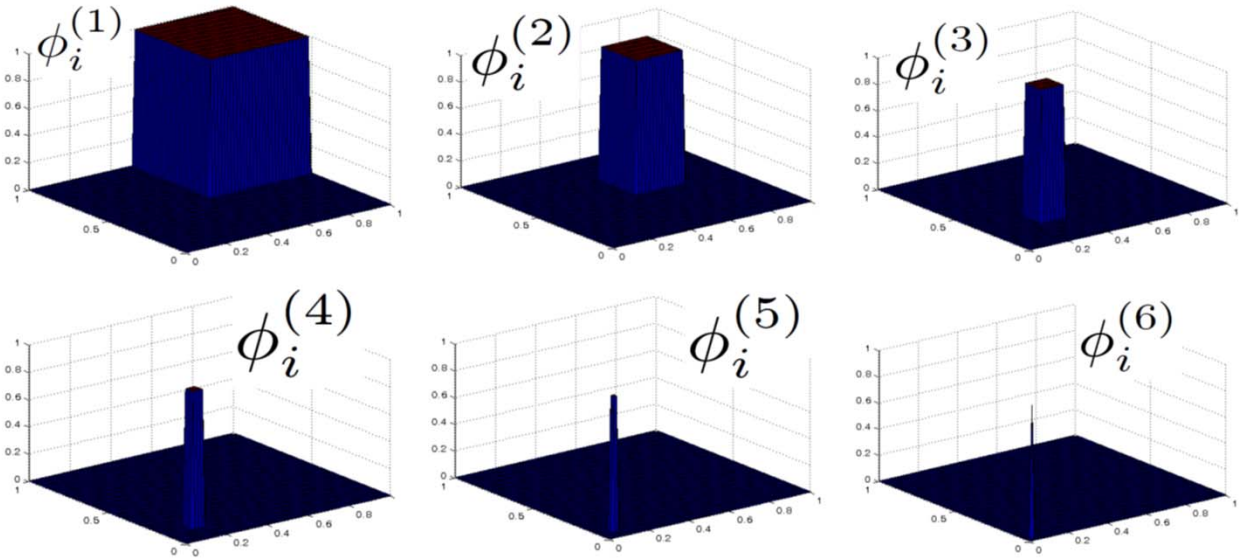
Example







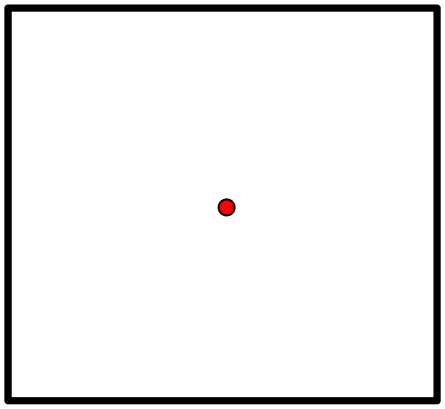
$\phi_i^{(k)}$: Haar (pre)-wavelets



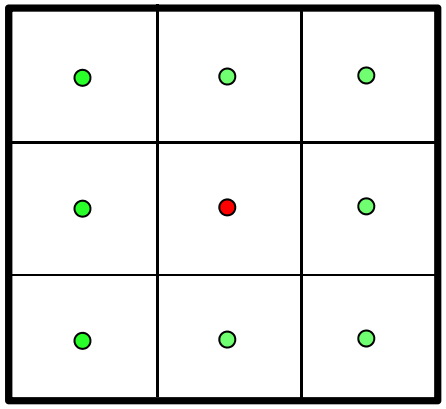
Example

$\phi_i^{(k)}$: Sub-sampled diracs

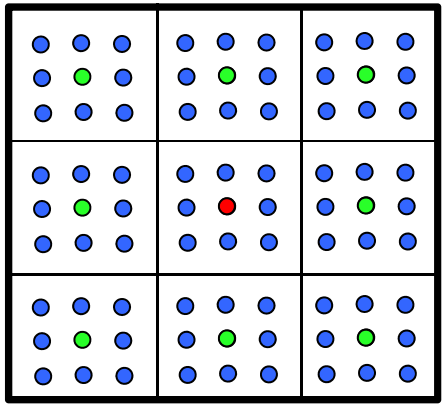
$$s > d/2$$



$\phi_i^{(1)} = \delta(x - x_i^{(1)})$



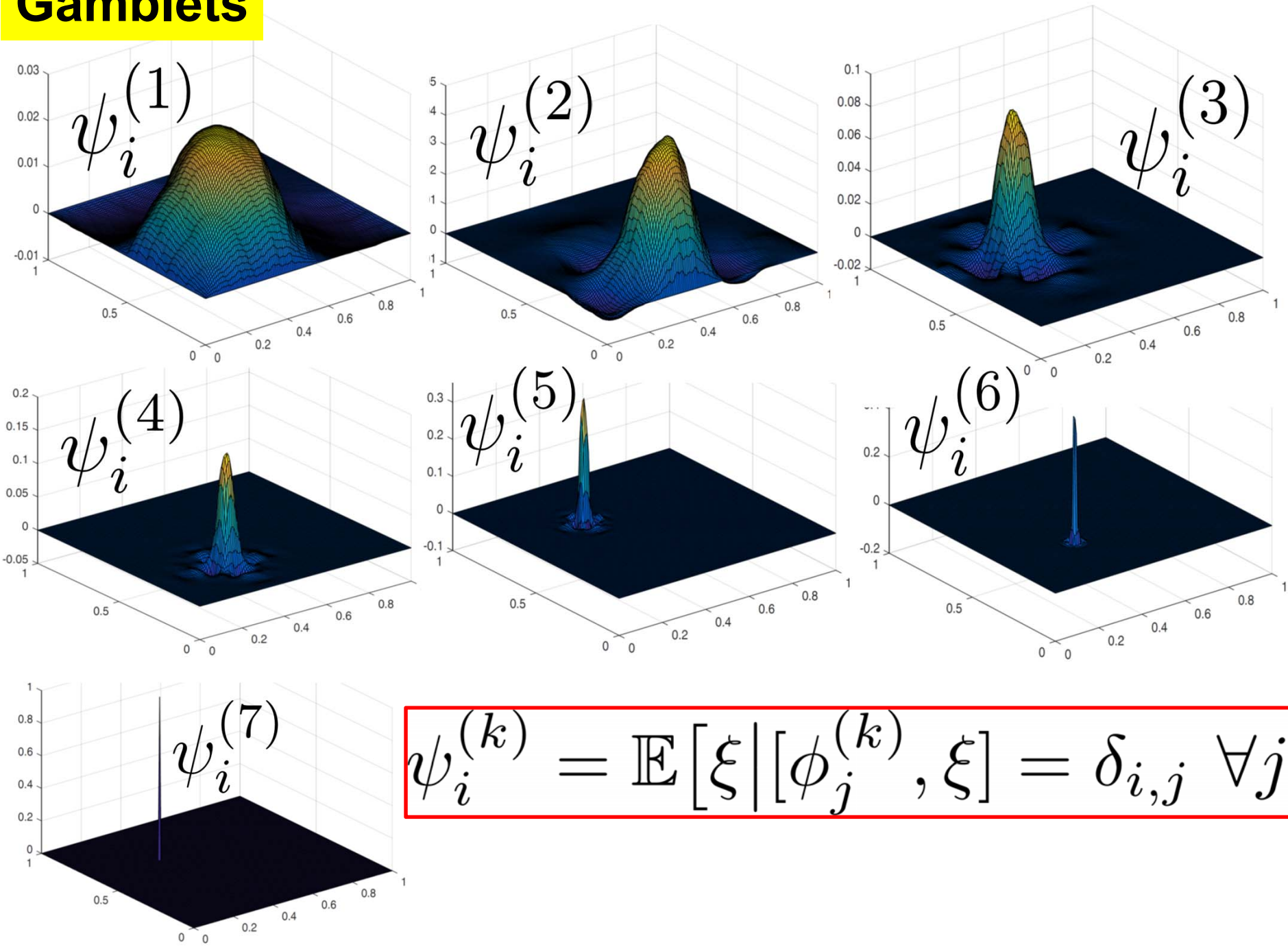
$\phi_j^{(2)} = \delta(x - x_j^{(2)})$



$\phi_l^{(3)} = \delta(x - x_l^{(3)})$

[Schäfer, Sullivan, O., 2017]

Gamblets



$$\psi_i^{(k)} = \mathbb{E}\left[\xi \middle| [\phi_j^{(k)}, \xi] = \delta_{i,j} \ \forall j\right]$$

Measurement functions are nested

$$\phi_i^{(k)} = \sum_j \pi_{i,j}^{(k,k+1)} \phi_j^{(k+1)}$$

$$\psi_i^{(k)} = \sum_{j \in \mathcal{I}^{(k)}} \Theta_{i,j}^{(k),-1} \mathcal{L}^{-1} \phi_j^{(k)}$$

$$\Theta_{i,j}^{(k)} := \int \phi_i^{(k)} \mathcal{L}^{-1} \phi_j^{(k)}$$

Gamblets are nested

$$\psi_i^{(k)} = \sum_{j \in \mathcal{I}^{(k+1)}} R_{i,j}^{(k,k+1)} \psi_j^{(k+1)}$$

$$\mathfrak{V}^{(k)} := \mathrm{span}\{\psi_i^{(k)} \mid i\}$$

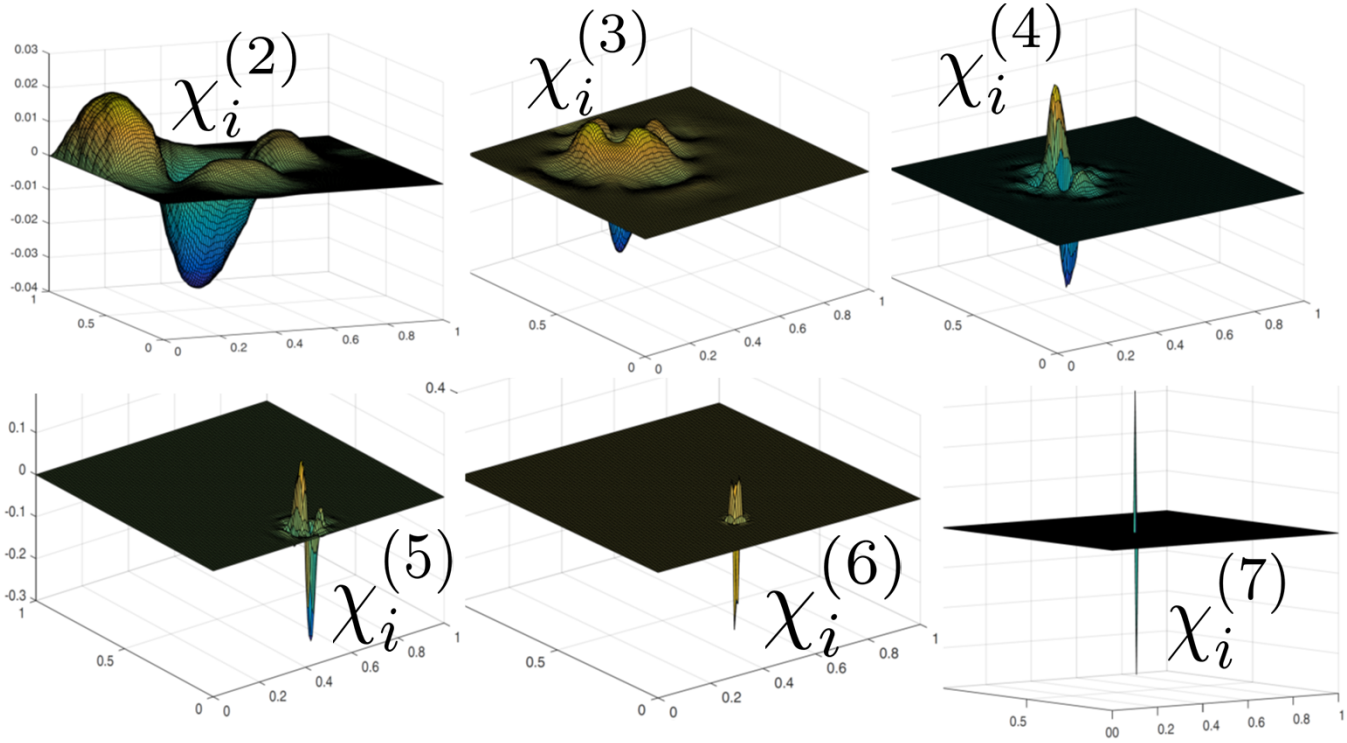
$$\mathfrak{V}^{(k-1)} \subset \mathfrak{V}^{(k)}$$

$$\mathfrak{V}^{(k)} = \mathfrak{V}^{(k-1)} \oplus \mathfrak{W}^{(k)}$$

$$\chi_i^{(k)} := \sum_{j \in \mathcal{I}^{(k)}} W_{i,j}^{(k)} \psi_j^{(k)}$$

$$\begin{array}{c} \mathrm{Img}(W^{(k),T}) \\ \| \\ \mathrm{Ker}(\pi^{(k-1,k)}) \end{array}$$

$$\phi_i^{(k)} = \sum_j \pi_{i,j}^{(k,k+1)} \phi_j^{(k+1)}$$



$$\pi_{i,\cdot}^{(1,2)} \begin{bmatrix} 0 & 0 & 1/2 & 1/2 \\ 0 & 0 & 1/2 & 1/2 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix}$$

$$W_{t,\cdot}^{(2)} \begin{bmatrix} 0 & 0 & \frac{1}{\sqrt{2}} & -\frac{1}{\sqrt{2}} \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix}$$

$$\mathfrak{W}^{(k)} := \mathrm{span}\{\chi_i^{(k)} \mid i\}$$

Theorem

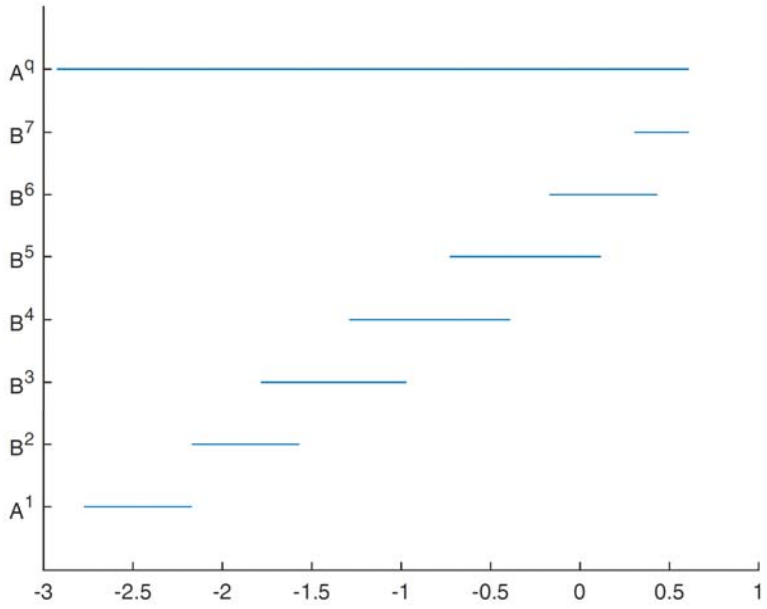
$$H_0^s(\Omega) = \mathfrak{V}^{(1)} \oplus \mathfrak{W}^{(2)} \oplus \mathfrak{W}^{(3)} \oplus \cdots$$

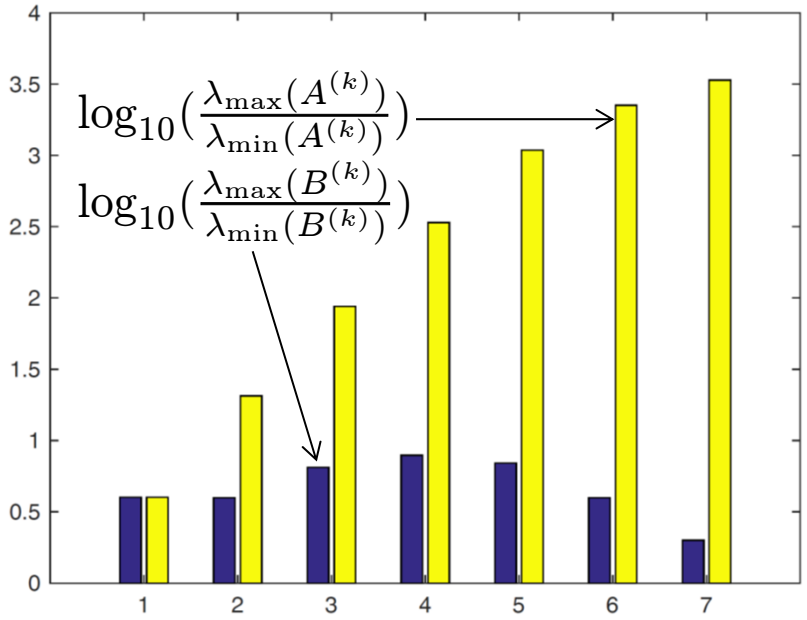
The operator is well conditioned in each subband

$$A_{i,j}^{(k)} = \langle \psi_i^{(k)}, \psi_j^{(k)} \rangle \qquad B_{i,j}^{(k)} = \langle \chi_i^{(k)}, \chi_j^{(k)} \rangle$$

Theorem

$$\mathrm{Cond}(A^{(1)}) \leq C \qquad \mathrm{Cond}(B^{(k)}) \leq C$$





[O., Scovel, 2017] [Schäfer, Sullivan, O., 2017]

Gamblet Transform

1: $\psi_i^{(q)} = \varphi_i$
2: $A_{i,j}^{(q)} = \langle \psi_i^{(q)}, \psi_j^{(q)} \rangle$
3: **for** $k = q$ to 2 **do**
4: $\quad B^{(k)} = W^{(k)} A^{(k)} W^{(k),T}$
5: $\quad \chi_i^{(k)} = \sum_{j \in \mathcal{I}^{(k)}} W_{i,j}^{(k)} \psi_j^{(k)}$
6: $\quad R^{(k-1,k)} = \pi^{(k-1,k)} (I^{(k)} - A^{(k)} W^{(k),T} B^{(k),-1} W^{(k)})$
7: $\quad A^{(k-1)} = R^{(k-1,k)} A^{(k)} R^{(k,k-1)}$
8: $\quad \psi_i^{(k-1)} = \sum_{j \in \mathcal{I}^{(k)}} R_{i,j}^{(k-1,k)} \psi_j^{(k)}$
9: **end for**

Theorem

$\mathcal{O}\big(N \log^{2d+1}(N)\big)$ complexity
to achieve grid size accuracy in energy norm

Gamblet Solve

1: $f_i^{(q)} = \int_\Omega f \psi_i^{(q)}$
2: **for** $k = q$ to 2 **do**
3: $\quad w^{(k)} = B^{(k),-1} W^{(k)} f^{(k)}$
4: $\quad u^{(k)} - u^{(k-1)} = \sum_{i \in \mathcal{J}^{(k)}} w_i^{(k)} \chi_i^{(k)}$
5: $\quad f^{(k-1)} = R^{(k-1,k)} f^{(k)}$
6: **end for**
7: $U^{(1)} = A^{(1),-1} f^{(1)}$
8: $u^{(1)} = \sum_{i \in \mathcal{I}^{(1)}} U_i^{(1)} \psi_i^{(1)}$
9: $u = u^{(1)} + (u^{(2)} - u^{(1)}) + \cdots + (u^{(q)} - u^{(q-1)})$

Theorem

$\mathcal{O}\big(N \log^{d+1}(N)\big)$ complexity
to achieve grid size accuracy in energy norm

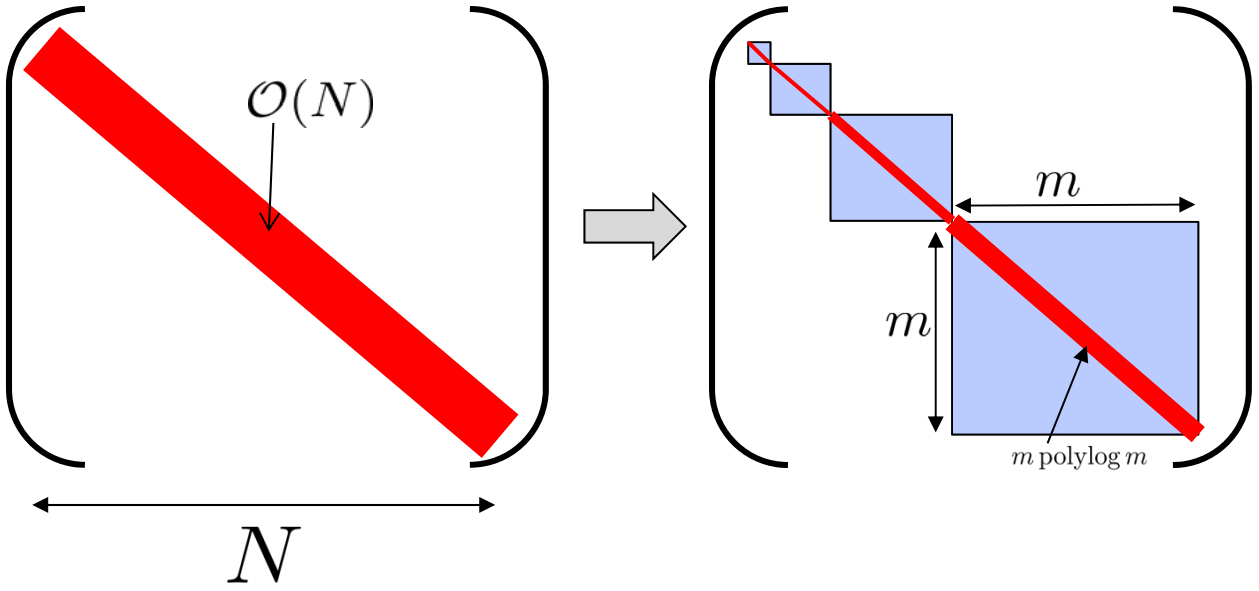
$\mathcal{L}$: Arbitrary symmetric positive continuous linear bijection

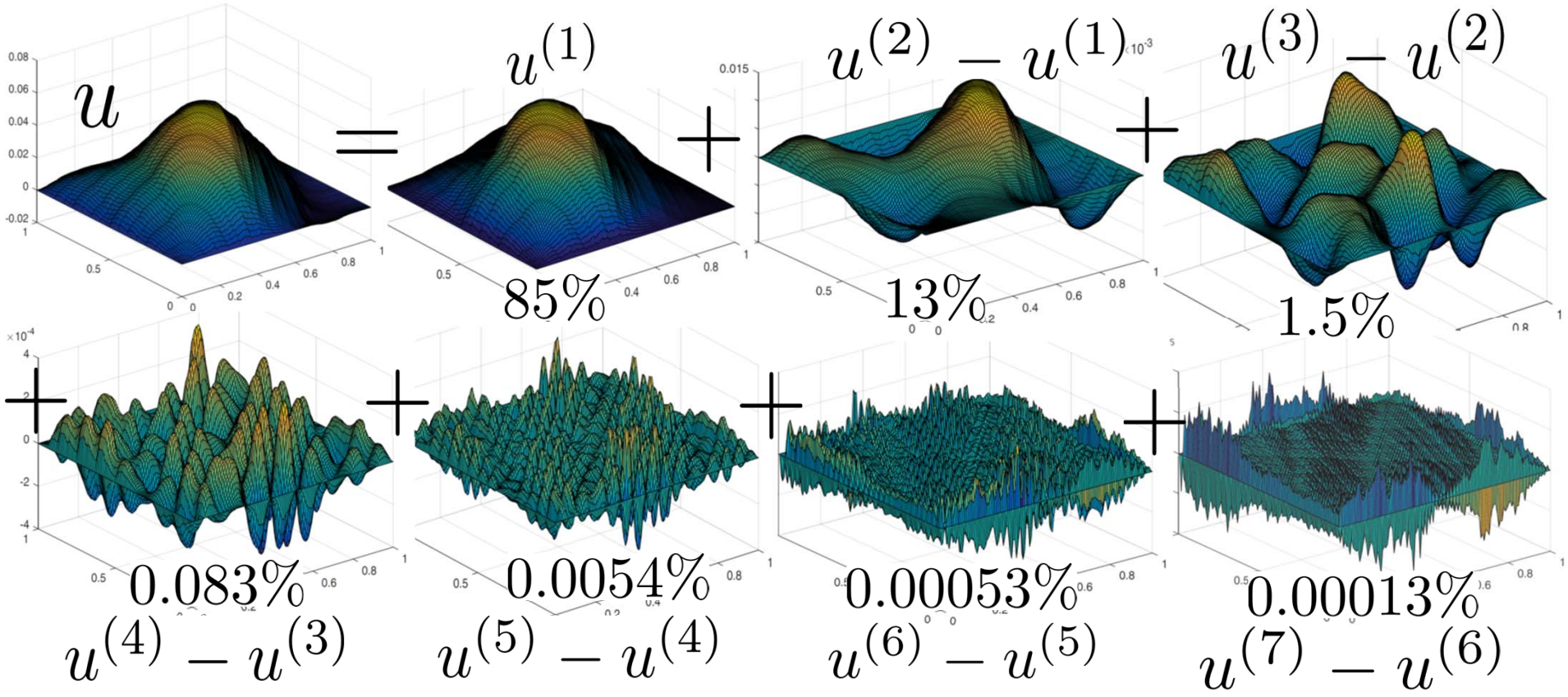
$$H_0^s(\Omega) \xrightarrow{\quad\mathcal{L}\quad} H^{-s}(\Omega)$$

Gamblet transform

$$H_0^s(\Omega) = \mathfrak{V}^{(1)} \oplus \mathfrak{W}^{(2)} \oplus \mathfrak{W}^{(3)} \oplus \cdots$$

$$\|u\|^2 := \int_\Omega u\mathcal{L}u$$





Energy content

$$\begin{cases} -\operatorname{div}(a\nabla u) = f, & x \in \Omega, \\ u = 0, & x \in \partial\Omega, \end{cases}$$

$$f \in C^{\infty}(\Omega)$$

Florian Schäfer

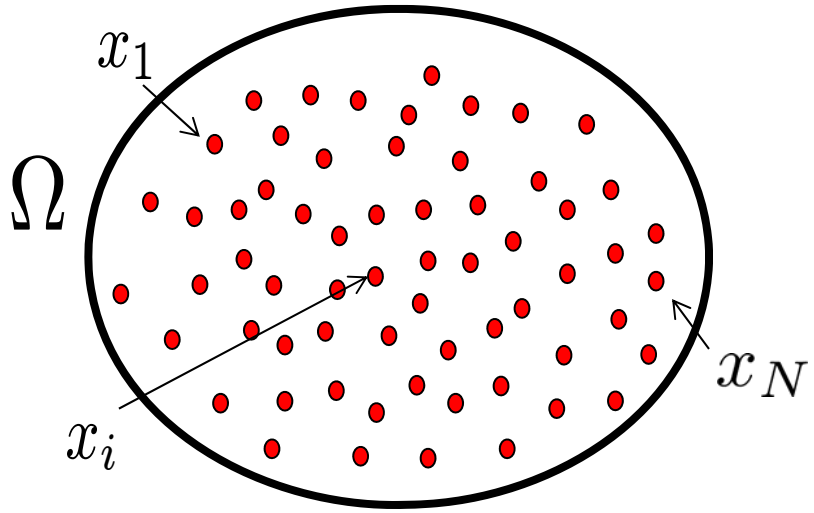
- Compression, inversion, and approximate PCA of dense kernel matrices at near-linear computational complexity. Schäfer, Sullivan, O. 2017. arXiv:1706.02205

https://github.com/f-t-s/nearLinKernel.git

Tim Sullivan

$$(H_0^s(\Omega), \|\cdot\|_{H_0^s(\Omega)}) \xrightarrow{\mathcal{L}} (H^{-s}(\Omega), \|\cdot\|_{H^{-s}(\Omega)})$$

G : Green's function of $\mathcal{L}$ $\qquad s > \frac{d}{2}$



$x_1, \ldots, x_N$: Approximately homogeneous

$$\Theta_{i,j} := G(x_i, x_j)$$

Computational bottleneck

Θ is dense, naively we have

- Storage, $\mathcal{O}(N^2)$
- Θv, $\mathcal{O}(N^2)$
- $\Theta^{-1} v$, $\mathcal{O}(N^3)$
- $\det(\Theta)$, $\mathcal{O}(N^3)$
- $\mathrm{PCA}(\Theta)$, $\mathcal{O}(N^3)$

Algorithm

For $\epsilon > 0$ knowing only Ω and $\{x_i\}_{1 \leq i \leq N}$, we will

- Select $\mathcal{O}(N \operatorname{polylog}(N) \operatorname{polylog}(\frac{1}{\epsilon}))$ entries of Θ and an ordering P of $\{x_i\}_{1 \leq i \leq N}$.
- From these entries compute a lower triangular matrix L such that $nnz(L) = \mathcal{O}(N \operatorname{polylog}(N) \operatorname{polylog}(\frac{1}{\epsilon}))$.

Theorem

The above can be done in complexity $N \operatorname{polylog}(N) \operatorname{polylog}(\frac{1}{\epsilon})$, in time and space, such that

$$\|\Theta - PLL^T P^T\| \leq \epsilon$$

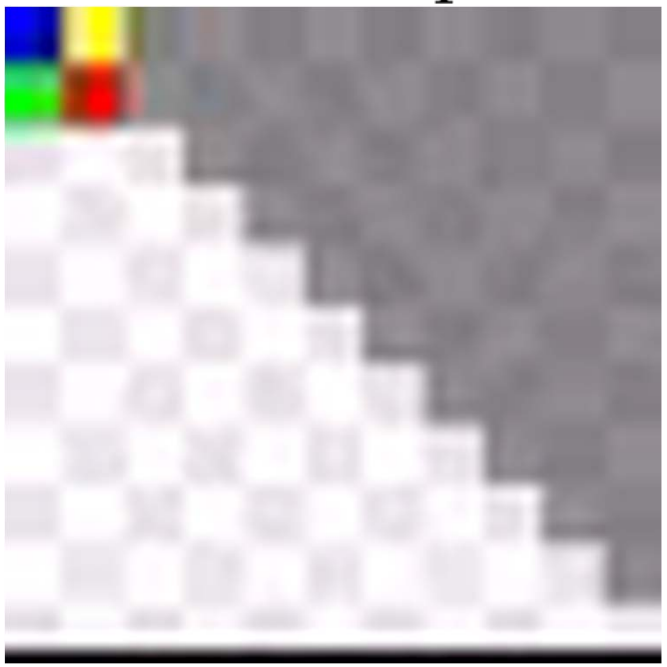
Allows to approximate Θv, $\Theta^{-1} v$, $\det(\Theta)$, in $\mathcal{O}(N \operatorname{polylogN} \operatorname{polylog} \frac{1}{\epsilon})$ complexity

Incomplete Cholesky factorization

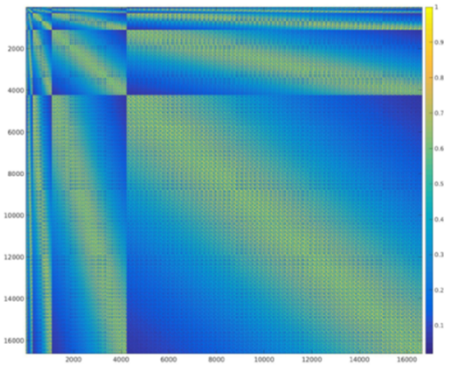
Cholesky factorization $A = LL^T$ can be computed as

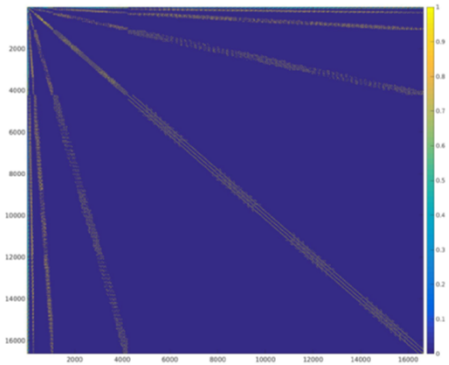
Algorithm 1: Cholesky factorisation

for $i \leftarrow 1$ *to* N **do**
  $A_{i,i} \leftarrow \sqrt{A_{i,i}}$;
  for $j \leftarrow i+1$ *to* N **do**
    for $k \leftarrow j$ *to* N **do**
      $A_{k,j} \leftarrow A_{k,j} - A_{k,i}A_{j,i}/A_{i,i}$;
  $A_{:,i} \leftarrow A_{:,i}/\sqrt{A_{i,i}}$;
return LowerTriang (A)



- One small Tweak: Skip all operations, for which (k, j), (k, i), or (j, i) are outside of the sparsity pattern.

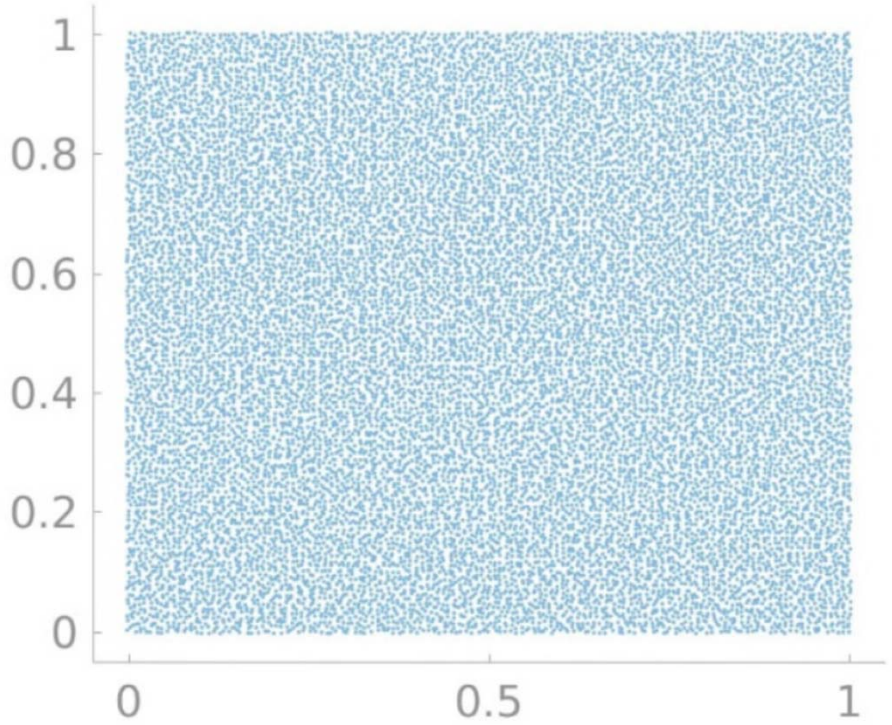




A simple algorithm

- Decompose $\{x_i\}_{i\in\mathcal{I}}$ into a nested hierarchy:

$$\{x_i\}_{i\in\mathcal{I}^{(1)}} \subset \{x_i\}_{i\in\mathcal{I}^{(2)}} \subset \{x_i\}_{i\in\mathcal{I}^{(3)}} \subset \cdots \subset \{x_i\}_{i\in\mathcal{I}^{(q)}}$$



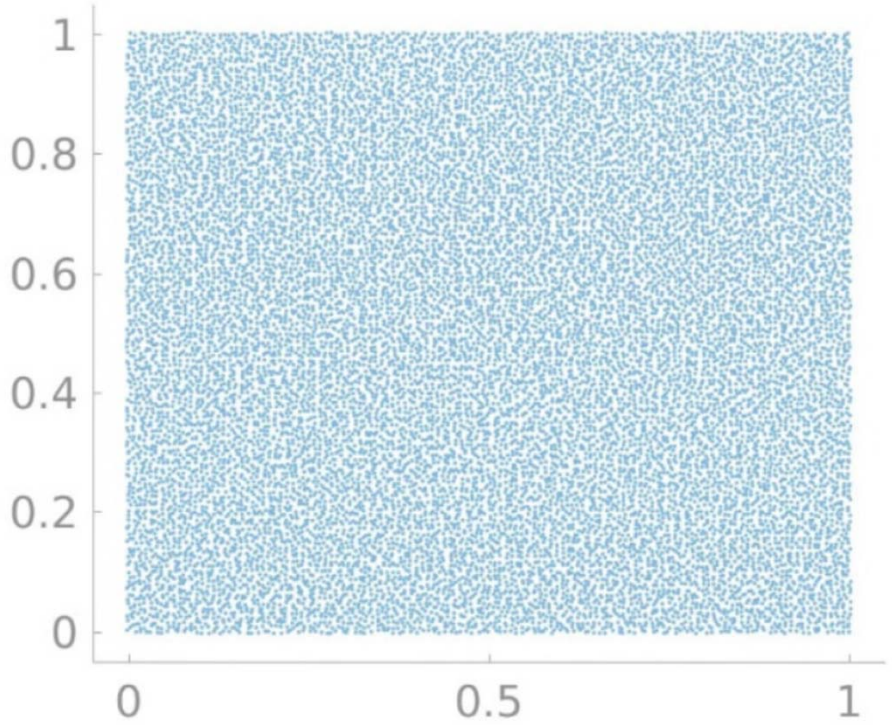
A simple algorithm

- Decompose $\{x_i\}_{i\in\mathcal{I}}$ into a nested hierarchy:

$$\{x_i\}_{i\in\mathcal{I}^{(1)}} \subset \{x_i\}_{i\in\mathcal{I}^{(2)}} \subset \{x_i\}_{i\in\mathcal{I}^{(3)}} \subset \cdots \subset \{x_i\}_{i\in\mathcal{I}^{(q)}}$$

- Define

$$\mathcal{J}^{(k)} := \mathcal{I}^{(k)} / I^{(k-1)}$$



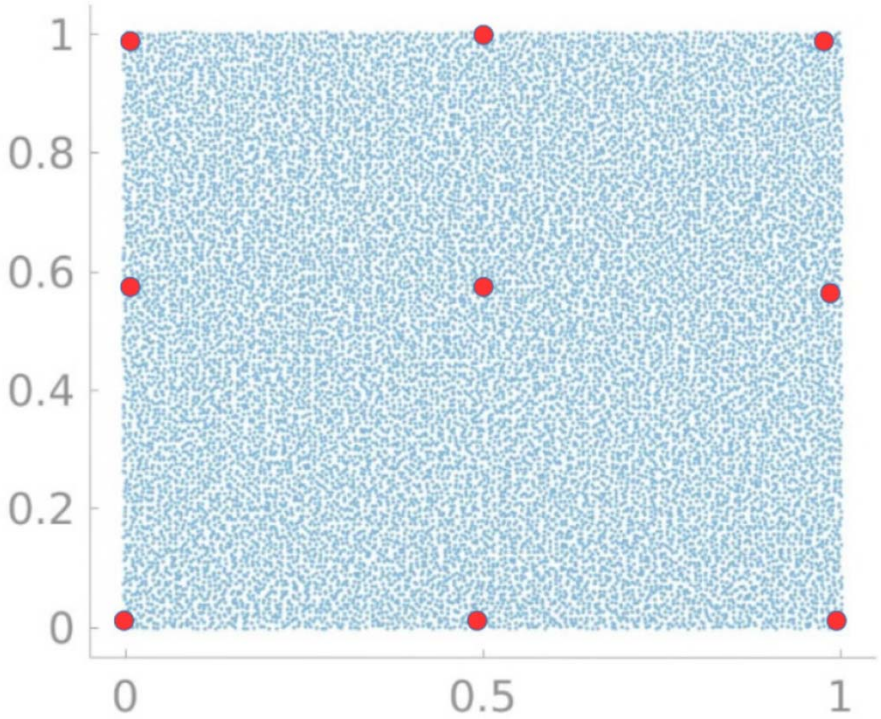
A simple algorithm

- Decompose $\{x_i\}_{i\in\mathcal{I}}$ into a nested hierarchy:

$$\{x_i\}_{i\in\mathcal{I}^{(1)}} \subset \{x_i\}_{i\in\mathcal{I}^{(2)}} \subset \{x_i\}_{i\in\mathcal{I}^{(3)}} \subset \cdots \subset \{x_i\}_{i\in\mathcal{I}^{(q)}}$$

- Define

$$\mathcal{I}^{(1)} = \mathcal{J}^{(1)}$$



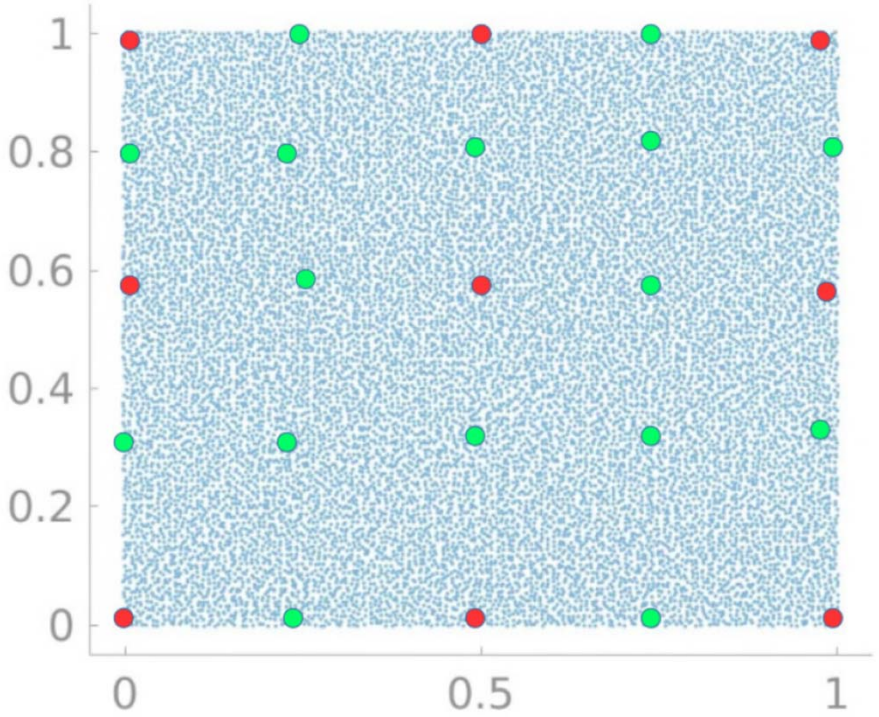
A simple algorithm

- Decompose $\{x_i\}_{i\in\mathcal{I}}$ into a nested hierarchy:

$$\{x_i\}_{i\in\mathcal{I}^{(1)}} \subset \{x_i\}_{i\in\mathcal{I}^{(2)}} \subset \{x_i\}_{i\in\mathcal{I}^{(3)}} \subset \cdots \subset \{x_i\}_{i\in\mathcal{I}^{(q)}}$$

- Define

$$\mathcal{I}^{(2)} = \mathcal{J}^{(1)} \cup \mathcal{J}^{(2)}$$



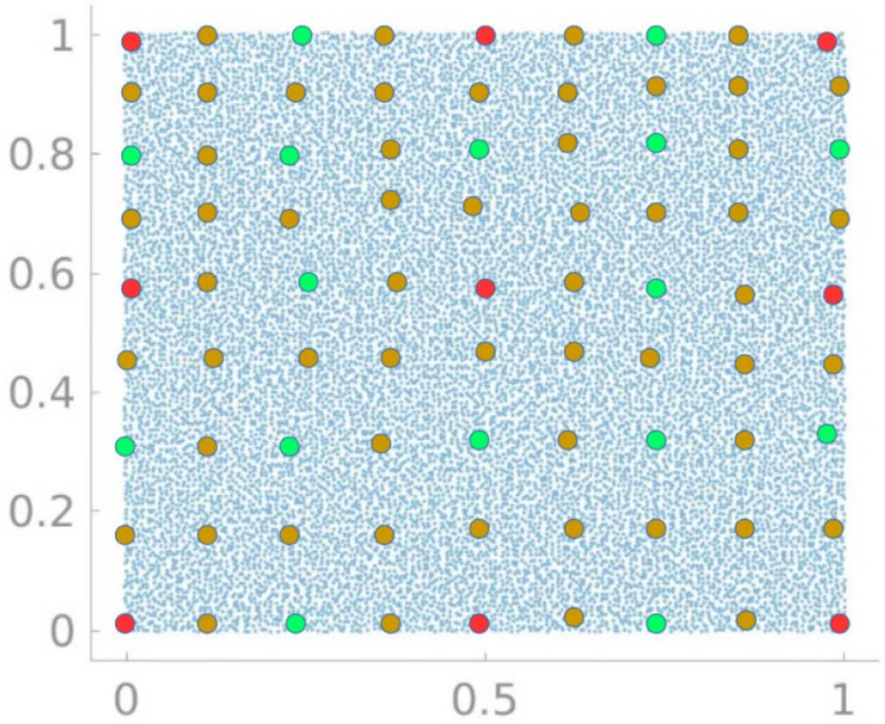
A simple algorithm

- Decompose $\{x_i\}_{i\in\mathcal{I}}$ into a nested hierarchy:

$$\{x_i\}_{i\in\mathcal{I}^{(1)}} \subset \{x_i\}_{i\in\mathcal{I}^{(2)}} \subset \{x_i\}_{i\in\mathcal{I}^{(3)}} \subset \cdots \subset \{x_i\}_{i\in\mathcal{I}^{(q)}}$$

- Define

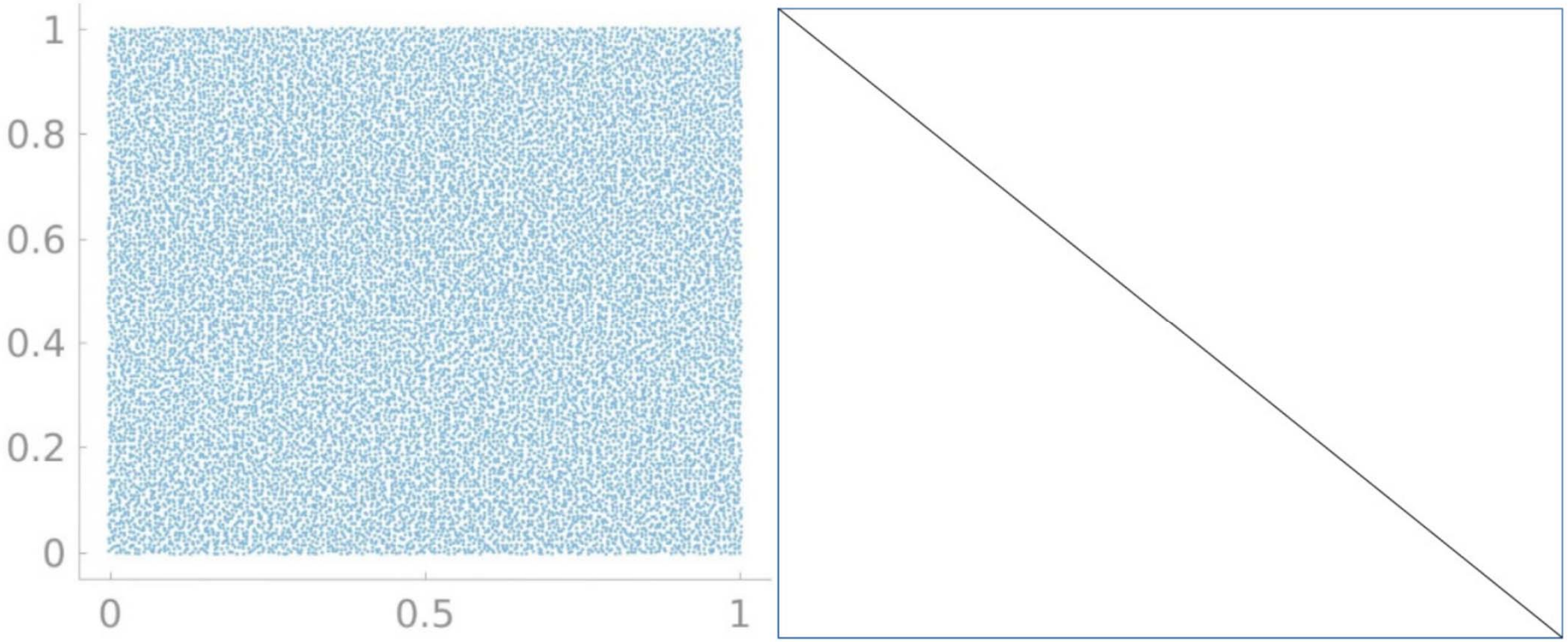
$$\mathcal{I}^{(3)} = \mathcal{J}^{(1)} \cup \mathcal{J}^{(2)} \cup \mathcal{J}^{(3)}$$



A simple algorithm

- We order the degrees of freedom from $\mathcal{J}^{(1)}$ to $\mathcal{J}^{(q)}$ and define the sparsity pattern:

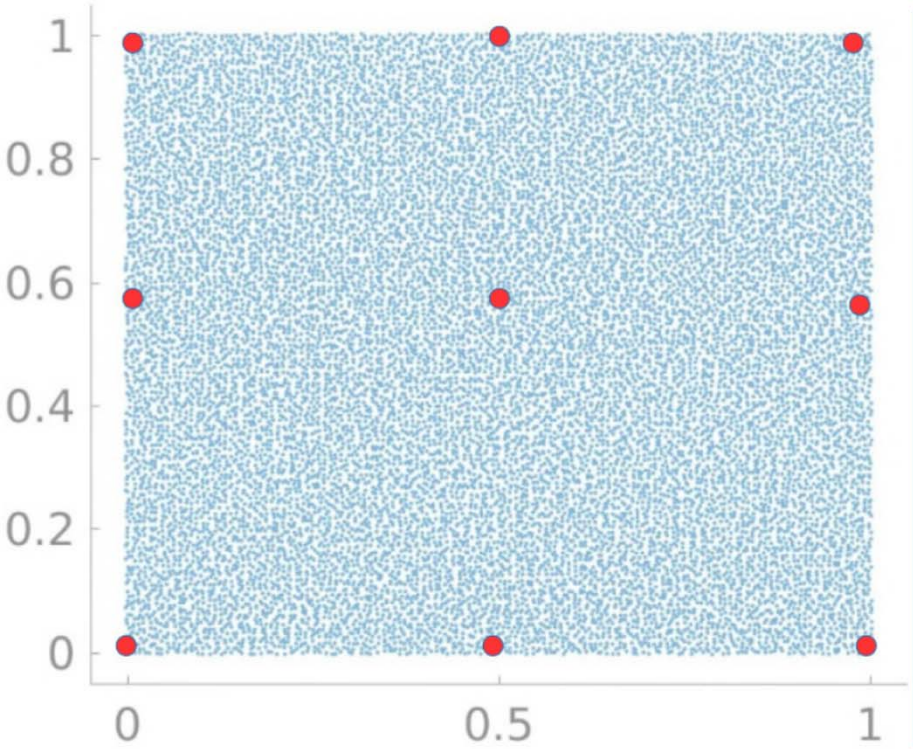
$$S := \left\{(i,j) \in \mathcal{I} \times \mathcal{I} \middle| i \in \mathcal{J}^{(k)},\, j \in \mathcal{J}^{(l)},\, \mathrm{dist}(x_i, x_j) \leq \ln \tfrac{1}{\epsilon} \times 2^{-\min(k,l)} \right\}$$

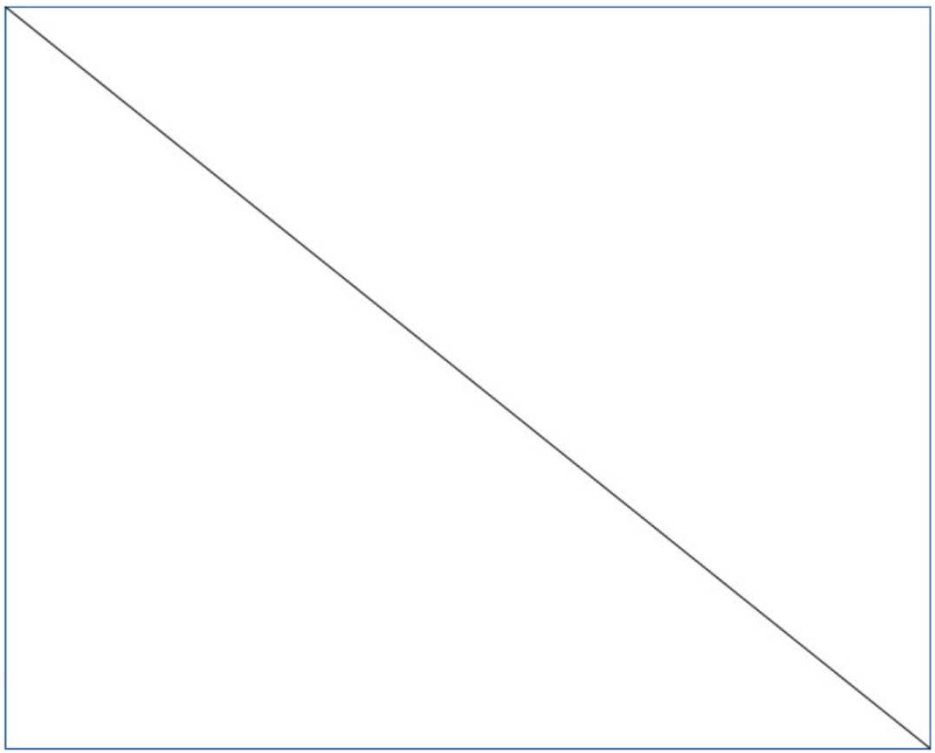


A simple algorithm

- We order the degrees of freedom from $\mathcal{J}^{(1)}$ to $\mathcal{J}^{(q)}$ and define the sparsity pattern:

$$S := \left\{(i,j) \in \mathcal{I} \times \mathcal{I} \middle| i \in \mathcal{J}^{(k)},\, j \in \mathcal{J}^{(l)},\, \mathrm{dist}(x_i, x_j) \leq \ln \tfrac{1}{\epsilon} \times 2^{-\min(k,l)} \right\}$$

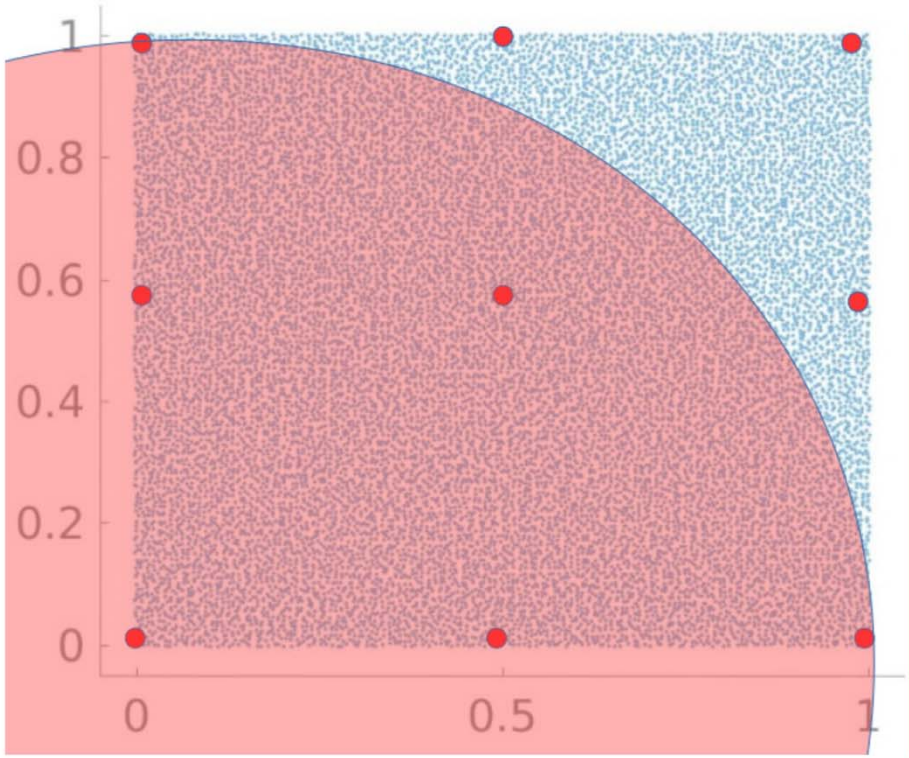


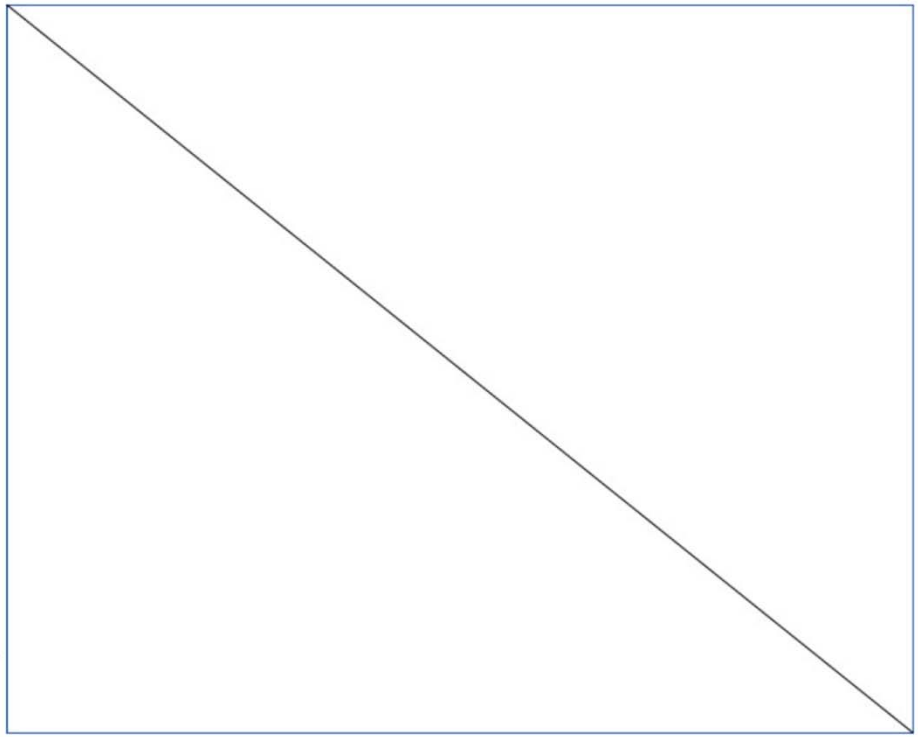


A simple algorithm

- We order the degrees of freedom from $\mathcal{J}^{(1)}$ to $\mathcal{J}^{(q)}$ and define the sparsity pattern:

$$S := \left\{(i,j) \in \mathcal{I} \times \mathcal{I} \middle| i \in \mathcal{J}^{(k)},\, j \in \mathcal{J}^{(l)},\, \mathrm{dist}(x_i, x_j) \leq \ln \tfrac{1}{\epsilon} \times 2^{-\min(k,l)} \right\}$$

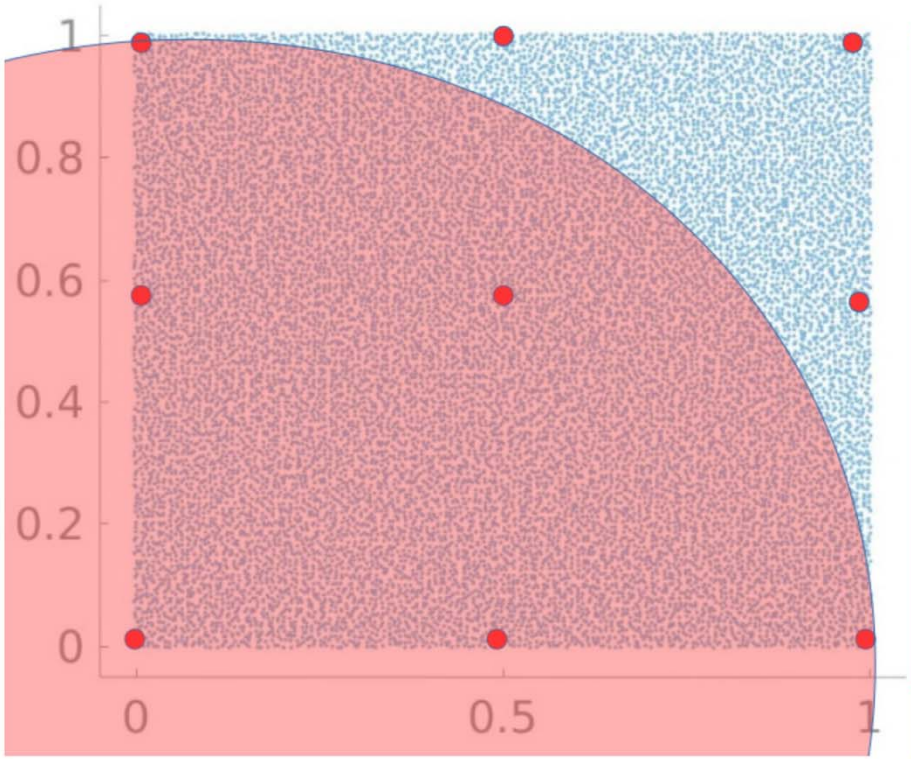


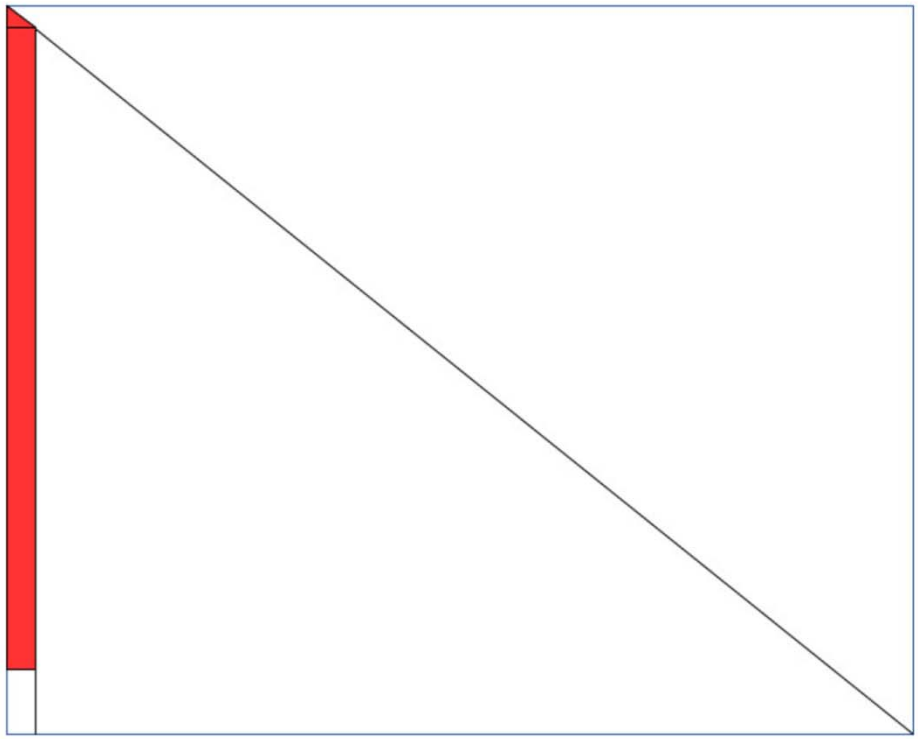


A simple algorithm

- We order the degrees of freedom from $\mathcal{J}^{(1)}$ to $\mathcal{J}^{(q)}$ and define the sparsity pattern:

$$S := \left\{(i,j) \in \mathcal{I} \times \mathcal{I} \middle| i \in \mathcal{J}^{(k)},\, j \in \mathcal{J}^{(l)},\, \mathrm{dist}(x_i, x_j) \leq \ln \tfrac{1}{\epsilon} \times 2^{-\min(k,l)} \right\}$$

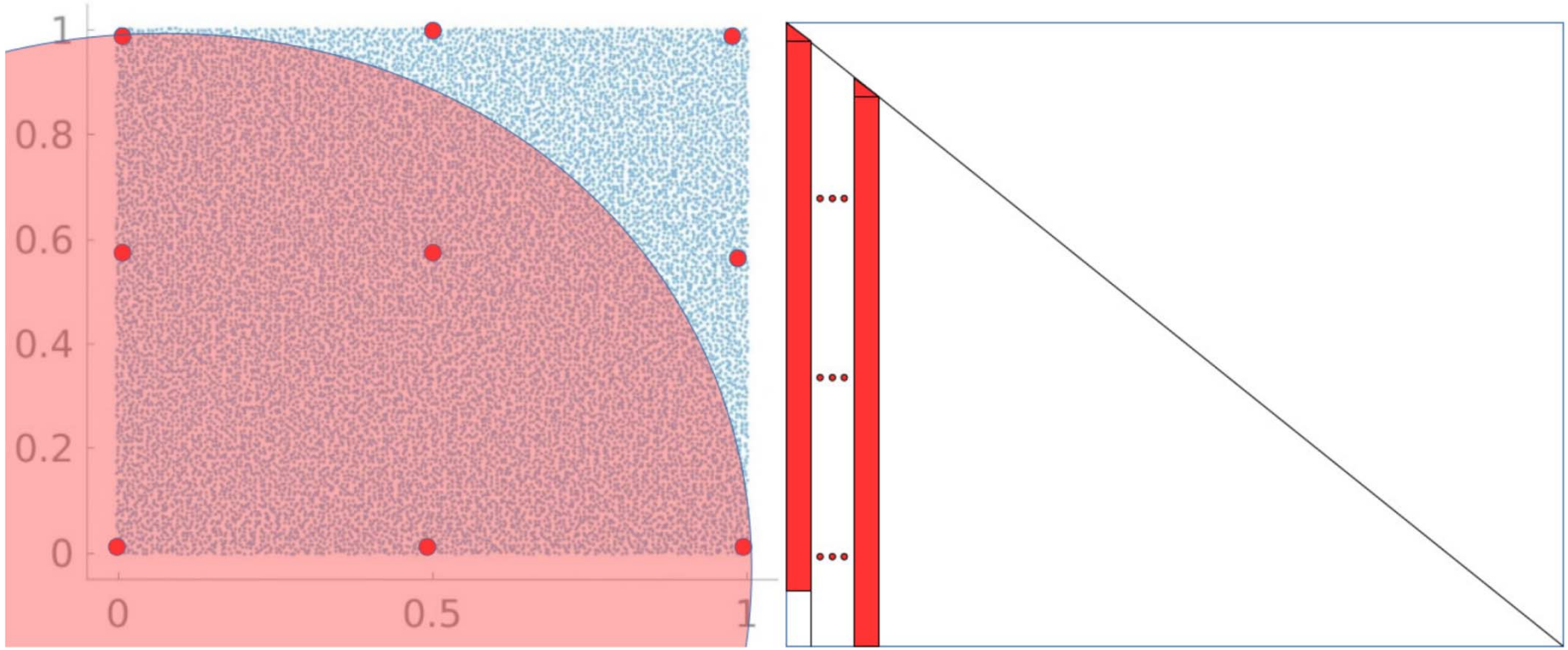




A simple algorithm

- We order the degrees of freedom from $\mathcal{J}^{(1)}$ to $\mathcal{J}^{(q)}$ and define the sparsity pattern:

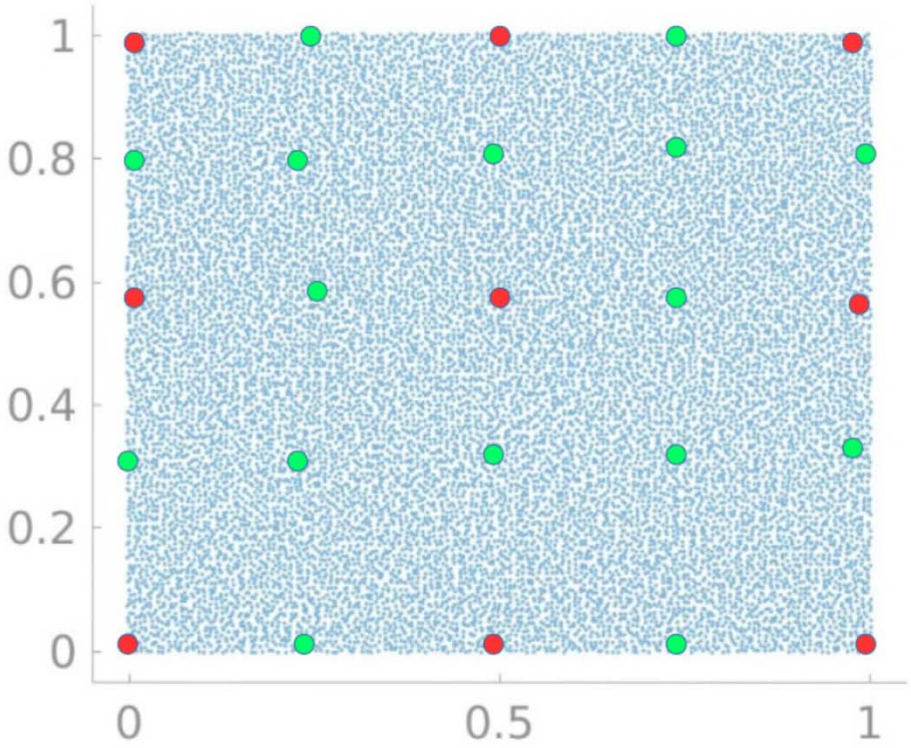
$$S := \left\{(i,j) \in \mathcal{I} \times \mathcal{I} \middle| i \in \mathcal{J}^{(k)},\, j \in \mathcal{J}^{(l)},\, \mathrm{dist}(x_i, x_j) \leq \ln \tfrac{1}{\epsilon} \times 2^{-\min(k,l)} \right\}$$

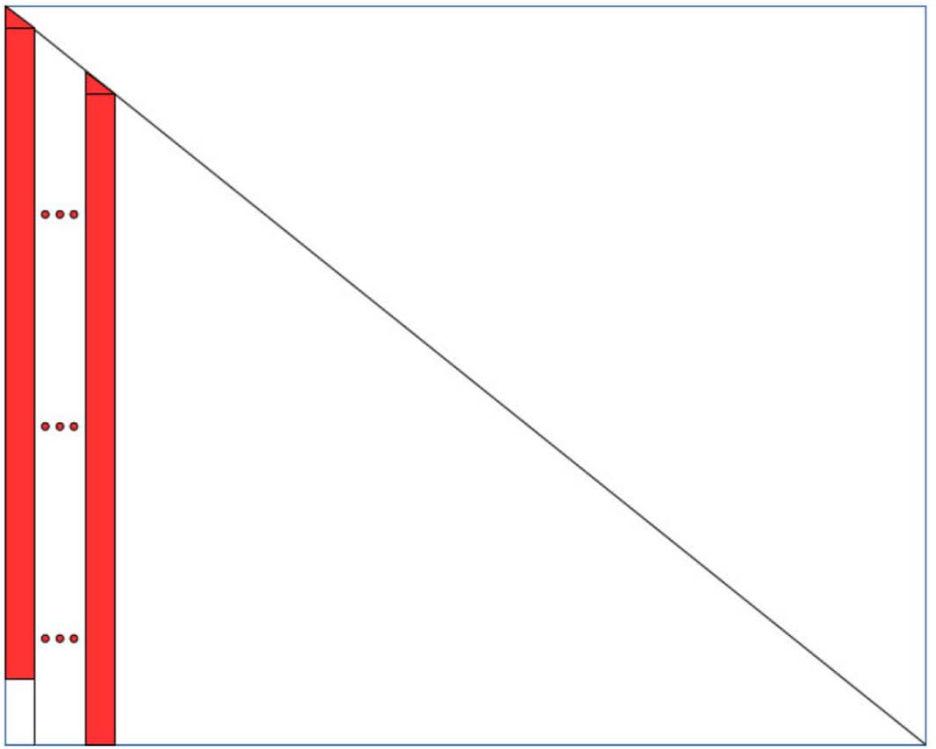


A simple algorithm

- We order the degrees of freedom from $\mathcal{J}^{(1)}$ to $\mathcal{J}^{(q)}$ and define the sparsity pattern:

$$S := \left\{(i,j) \in \mathcal{I} \times \mathcal{I} \middle| i \in \mathcal{J}^{(k)},\, j \in \mathcal{J}^{(l)},\, \mathrm{dist}(x_i, x_j) \leq \ln \tfrac{1}{\epsilon} \times 2^{-\min(k,l)} \right\}$$

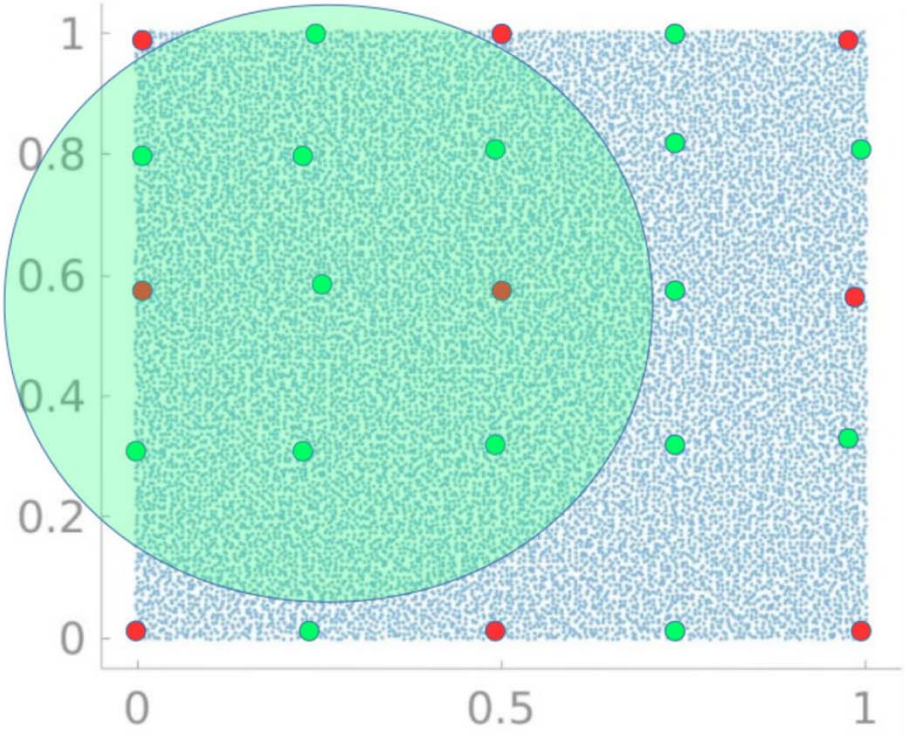


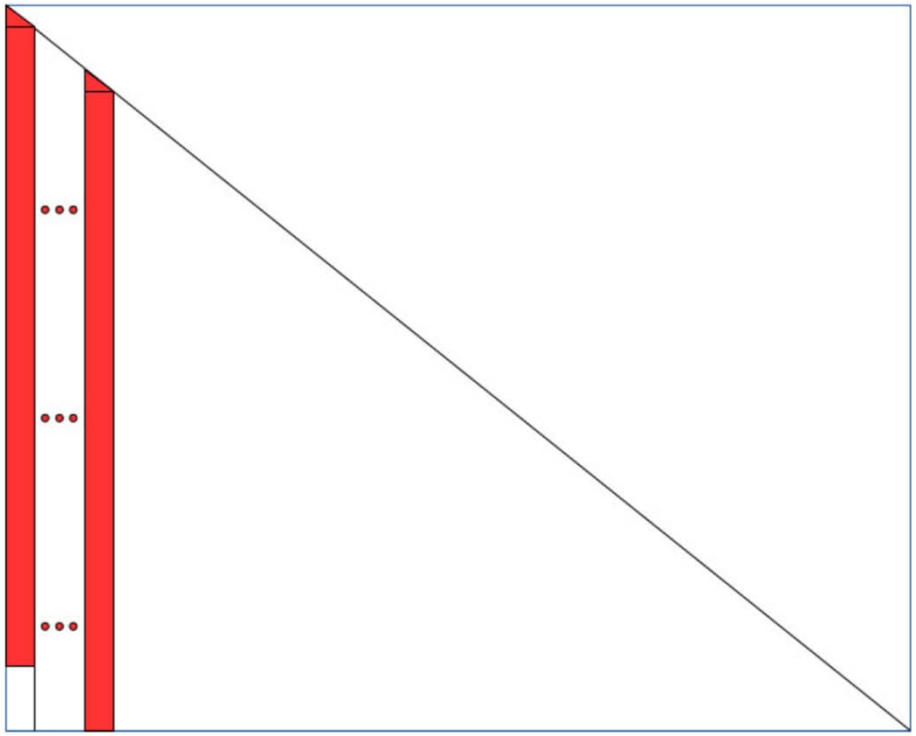


A simple algorithm

- We order the degrees of freedom from $\mathcal{J}^{(1)}$ to $\mathcal{J}^{(q)}$ and define the sparsity pattern:

$$S := \left\{(i,j) \in \mathcal{I} \times \mathcal{I} \middle| i \in \mathcal{J}^{(k)},\, j \in \mathcal{J}^{(l)},\, \mathrm{dist}(x_i, x_j) \leq \ln \tfrac{1}{\epsilon} \times 2^{-\min(k,l)} \right\}$$

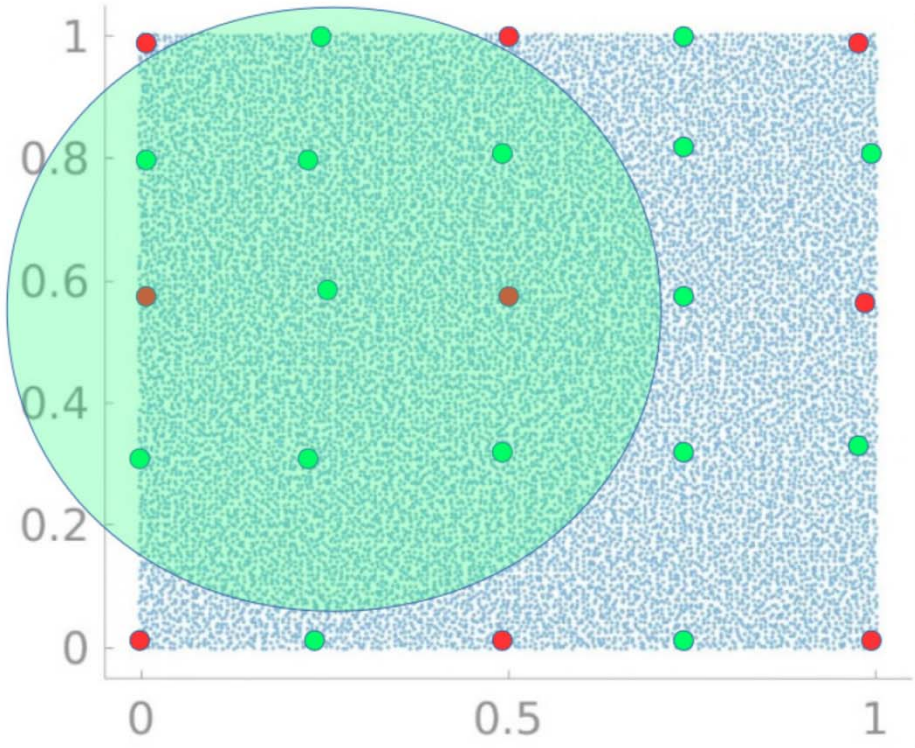


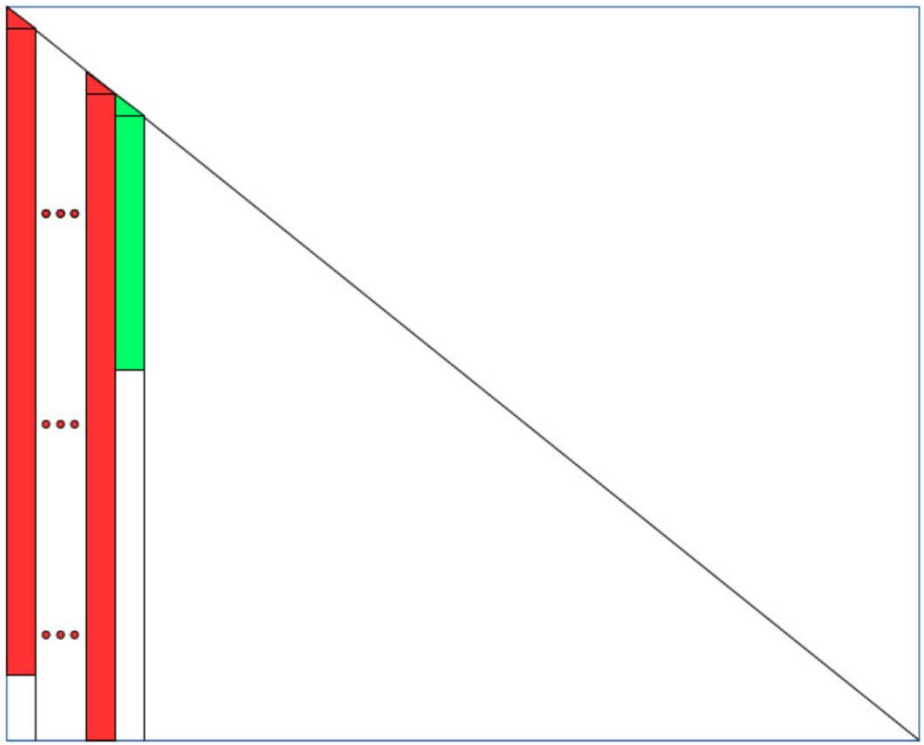


A simple algorithm

- We order the degrees of freedom from $\mathcal{J}^{(1)}$ to $\mathcal{J}^{(q)}$ and define the sparsity pattern:

$$S := \left\{(i,j) \in \mathcal{I} \times \mathcal{I} \middle| i \in \mathcal{J}^{(k)},\, j \in \mathcal{J}^{(l)},\, \mathrm{dist}(x_i, x_j) \leq \ln \tfrac{1}{\epsilon} \times 2^{-\min(k,l)} \right\}$$

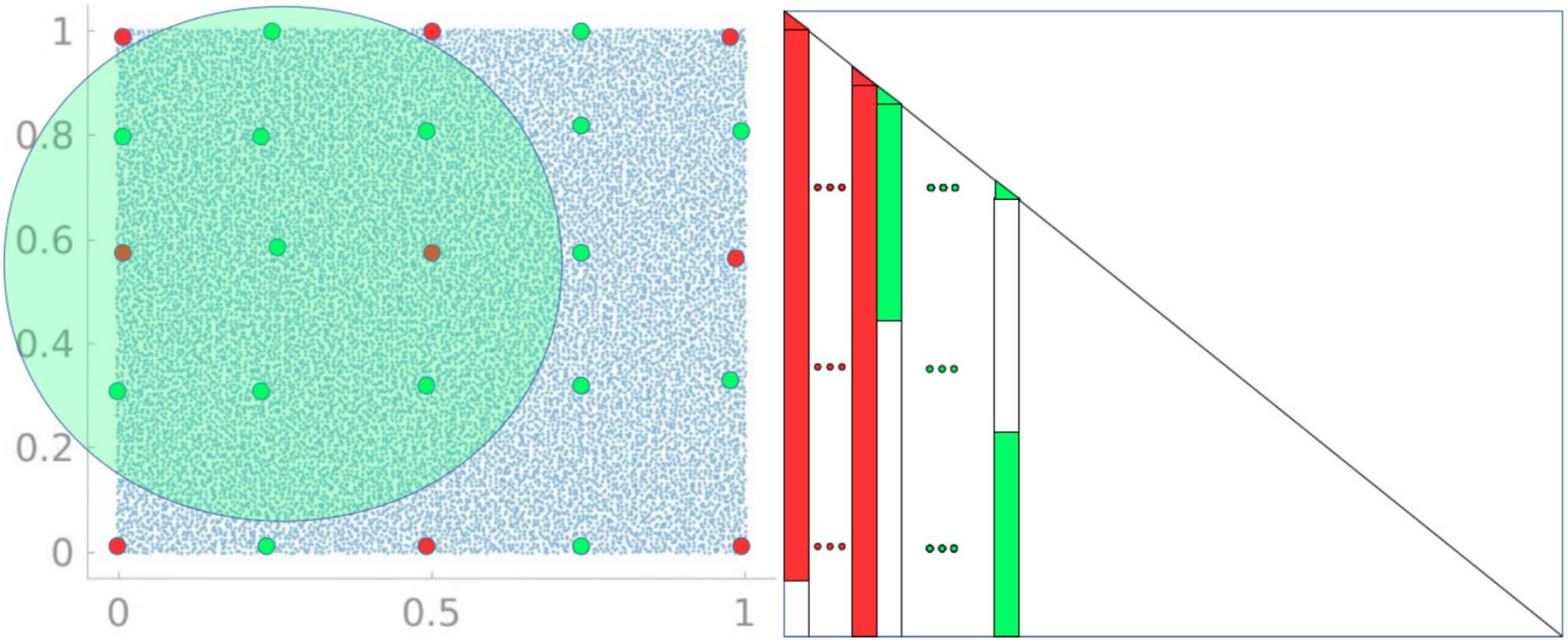




A simple algorithm

- We order the degrees of freedom from $\mathcal{J}^{(1)}$ to $\mathcal{J}^{(q)}$ and define the sparsity pattern:

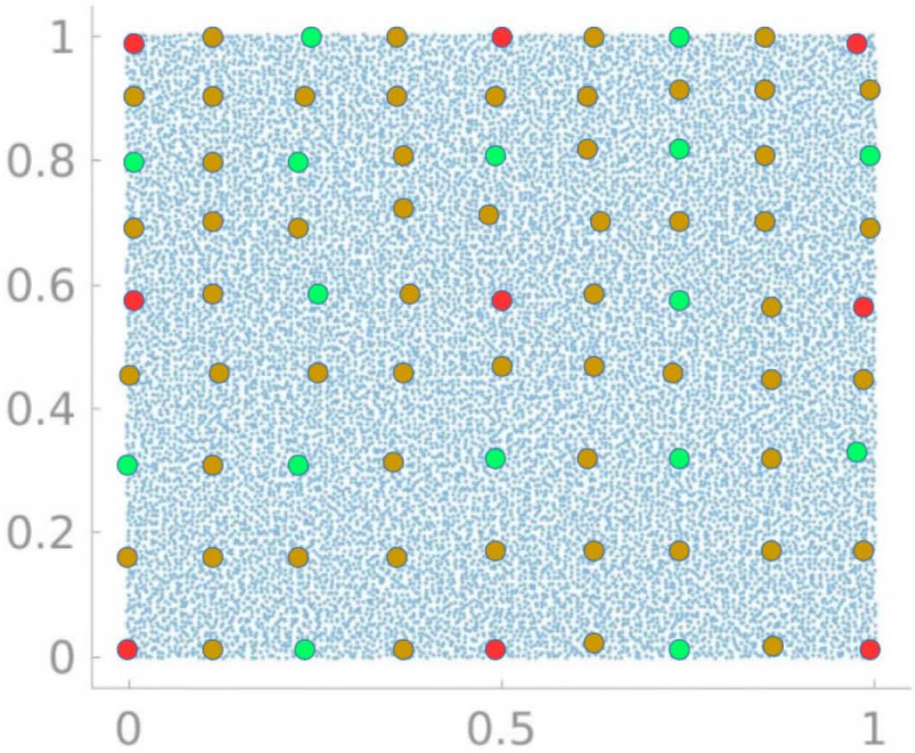
$$S := \left\{(i,j) \in \mathcal{I} \times \mathcal{I} \middle| i \in \mathcal{J}^{(k)},\, j \in \mathcal{J}^{(l)},\, \mathrm{dist}(x_i, x_j) \leq \ln \tfrac{1}{\epsilon} \times 2^{-\min(k,l)} \right\}$$



A simple algorithm

- We order the degrees of freedom from $\mathcal{J}^{(1)}$ to $\mathcal{J}^{(q)}$ and define the sparsity pattern:

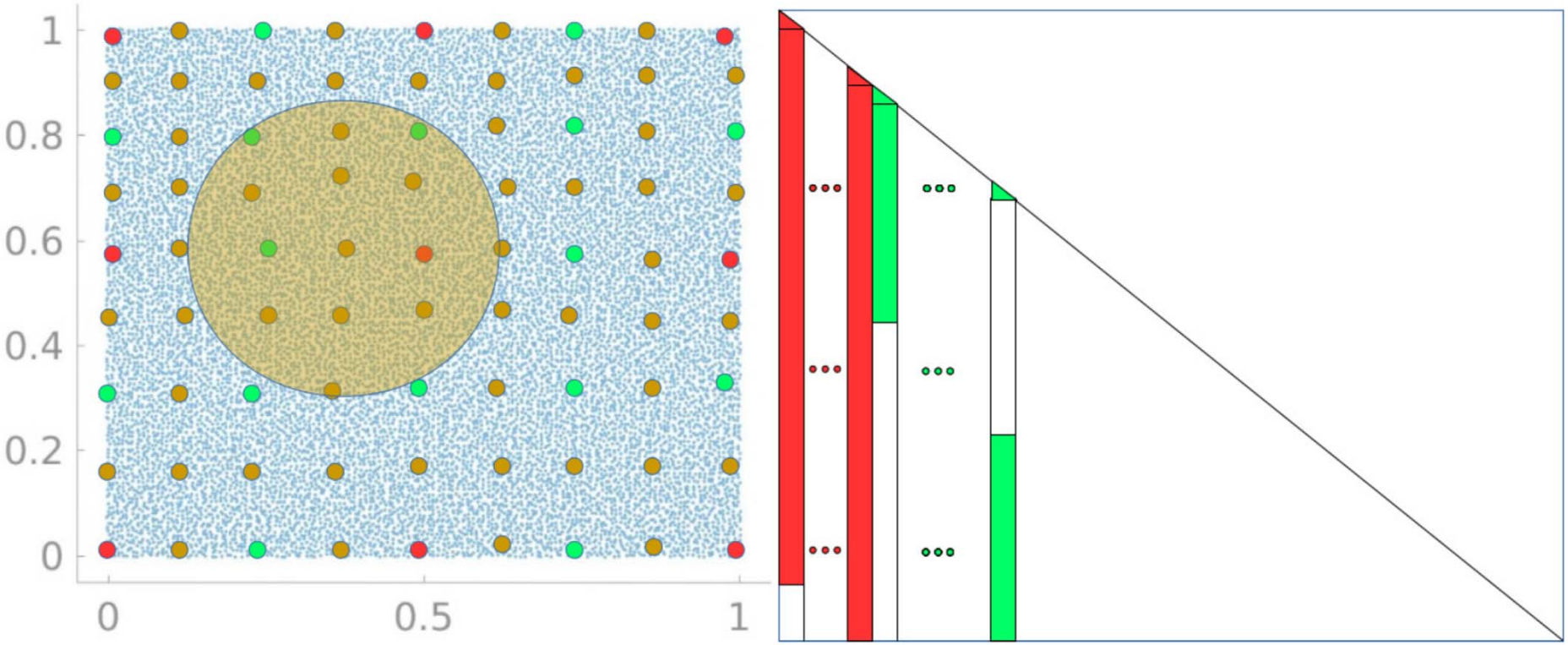
$$S := \left\{(i,j) \in \mathcal{I} \times \mathcal{I} \middle| i \in \mathcal{J}^{(k)},\, j \in \mathcal{J}^{(l)},\, \mathrm{dist}(x_i, x_j) \leq \ln \tfrac{1}{\epsilon} \times 2^{-\min(k,l)} \right\}$$



A simple algorithm

- We order the degrees of freedom from $\mathcal{J}^{(1)}$ to $\mathcal{J}^{(q)}$ and define the sparsity pattern:

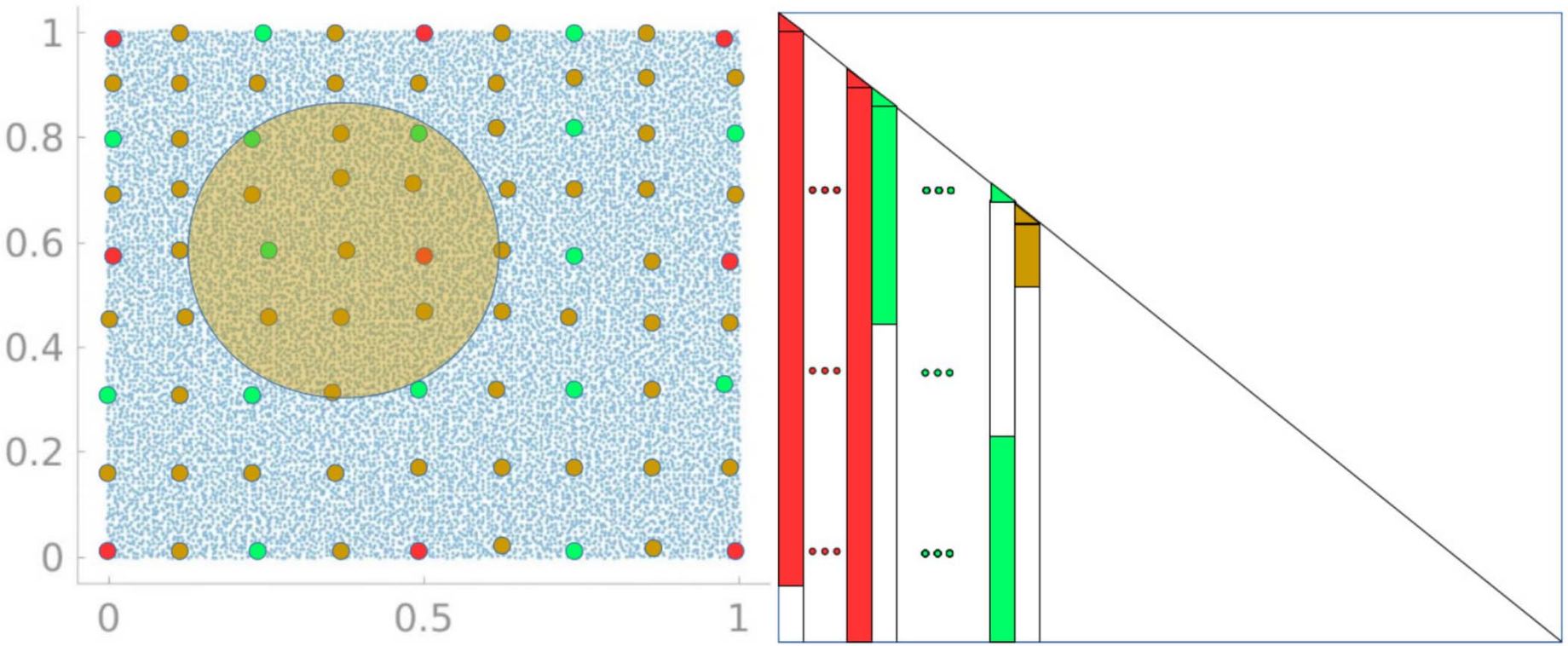
$$S := \left\{(i,j) \in \mathcal{I} \times \mathcal{I} \middle| i \in \mathcal{J}^{(k)},\, j \in \mathcal{J}^{(l)},\, \mathrm{dist}(x_i, x_j) \leq \ln \tfrac{1}{\epsilon} \times 2^{-\min(k,l)} \right\}$$



A simple algorithm

- We order the degrees of freedom from $\mathcal{J}^{(1)}$ to $\mathcal{J}^{(q)}$ and define the sparsity pattern:

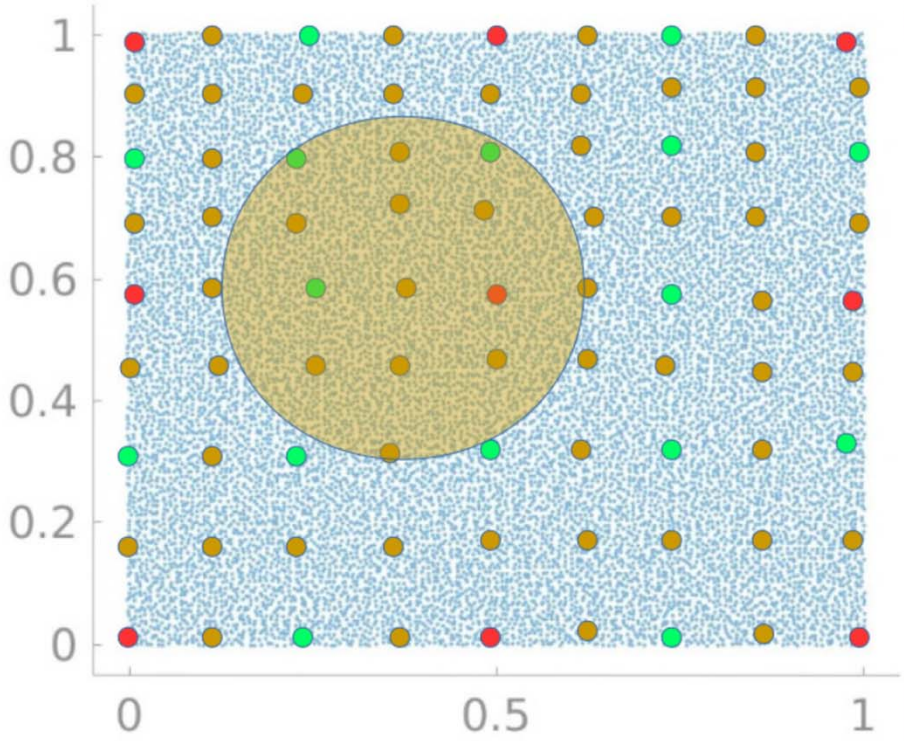
$$S := \left\{(i,j) \in \mathcal{I} \times \mathcal{I} \middle| i \in \mathcal{J}^{(k)},\, j \in \mathcal{J}^{(l)},\, \mathrm{dist}(x_i, x_j) \leq \ln \tfrac{1}{\epsilon} \times 2^{-\min(k,l)} \right\}$$

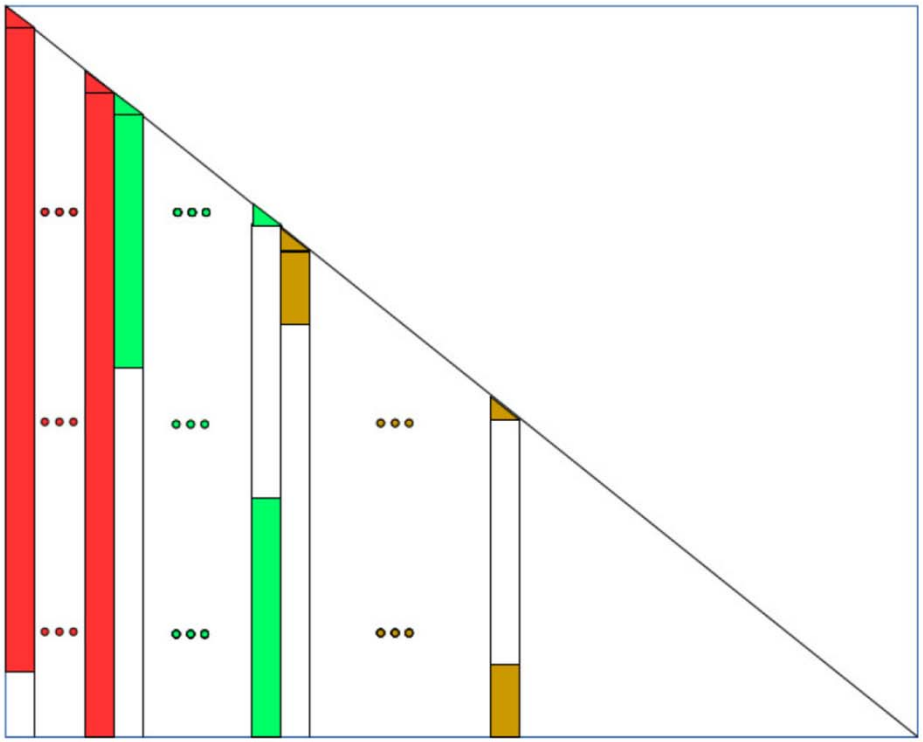


A simple algorithm

- We order the degrees of freedom from $\mathcal{J}^{(1)}$ to $\mathcal{J}^{(q)}$ and define the sparsity pattern:

$$S := \left\{(i,j) \in \mathcal{I} \times \mathcal{I} \middle| i \in \mathcal{J}^{(k)},\, j \in \mathcal{J}^{(l)},\, \mathrm{dist}(x_i, x_j) \leq \ln \tfrac{1}{\epsilon} \times 2^{-\min(k,l)} \right\}$$

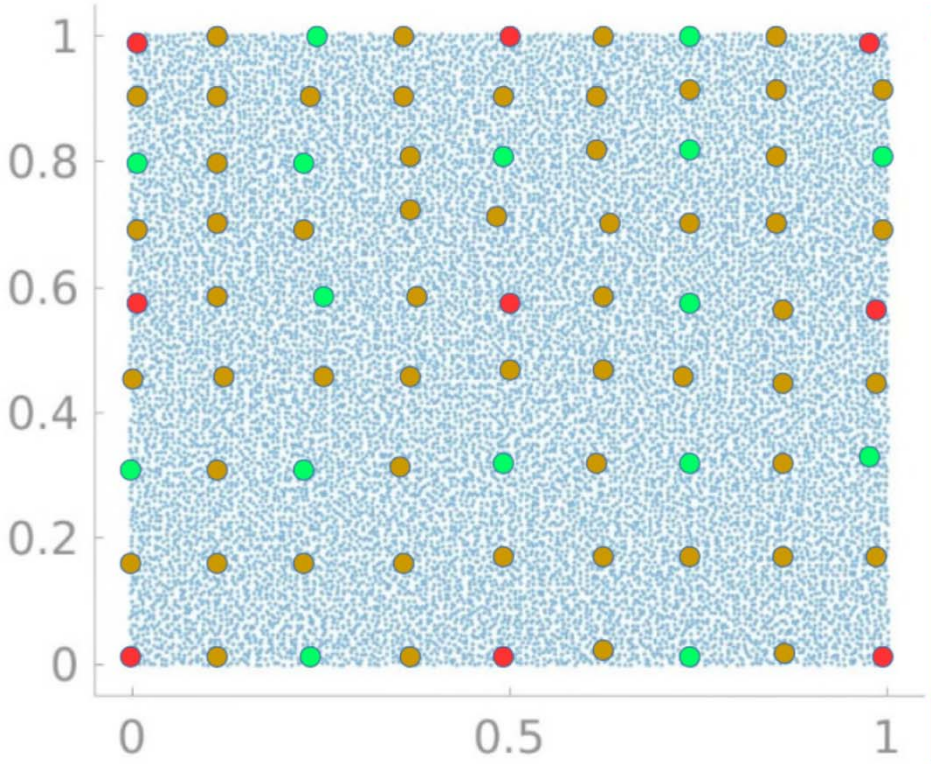


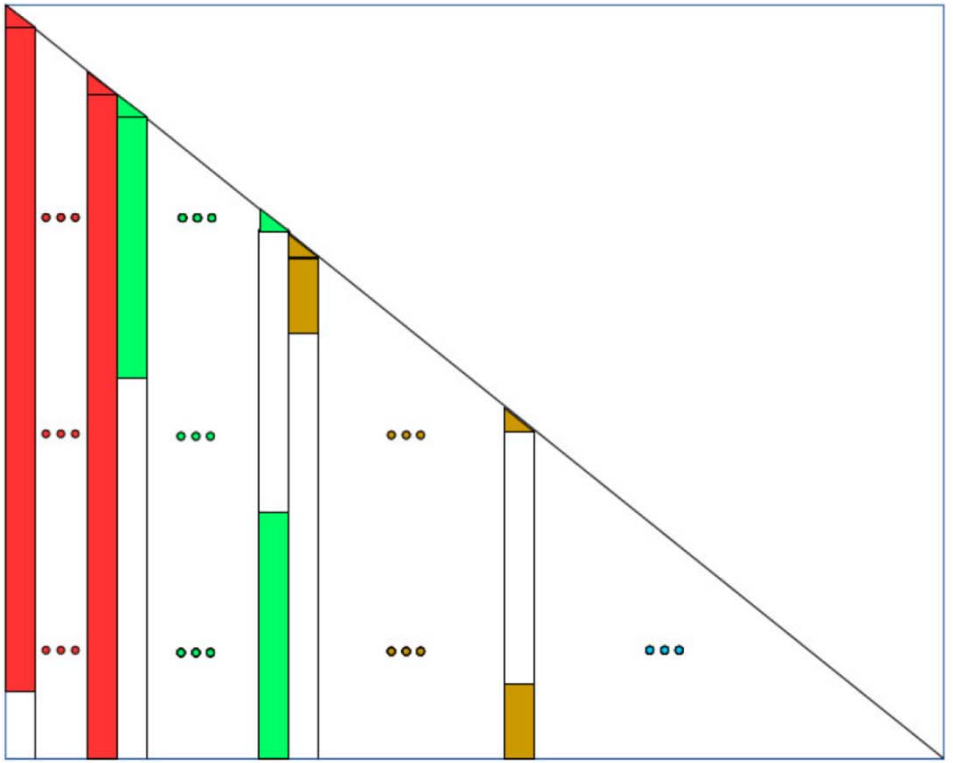


A simple algorithm

- We order the degrees of freedom from $\mathcal{J}^{(1)}$ to $\mathcal{J}^{(q)}$ and define the sparsity pattern:

$$S := \left\{(i,j) \in \mathcal{I} \times \mathcal{I} \middle| i \in \mathcal{J}^{(k)},\, j \in \mathcal{J}^{(l)},\, \mathrm{dist}(x_i, x_j) \leq \ln \tfrac{1}{\epsilon} \times 2^{-\min(k,l)} \right\}$$





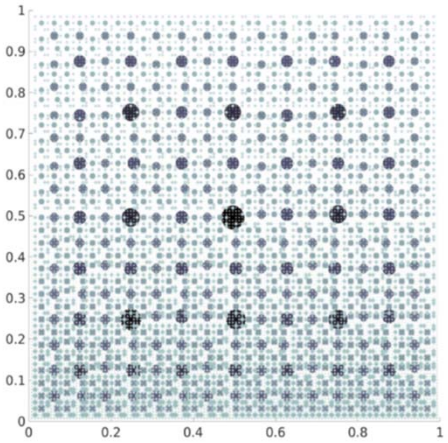
Complete vs Incomplete Cholesky factorization

The algorithm is oblivious to exact knowledge of the PDE and uses only the geometry of the discretisation.

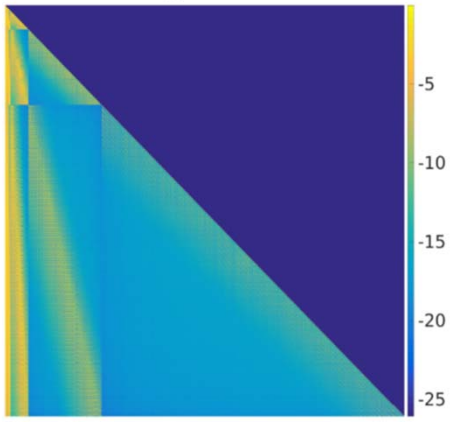
Why does it work?

Hierarchical ordering → Sparse Cholesky Factors

Hierarchical ordering



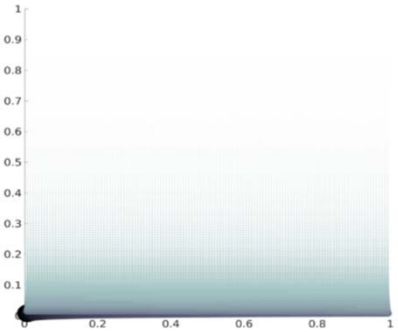
Ordering



L

Not true with the lexicographic ordering!

Lexicographic ordering



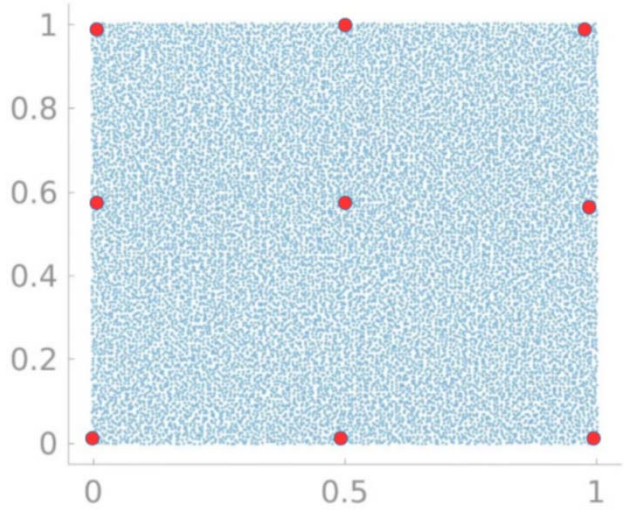
Ordering

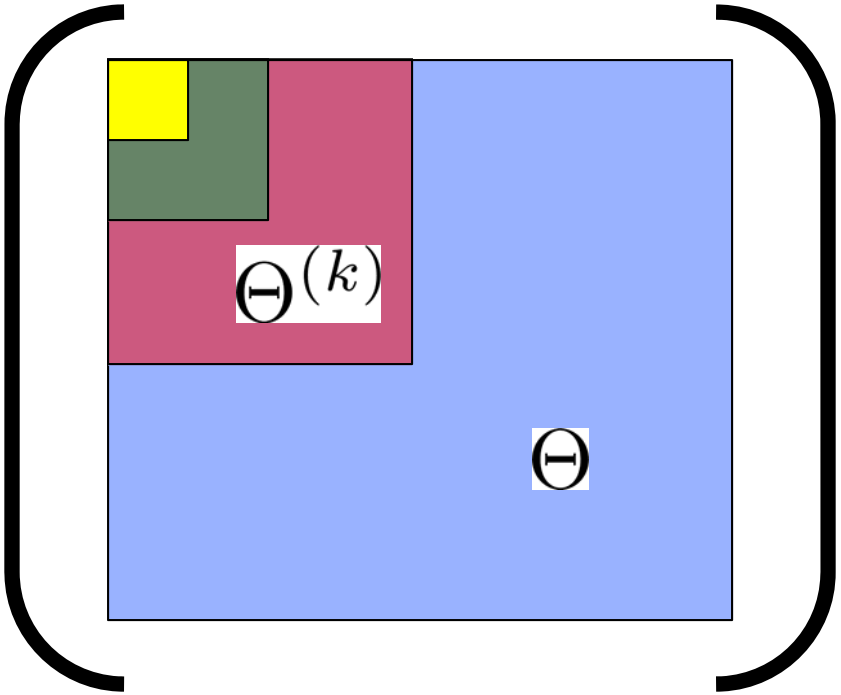
L

Why are the Cholesky factors sparse?

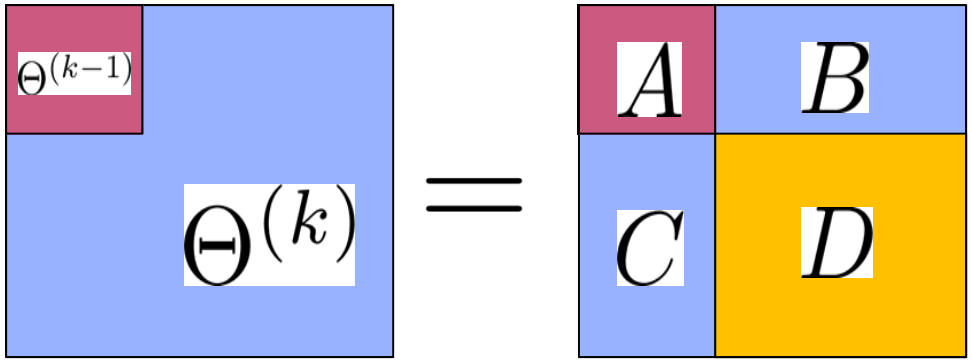
$$\Theta_{i,j}^{(k)} = G(x_i^{(k)}, x_j^{(k)})$$

$\Theta_{i,j}^{(k)}$: Sub-matrix of Θ





Single step of (Block-) Cholesky decomposition



$=$

$=$

$$\begin{array}{|c|c|}\hline I & 0 \\ \hline CA^{-1} & I \\ \hline \end{array} \qquad \begin{array}{|c|c|}\hline A & 0 \\ \hline 0 & D - CA^{-1}B \\ \hline \end{array} \qquad \begin{array}{|c|c|}\hline I & A^{-1}B \\ \hline 0 & I \\ \hline \end{array}$$

$S^{(k)}$

$S^{(k)} =$ Schur complement of the block $\Theta^{(k-1)}$

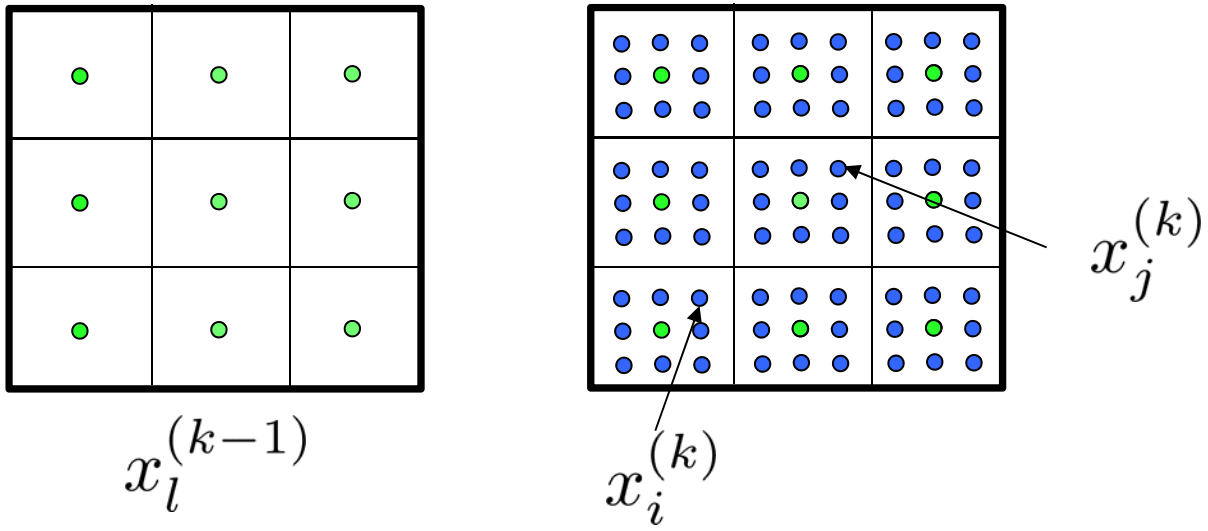
of the matrix $\Theta^{(k)}$

The Schur complement is sparse!

Why?

$$S_{i,j}^{(k)} = \mathrm{Cov}\left(\xi(x_i^{(k)}), \xi(x_j^{(k)}) \middle| \xi(x_l^{(k-1)}),\ \forall l\right)$$

$$\xi \sim \mathcal{N}(0, G)$$



Screening effect ⇒ $S^{(k)}$ is sparse

$$(B^{(k)})_{i,j}^{-1} = \mathrm{Cov}\left(\xi(x_i^{(k)}), \xi(x_j^{(k)}) \middle| \xi(x_l^{(k-1)}),\ \forall l\right)$$

$$B_{i,j}^{(k)} = \left\langle \chi_i^{(k)}, \chi_j^{(k)} \right\rangle$$

Gene Ryan Yoo

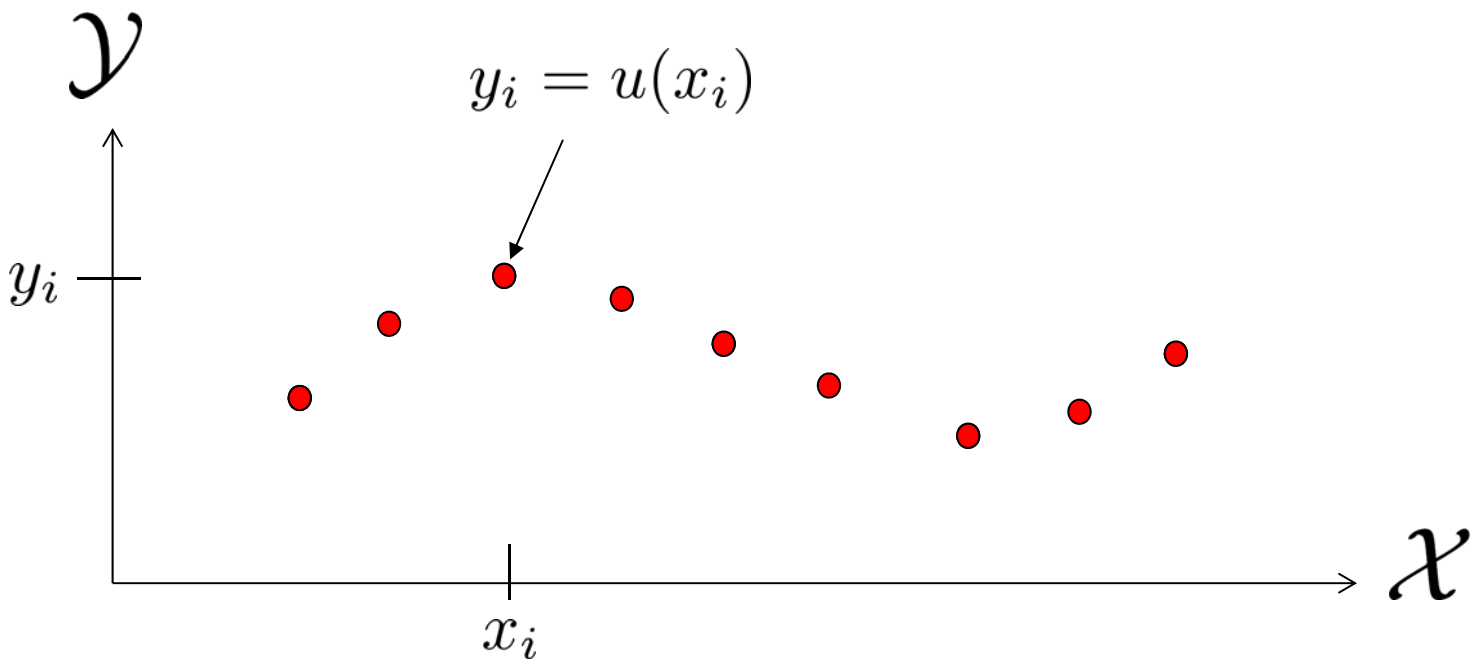
Learning as an interpolation problem

- Kernel Flows: from learning kernels from data into the abyss. H. Owhadi and G. R. Yoo, arXiv:1808.04475, 2018.

$$\mathcal{X} \xrightarrow{\quad u \quad} \mathcal{Y}$$

u : Unknown

Given $y_i = u(x_i)$ for $i = 1, \ldots, N$, approximate u

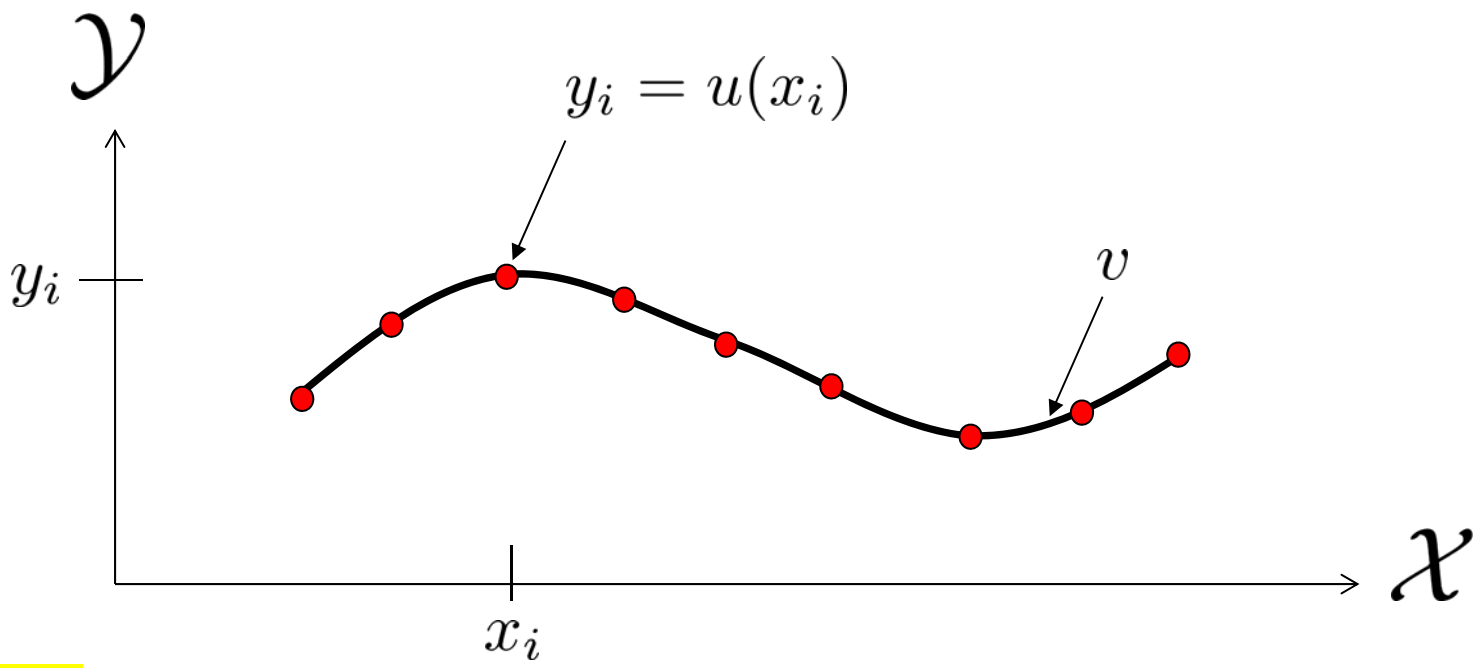


Solution: Kriging/GPR/SVM

Given kernel K approximate $u(x)$ with

$$v(x) = \sum_i c_i K(x_i, x)$$

c such that $v(x_i) = y_i$ for all i



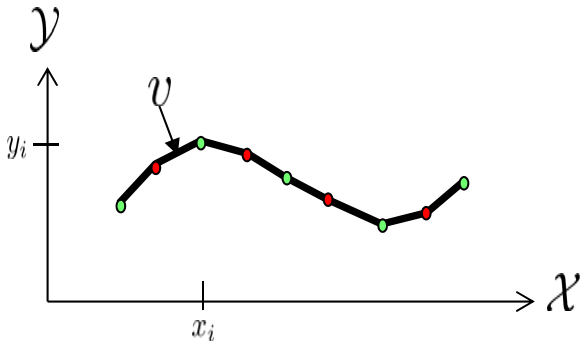
BUT

- **Which kernel do we pick?**

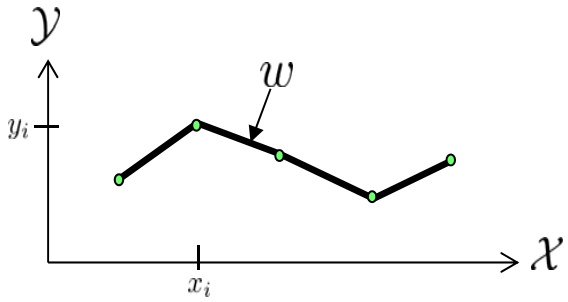
Premise

A kernel K is good if the number of interpolation points can be halved without significant loss in accuracy

v: Interpolate with K and N points



w: Interpolate with K and $N/2$ points



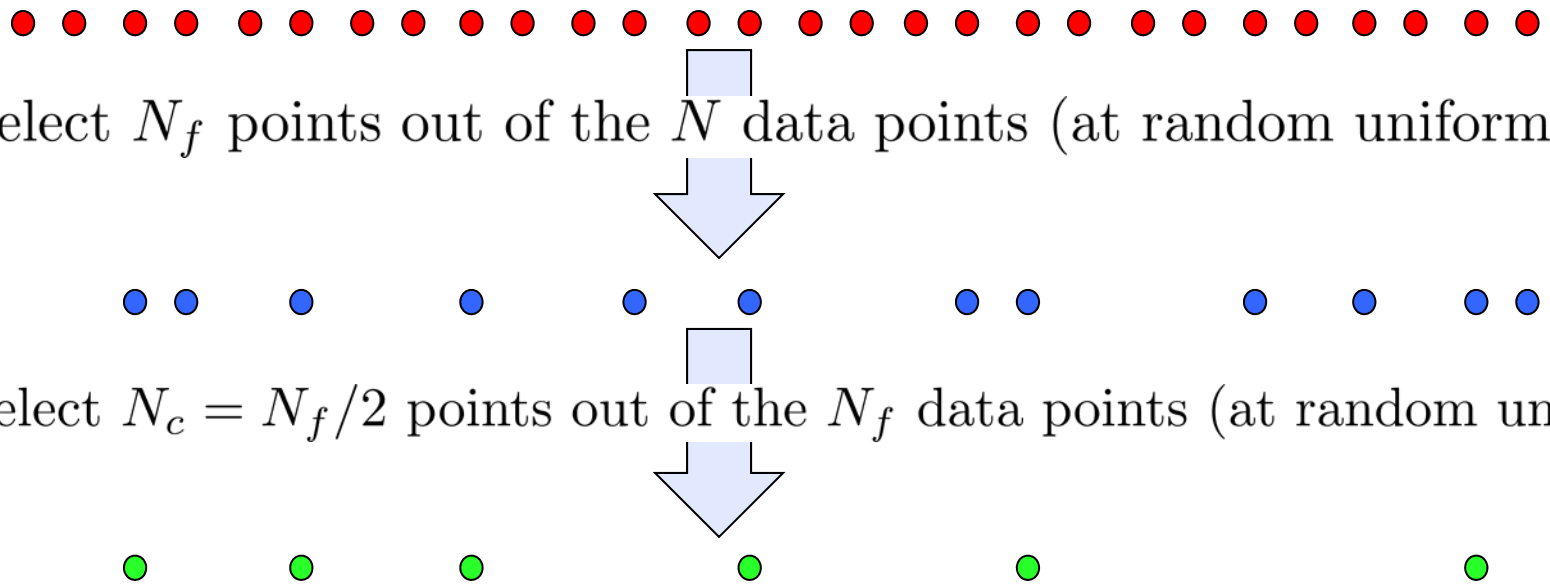
$$\rho = \frac{\|v-w\|^2}{\|v\|^2}$$

$$\|v\|^2 = \sup_\phi \frac{(\int \phi(x)v(x)\,dx)^2}{\int \phi(x)K(x,x')\phi(x')\,dx\,dx'}$$

Good kernel ⟺ Small ρ

Algorithm

Step $n \to n+1$ $K(\alpha)$: parametrized family of kernels



Select N_f points out of the N data points (at random uniformly)

Select $N_c = N_f/2$ points out of the N_f data points (at random uniformly)

v: Kriging with N_f points w: Kriging with N_c points

$$\rho = \frac{\|v-w\|^2}{\|v\|^2}$$

$$\alpha \to \alpha - \epsilon \nabla_\alpha \rho(\alpha)$$

$\nabla_\alpha \rho(\alpha)$: Computable using the representer theorem

A simple example

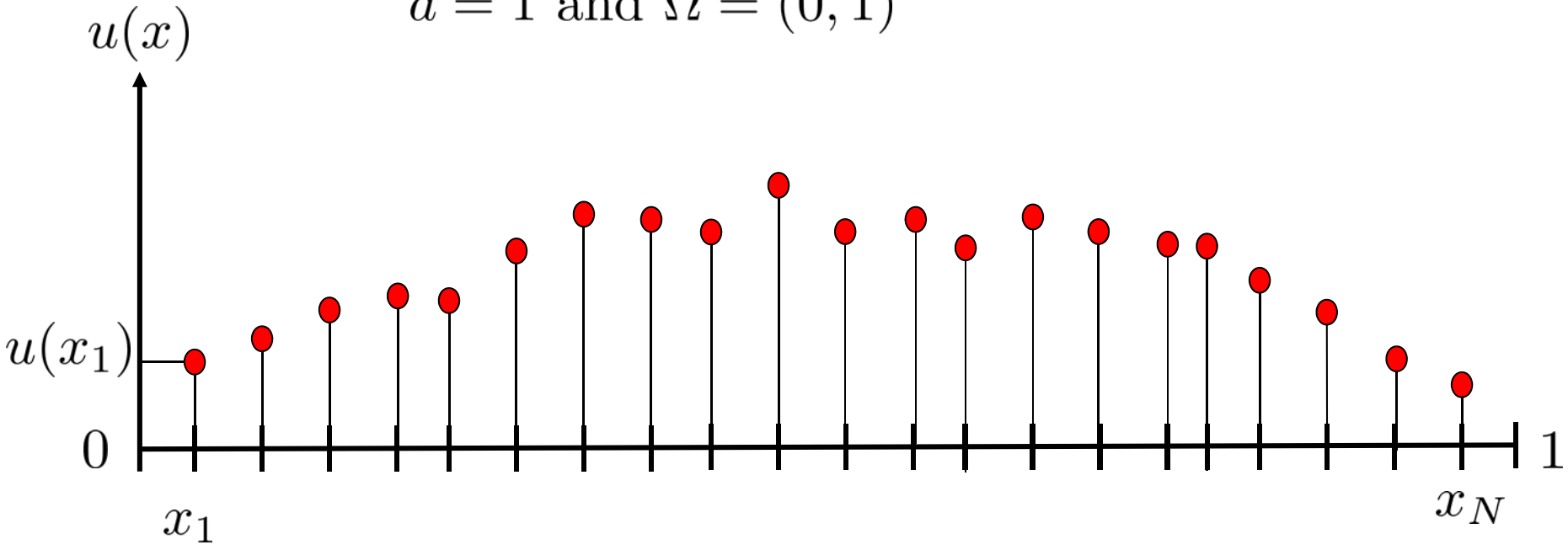
$$\begin{cases} -\operatorname{div}(a\nabla u) = f, & x \in \Omega, \\ u = 0, & x \in \partial\Omega, \end{cases} \qquad f \in L^2(\Omega)$$

a, u, f: unknown

You see $(y_i = u(x_i))_{1 \le i \le N}$ You know $\|f\|_{L^2} \le 1$

You want to recover u and a

$d = 1$ and $\Omega = (0, 1)$



Solution

(1) $\begin{cases} -\operatorname{div}(\alpha \nabla u) = f, & x \in \Omega, \\ u = 0, & x \in \partial\Omega, \end{cases}$ G_α: Green's function of (1)

Approximate u with

$$v(x) := \sum_{i=1}^{N} c_i G_\alpha(x, x_i)$$

c such that $v(x_i) = y_i$ for all i

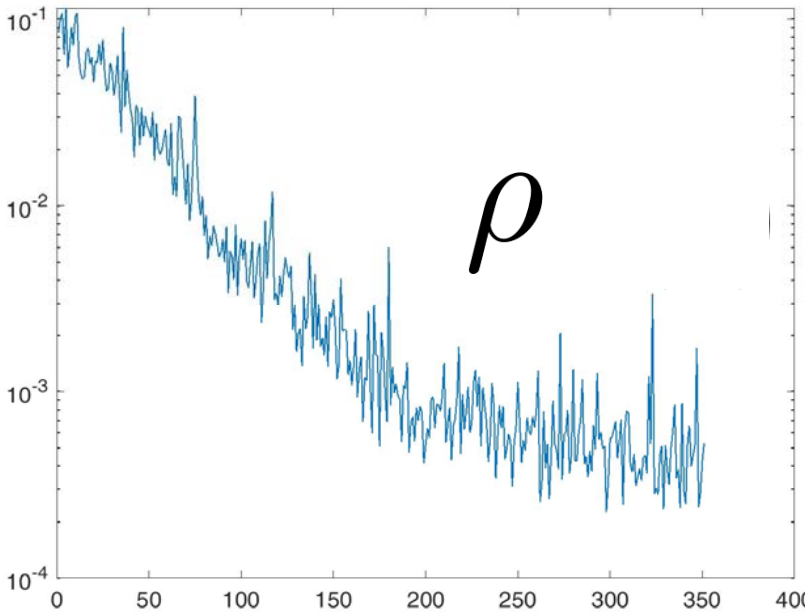
Evolve α in the gradient descent direction of ρ

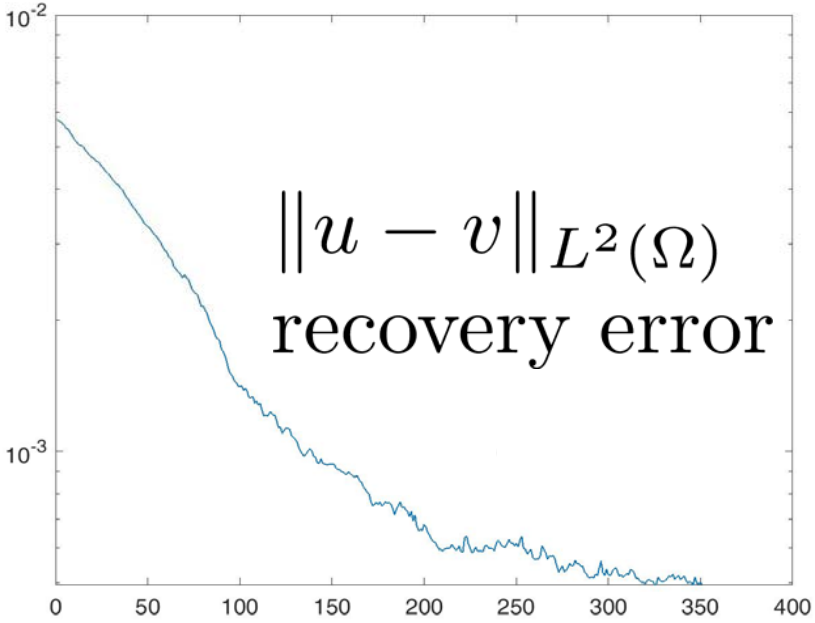
$$\alpha \to \alpha - \epsilon \nabla_\alpha \rho(\alpha)$$

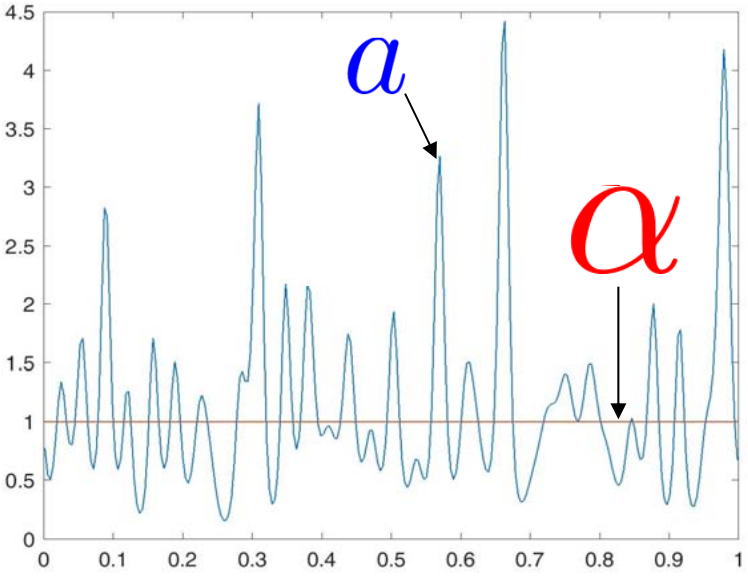
$$\rho = \frac{\|v - w\|_\alpha^2}{\|v\|_\alpha^2} \qquad \|v\|_\alpha^2 = \int_\Omega (\nabla v)^T \alpha \nabla v$$

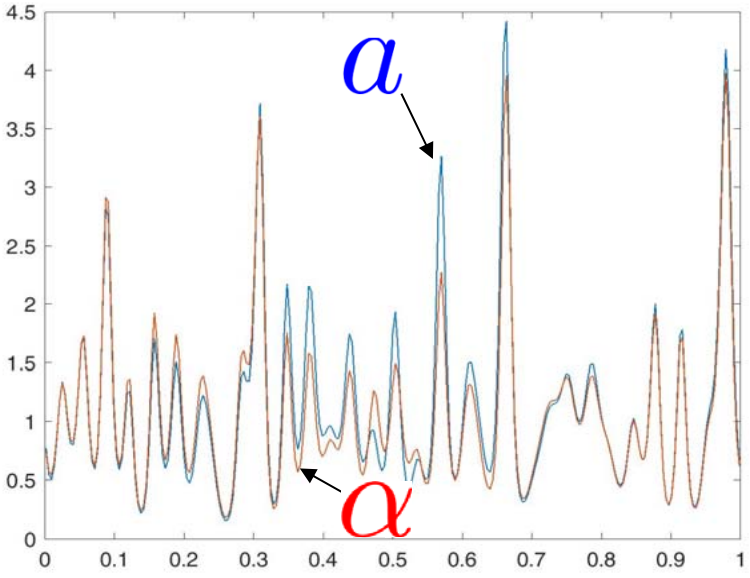
w: Interpolant computed with G_α and $N/2$ points (selected at radom)

Implementation of the algorithm









Non parametric version (kernel flows)

Learns kernels of the form

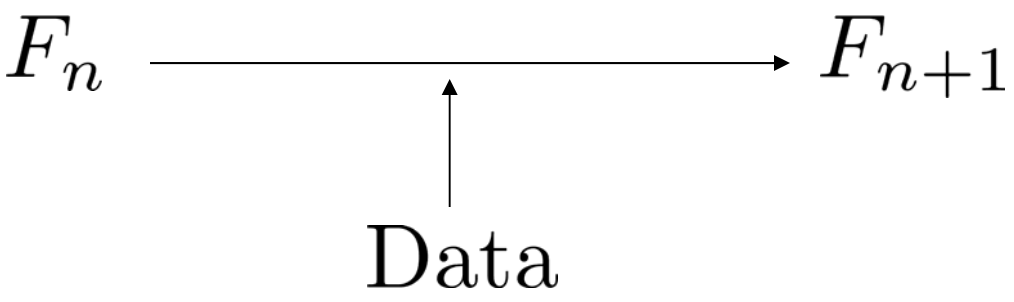
$$K_n(x, x') = K_1(F_n(x), F_n(x'))$$

K_1: kernel (e.g. $K_1(x, x') = e^{-\frac{|x-x'|^2}{\gamma^2}}$)

F_n : Flow in input space

$$F_n : \mathcal{X} \to \mathcal{X}$$

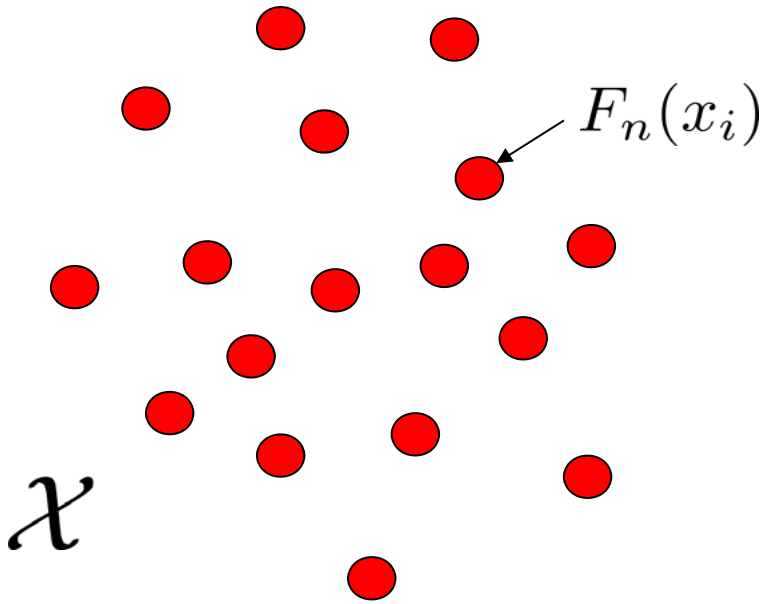
$$F_1 = I_d$$



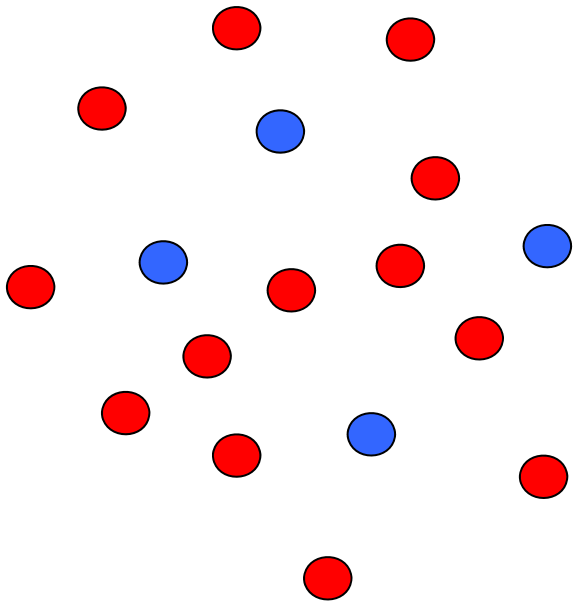
Step $n \to n+1$

Assume F_n known

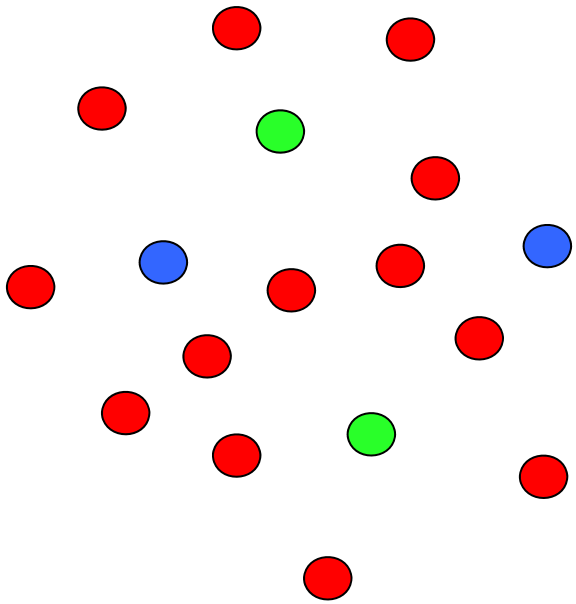
Images of the N training points under F_n



Select N_f at random out of N



Select $N_f/2$ at random out of N_f

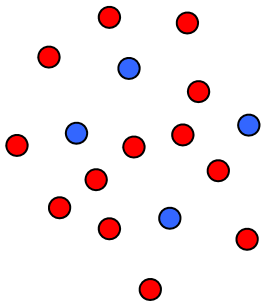


Player I

Selects the values/labels of the blue points $F_n(x_i)$ to be y_i (training labels)

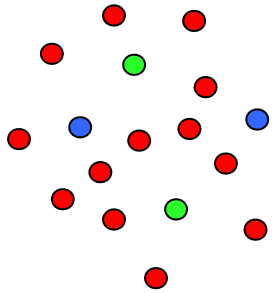
Player II

Sees values/labels y_i of the $N_c = N_f/2$ green points must predict the values of the blue points



Max

Min

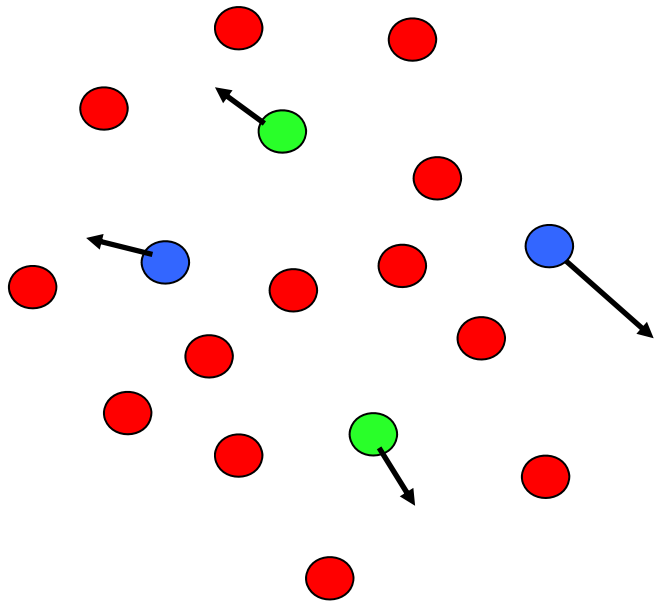


ρ

ρ: Relative error in $\|\cdot\|$ norm

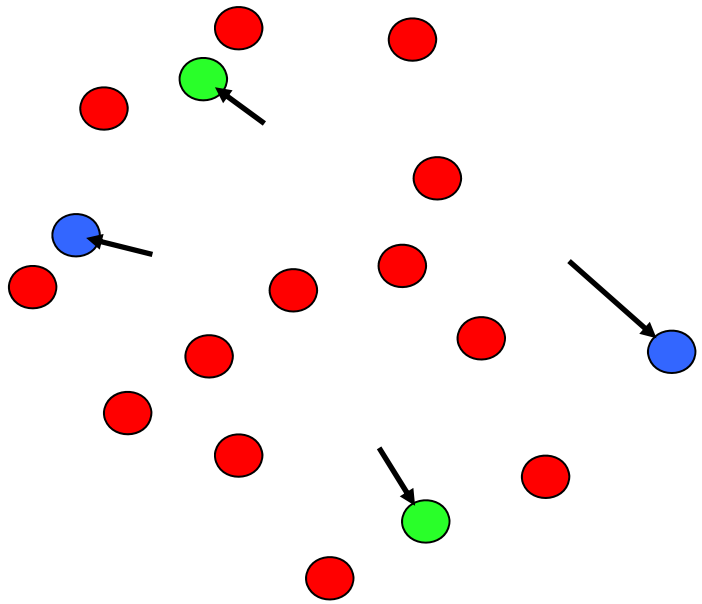
$\|\cdot\|$: RKHS norm associated with K_1

Move the N_f points in the gradient descent direction of ρ

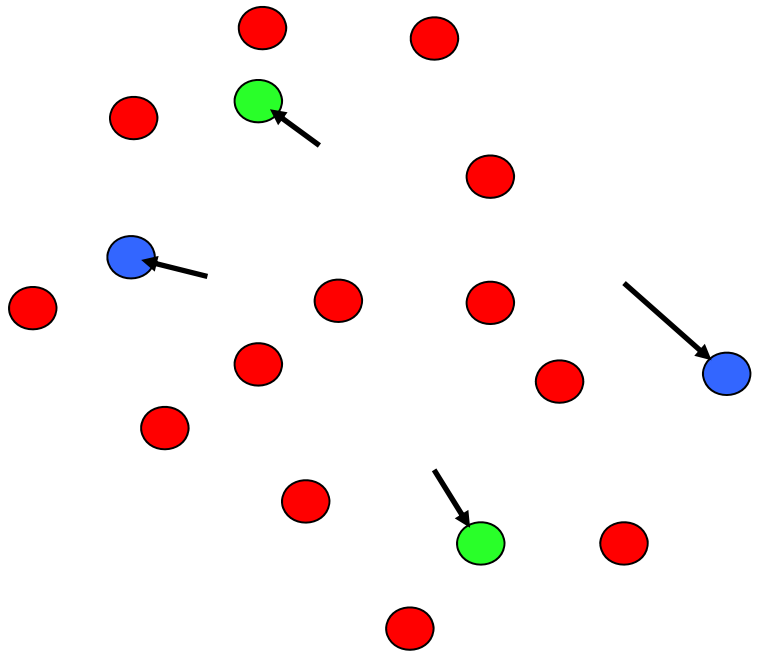


Rig the game in favor of Player II

Move the N_f points in the gradient descent direction of ρ



Move the remaining $N - N_f$ points
via interpolation with kernel K_1



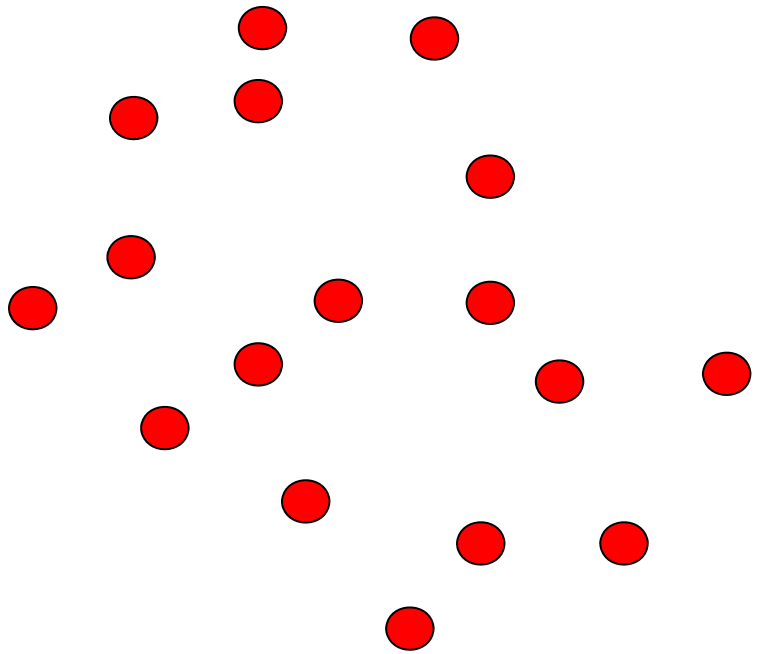
Move any point x
via interpolation with kernel K_1

F_{n+1}

Repeat

F_{n+1} known

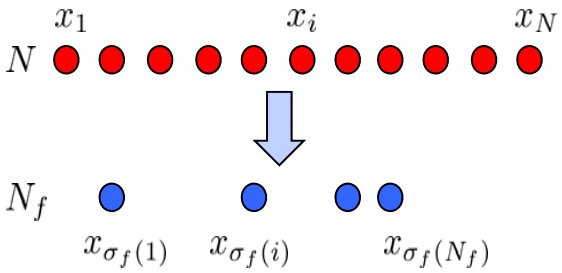
Images of the N training points under F_{n+1}

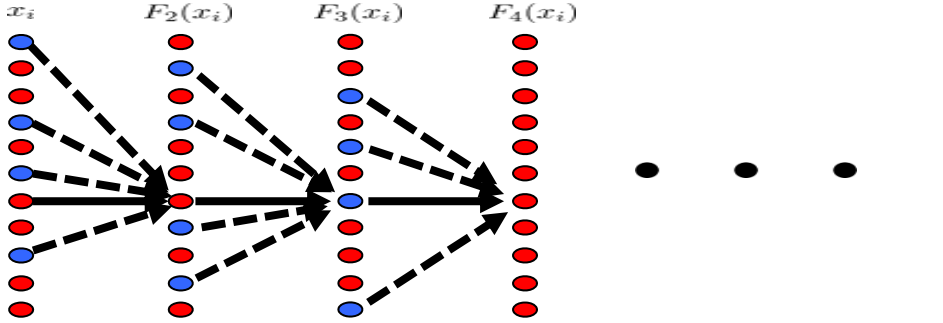


Kernel Flow

Produces a deep hierarchical kernel of the form

$$K_n(x,x') = K_{n-1}(x + \epsilon \sum_{i=1}^{N_f} c_i K_{n-1}(x_{\sigma_f(i)}, x), x' + \epsilon \sum_{i=1}^{N_f} c_i K_{n-1}(x_{\sigma_f(i)}, x'))$$





Randomized bottomless network

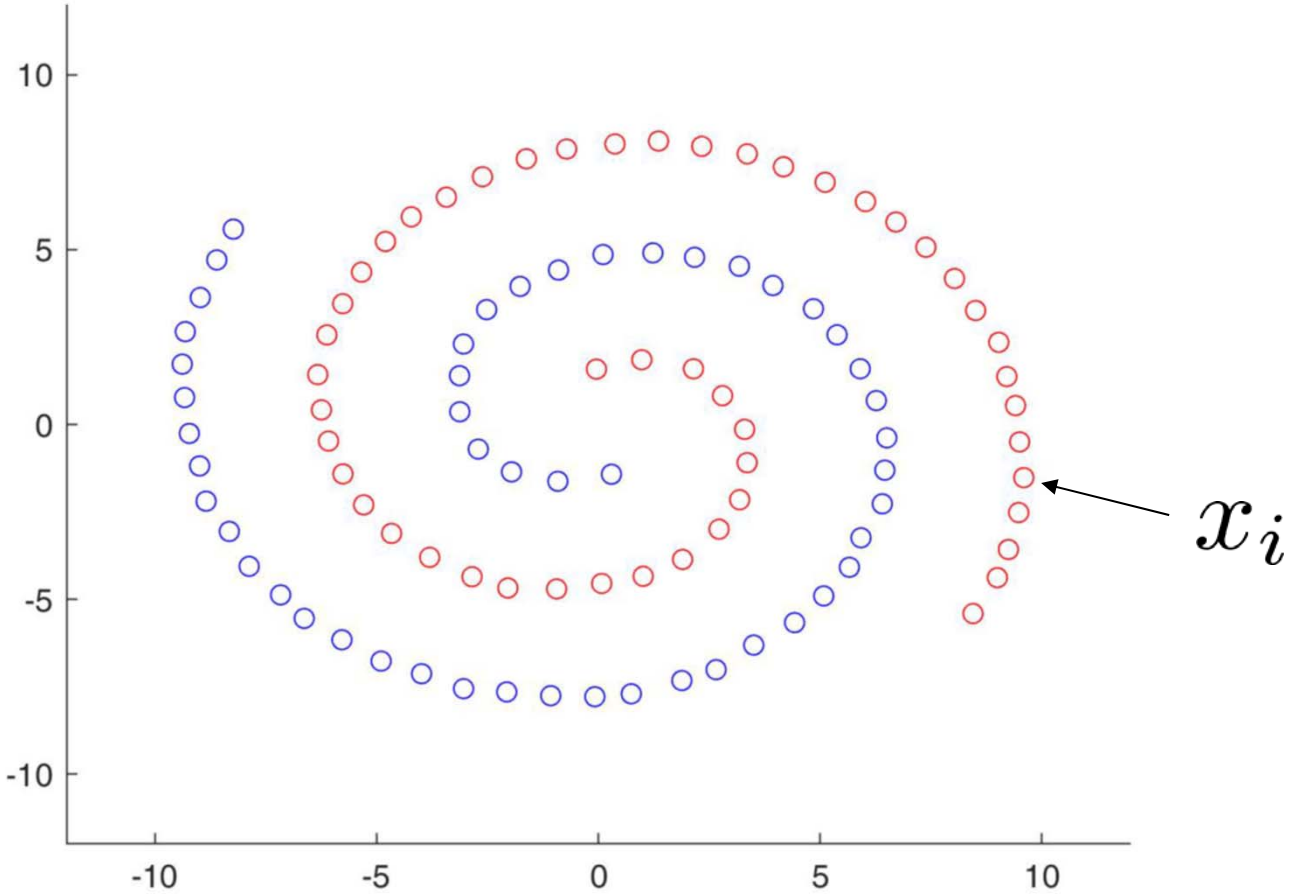
and a flow of the form

$$F_{n+1} = (I_d + \epsilon G_{n+1}) \circ F_n$$

$$G_{n+1}(x) = \sum_{i=1}^{N_f} c_i K_1(F_n(x_{\sigma_f(i)}), x)$$

Identified as the steepest gradient descent direction of ρ.

Application: Swiss Roll Cheesecake



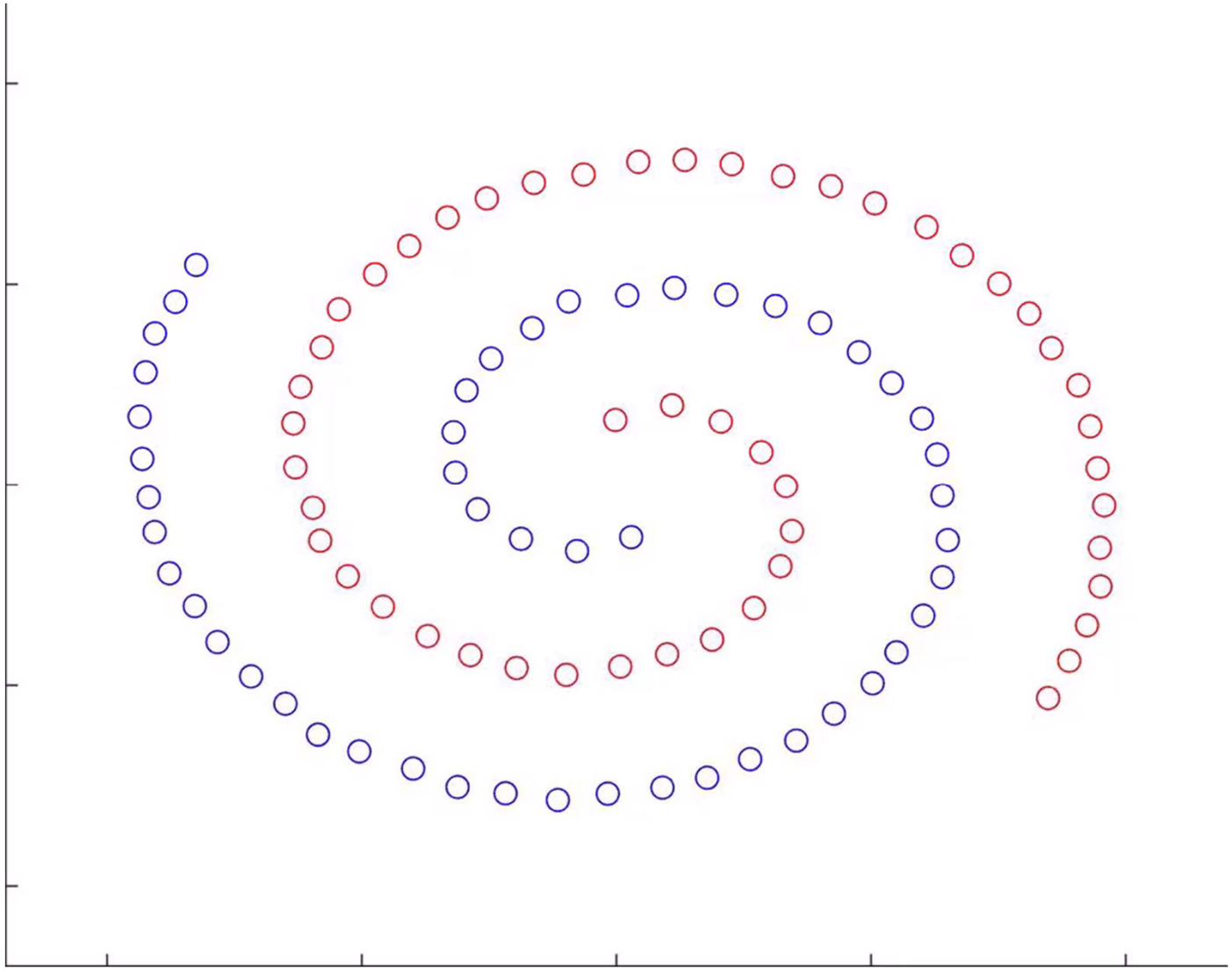
$N = 100$ data points $x_i \in \mathbb{R}^2$

$y_i = +1$ if point at x_i is red

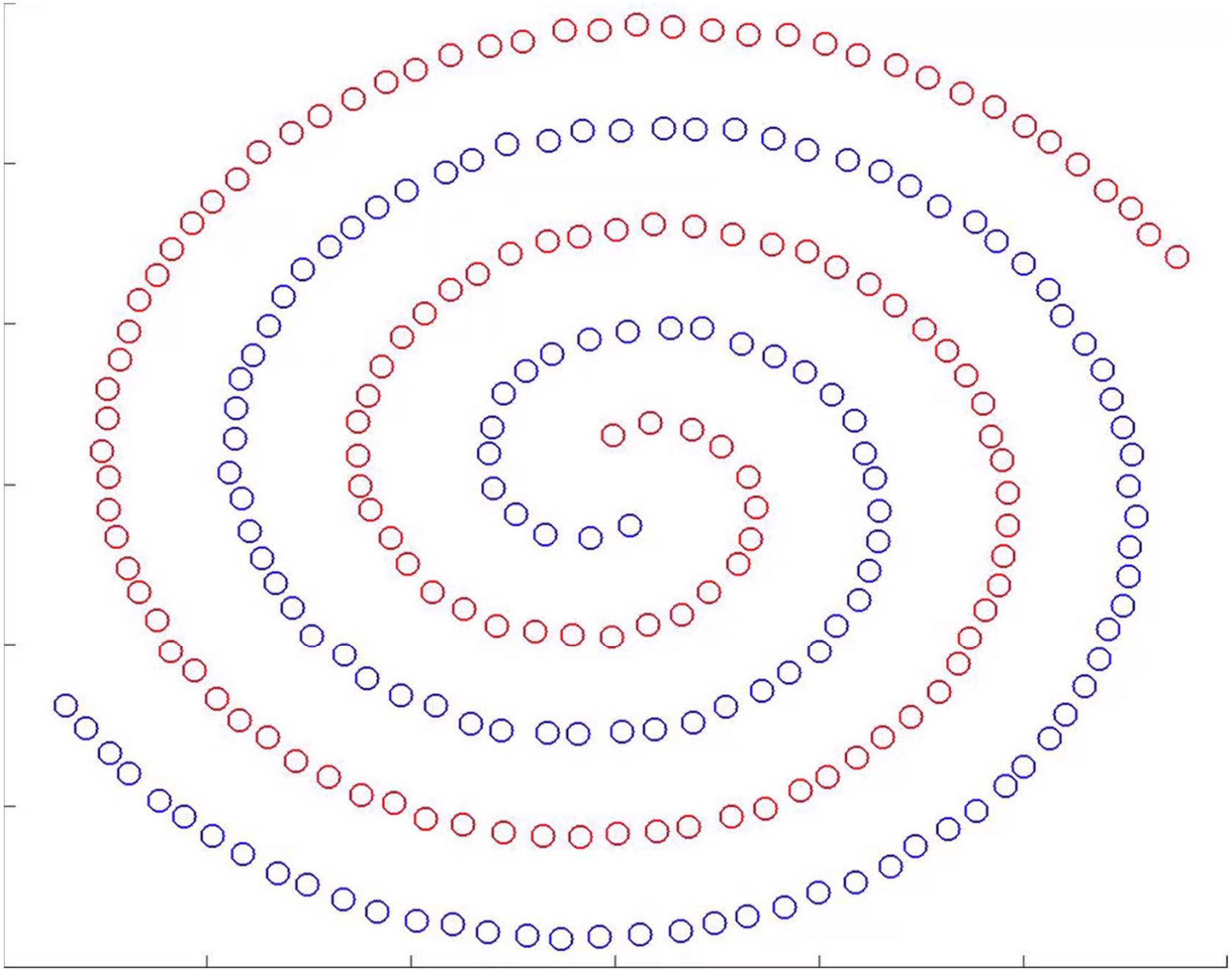
$y_i = -1$ if point at x_i is blue

Objective:
Visualize $n \to F_n(x_i)$

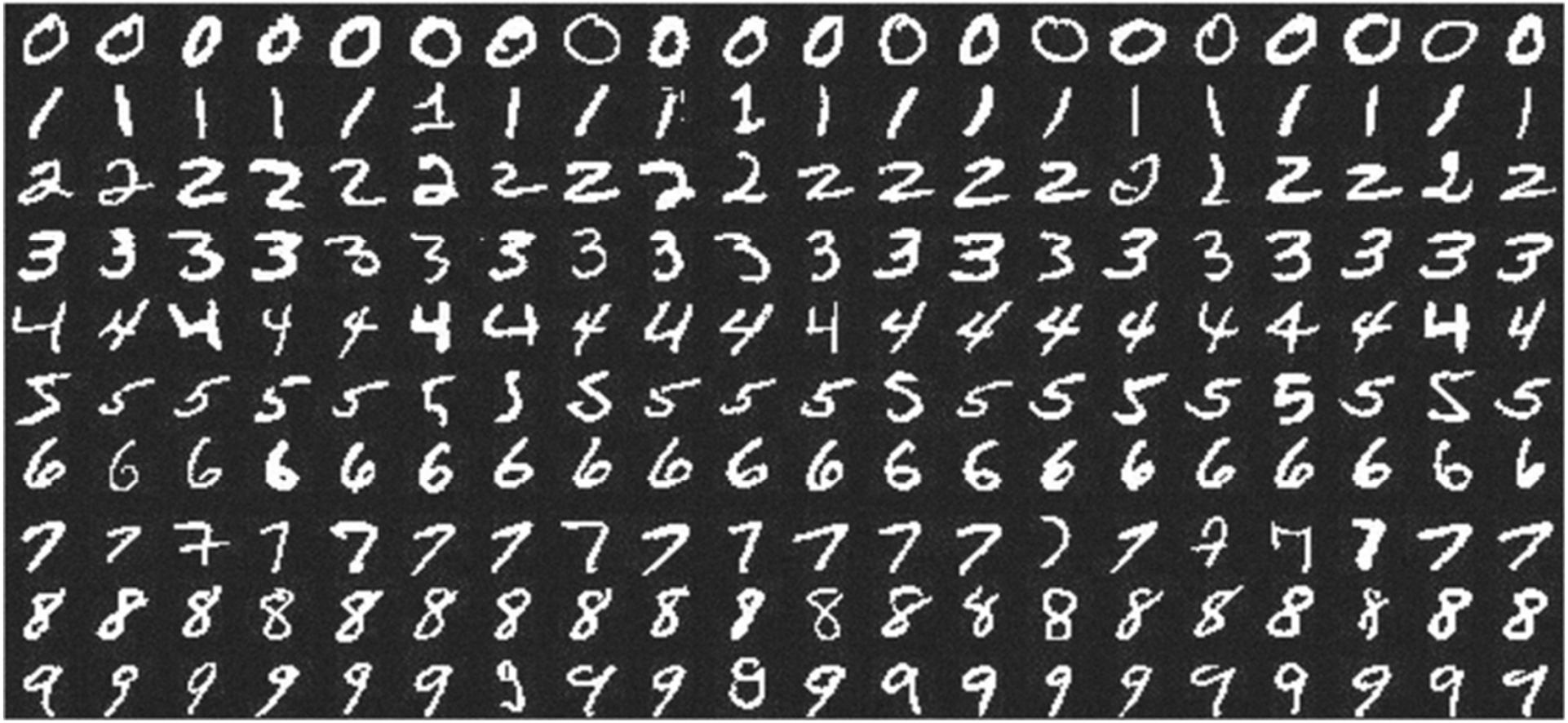
$F_n(x_i)$ Gaussian Kernel, $N_f = N$



$F_n(x_i)$ Gaussian Kernel, $N_f = N$, large ϵ



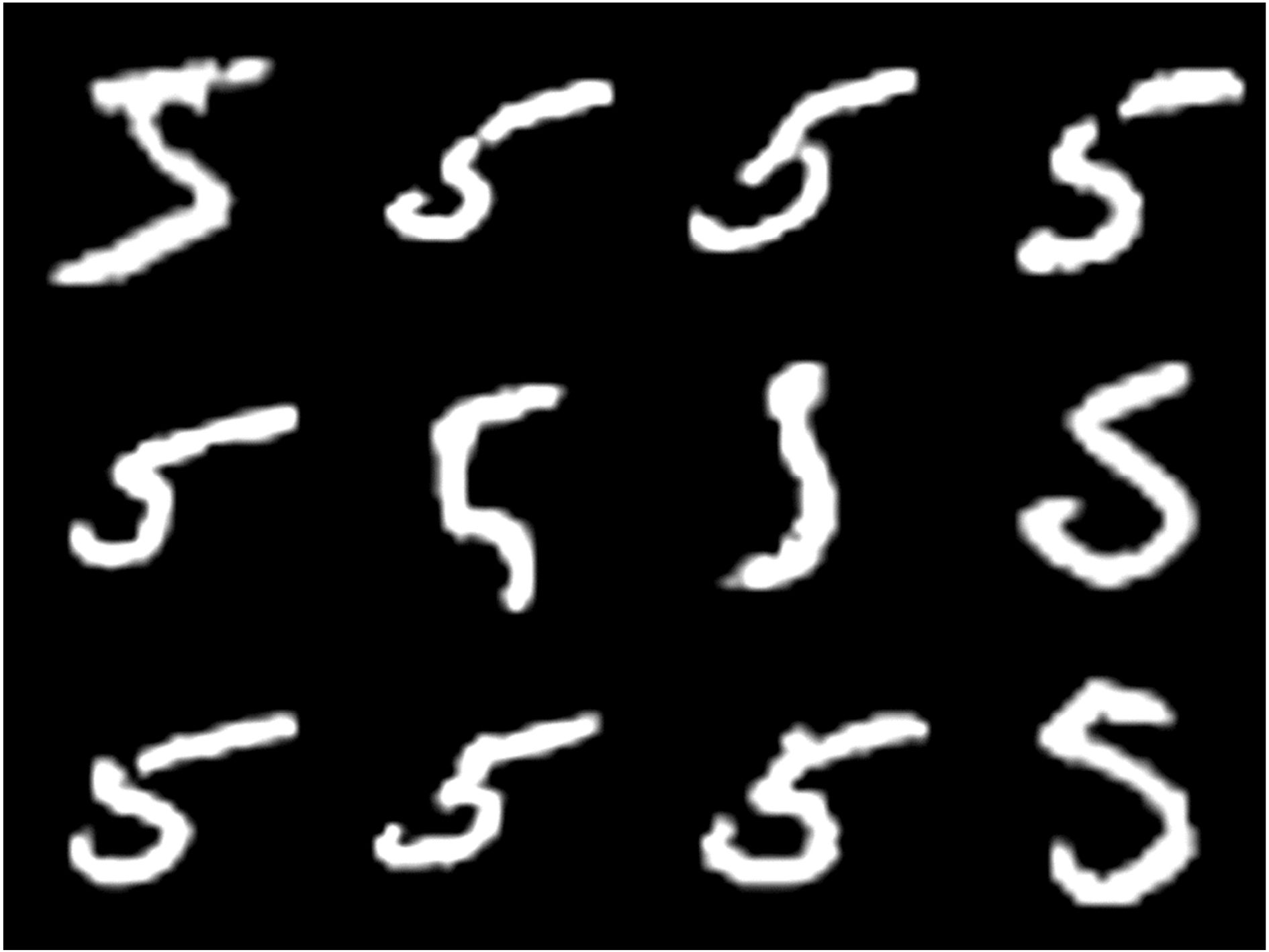
Application to MNIST



$N = 60000$

$N_f = 600$

12000 layers



Application to Fashion-MNIST

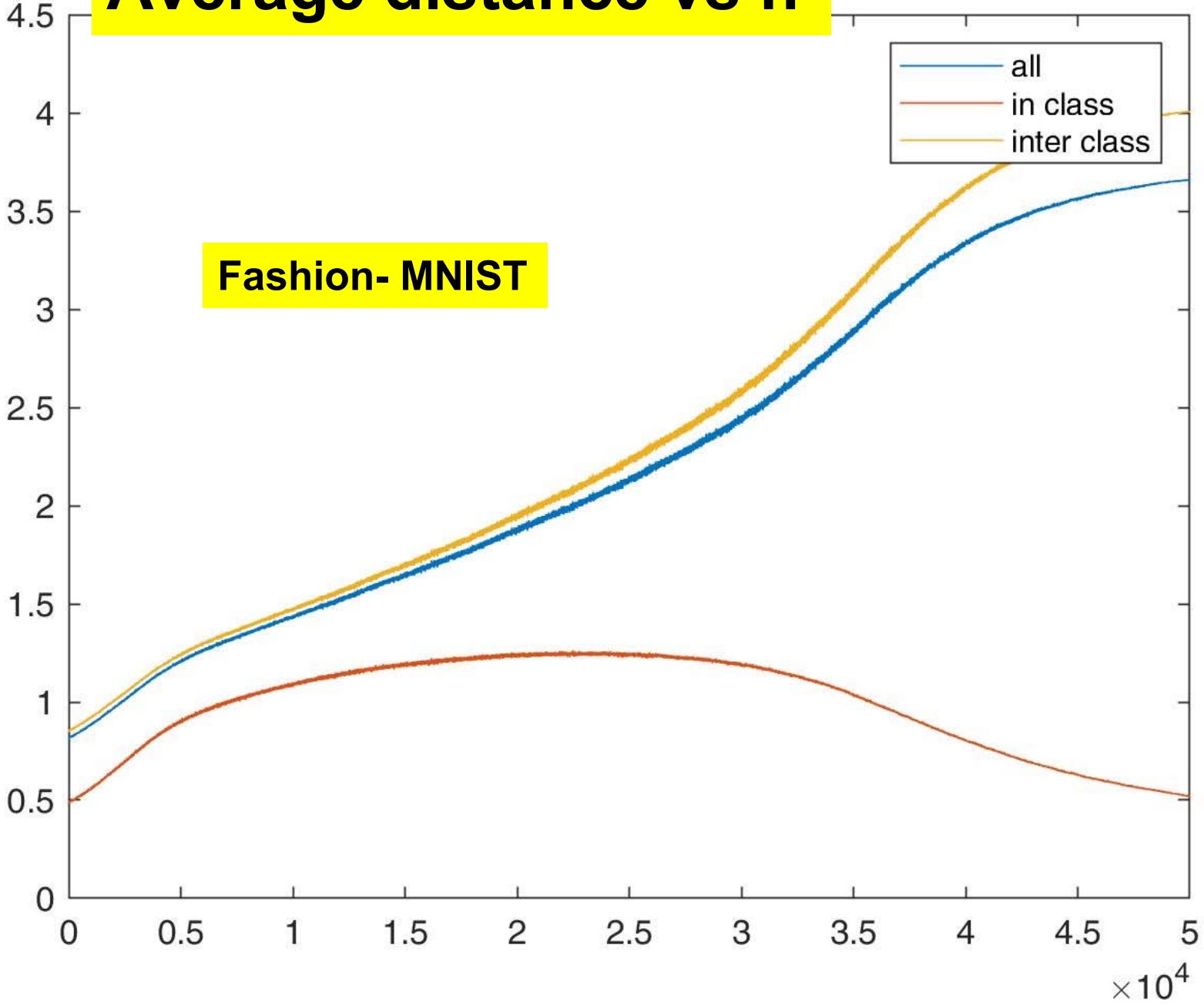
$$N = 60000$$

$$N_f = 600$$

12000 layers, large steps

50000 layers, small steps

Average distance vs n

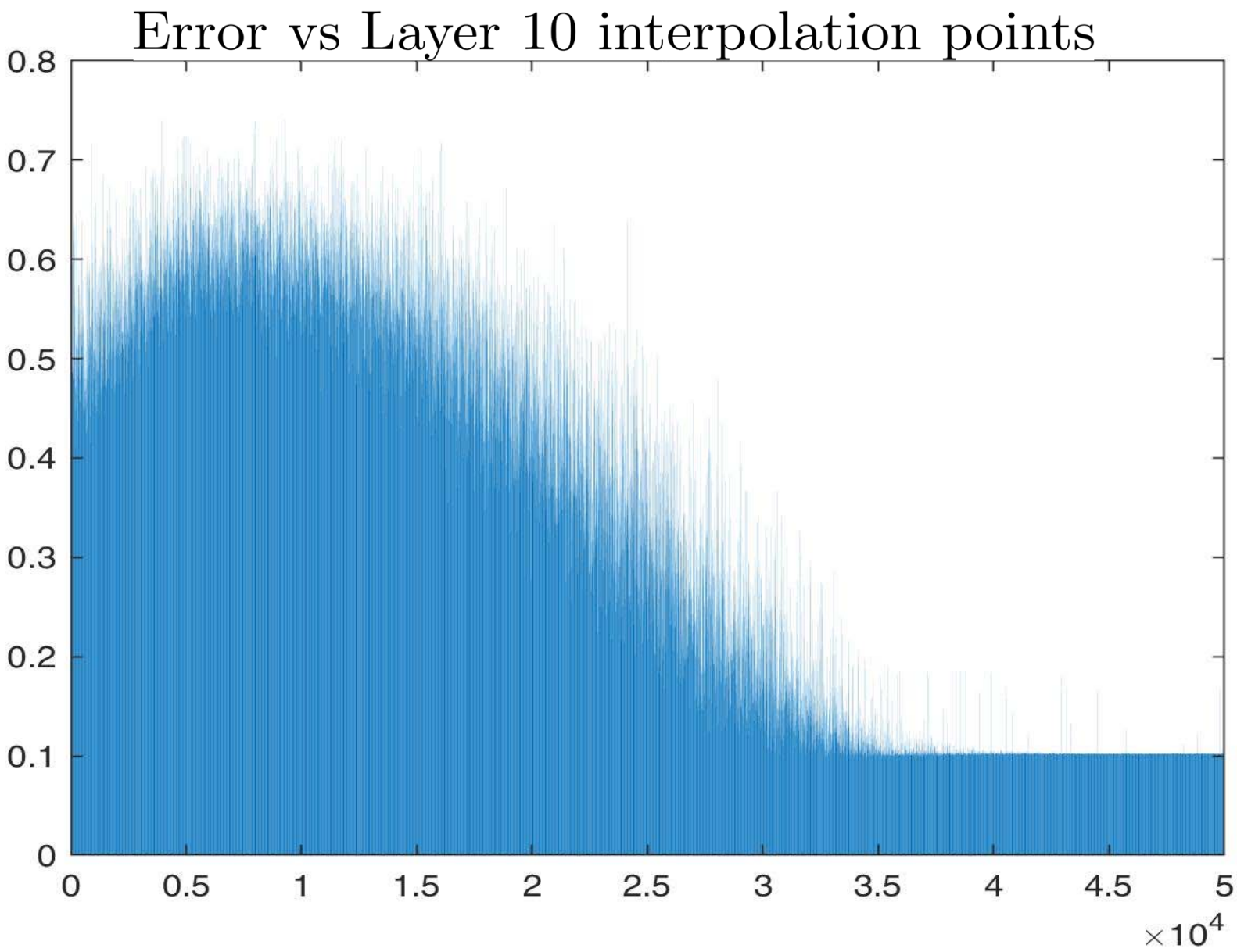


Classify 10000 test points

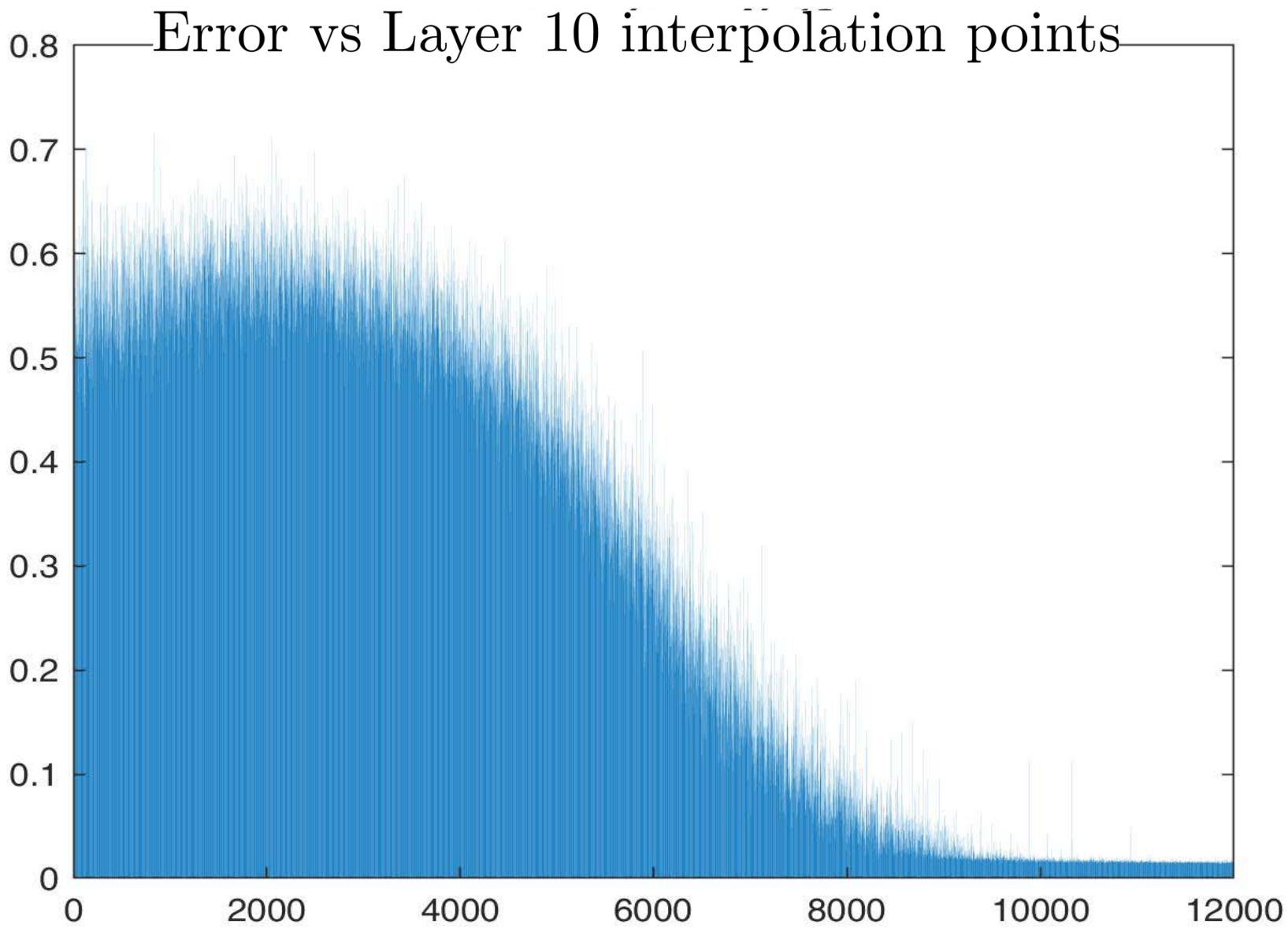
Use kernel K_n
and N_I interpolation points
selected at random

$N_I = 10 \iff$ Interpolate with only 1 point per class

Fashion-MNIST Test Error vs layer



MNIST Test Error vs layer



Thank you