A Sparse Approximation Perspective on Operator Learning for Differential Equations

Fabian Taubert

joint work with Lutz Kämmerer, Sebastian Neumayer and Daniel Potts

Session: Sparse and high-dimensional approximation (2)

11.06.2026

UNIVERSITY OF TECHNOLOGY
IN THE EUROPEAN CAPITAL OF CULTURE
CHEMNITZ

Differential problem

$$Lu = f$$

s.t. $u|_{\partial\Omega\times(0,T)} = h$ and $u(\cdot,0) = u_0$

- differential operator $L : \mathcal{U} \to \mathcal{F}$
- inhomogeneity $f \in \mathcal{F}$, initial value $u_0 \in \mathcal{U}_0$, boundary condition $h \in \mathcal{H}$
- **solution operator:**
  $G : \mathcal{F} \times \mathcal{H} \times \mathcal{U}_0 \to \mathcal{U}, \quad (f, h, u_0) \mapsto u$

Parametrization of u_0

$$u_0(\boldsymbol{x}) \approx \sum_{j=1}^{n} a_j A_j(\boldsymbol{x}), \qquad \boldsymbol{x} \in \Omega$$

- fixed functions $A_j, j = 1, \ldots, n$ (B-splines, trig. polynomials, ...)
- parameters $\boldsymbol{a} = (a_1, \ldots, a_n) \in [-1, 1]^n$
- identify u_0 by these parameters $\boldsymbol{a}$

Discretization of the solution operator G

- In practice: Only a subset of the data (f, h, u_0) is varied. $\to$ Today: f and h fixed, $G : \mathcal{U}_0 \to \mathcal{U}$.
- Parametrization yields induced solution operator $G_n : [-1, 1]^n \to \mathcal{U}$ with $\boldsymbol{a} \mapsto u$.

Differential problem

$$Lu = f$$

s.t. $u|_{\partial\Omega\times(0,T)} = h$ and $u(\cdot,0) = u_0$

- differential operator $L : \mathcal{U} \to \mathcal{F}$
- inhomogeneity $f \in \mathcal{F}$, initial value $u_0 \in \mathcal{U}_0$, boundary condition $h \in \mathcal{H}$
- **solution operator:**
  $G : \mathcal{F} \times \mathcal{H} \times \mathcal{U}_0 \to \mathcal{U}, \quad (f, h, u_0) \mapsto u$

Parametrization of u_0

$$u_0(\boldsymbol{x}) \approx \sum_{j=1}^{n} a_j A_j(\boldsymbol{x}), \qquad \boldsymbol{x} \in \Omega$$

- fixed functions $A_j, j = 1, \ldots, n$ (B-splines, trig. polynomials, ...)
- parameters $\boldsymbol{a} = (a_1, \ldots, a_n) \in [-1, 1]^n$
- identify u_0 by these parameters $\boldsymbol{a}$

Discretization of the solution operator G

- In practice: Only a subset of the data (f, h, u_0) is varied. $\to$ Today: f and h fixed, $G : \mathcal{U}_0 \to \mathcal{U}$.
- Parametrization yields induced solution operator $G_n : [-1, 1]^n \to \mathcal{U}$ with $\boldsymbol{a} \mapsto u$.

Differential problem

$$Lu = f$$

$$\text{s.t. } u|_{\partial\Omega\times(0,T)} = h \text{ and } u(\cdot,0) = u_0$$

- differential operator $L : \mathcal{U} \to \mathcal{F}$
- inhomogeneity $f \in \mathcal{F}$, initial value $u_0 \in \mathcal{U}_0$, boundary condition $h \in \mathcal{H}$
- **solution operator:**

$$G : \mathcal{F} \times \mathcal{H} \times \mathcal{U}_0 \to \mathcal{U}, \quad (f, h, u_0) \mapsto u$$

Parametrization of u_0

$$u_0(\boldsymbol{x}) \approx \sum_{j=1}^{n} a_j A_j(\boldsymbol{x}), \qquad \boldsymbol{x} \in \Omega$$

- fixed functions $A_j, j = 1, \ldots, n$ (B-splines, trig. polynomials, ...)
- parameters $\boldsymbol{a} = (a_1, \ldots, a_n) \in [-1,1]^n$
- identify u_0 by these parameters $\boldsymbol{a}$

Discretization of the solution operator G

- In practice: Only a subset of the data (f, h, u_0) is varied. $\to$ Today: f and h fixed, $G : \mathcal{U}_0 \to \mathcal{U}$.
- Parametrization yields induced solution operator $G_n : [-1,1]^n \to \mathcal{U}$ with $\boldsymbol{a} \mapsto u$.

Differential problem

$$Lu = f$$
$$\text{s.t. } u|_{\partial\Omega\times(0,T)} = h \text{ and } u(\cdot,0) = u_0$$

- differential operator $L : \mathcal{U} \to \mathcal{F}$
- inhomogeneity $f \in \mathcal{F}$, initial value $u_0 \in \mathcal{U}_0$, boundary condition $h \in \mathcal{H}$
- **solution operator:**
  $$G : \mathcal{F} \times \mathcal{H} \times \mathcal{U}_0 \to \mathcal{U}, \quad (f, h, u_0) \mapsto u$$

Parametrization of u_0

$$u_0(\boldsymbol{x}) \approx \sum_{j=1}^{n} a_j A_j(\boldsymbol{x}), \qquad \boldsymbol{x} \in \Omega$$

- fixed functions $A_j, j = 1, \ldots, n$ (B-splines, trig. polynomials, ...)
- parameters $\boldsymbol{a} = (a_1, \ldots, a_n) \in [-1, 1]^n$
- identify u_0 by these parameters $\boldsymbol{a}$

Discretization of the solution operator G

- In practice: Only a subset of the data (f, h, u_0) is varied. $\to$ Today: f and h fixed, $G : \mathcal{U}_0 \to \mathcal{U}$.
- Parametrization yields induced solution operator $G_n : [-1, 1]^n \to \mathcal{U}$ with $\boldsymbol{a} \mapsto u$.

Basis expansion of u in a bounded orthonormal product basis (BOPB)

- discretization G_n can be identified with a function u on $\mathcal{D} = \Omega \times [0,T] \times [-1,1]^n$

$$u(\boldsymbol{x},\tau,\boldsymbol{a}) := \sum_{\boldsymbol{k}\in\mathbb{N}^{d+n}} c_{\boldsymbol{k}} \Phi_{\boldsymbol{k}}(\boldsymbol{x},\tau,\boldsymbol{a}), \qquad (\boldsymbol{x},\tau,\boldsymbol{a}) \in \mathcal{D}$$

- coefficients $c_{\boldsymbol{k}} \in \mathbb{C}$ and functions $\Phi_{\boldsymbol{k}}$ bounded, orthonormal and of tensor-product structure

High-dimensional approximation problem

$$S_I^{\mathcal{A}} u(\boldsymbol{x},\tau,\boldsymbol{a}) := \sum_{\boldsymbol{k}\in I} \hat{u}_{\boldsymbol{k}} \Phi_{\boldsymbol{k}}(\boldsymbol{x},\tau,\boldsymbol{a})$$

- **index set:** $I \subset \mathbb{N}_0^{d+1+n}$ unknown but s-sparse ($|I| = s$)
- **coefficients:** $\hat{u}_{\boldsymbol{k}} \in \mathbb{C}$ approximations of true basis coefficients $c_{\boldsymbol{k}}$

Basis expansion of u in a bounded orthonormal product basis (BOPB)

- discretization G_n can be identified with a function u on $\mathcal{D} = \Omega \times [0,T] \times [-1,1]^n$

$$u(\boldsymbol{x},\tau,\boldsymbol{a}) := \sum_{\boldsymbol{k}\in\mathbb{N}^{d+n}} c_{\boldsymbol{k}} \Phi_{\boldsymbol{k}}(\boldsymbol{x},\tau,\boldsymbol{a}), \qquad (\boldsymbol{x},\tau,\boldsymbol{a}) \in \mathcal{D}$$

- coefficients $c_{\boldsymbol{k}} \in \mathbb{C}$ and functions $\Phi_{\boldsymbol{k}}$ bounded, orthonormal and of tensor-product structure

High-dimensional approximation problem

$$S_I^{\mathcal{A}} u(\boldsymbol{x},\tau,\boldsymbol{a}) := \sum_{\boldsymbol{k}\in I} \hat{u}_{\boldsymbol{k}} \Phi_{\boldsymbol{k}}(\boldsymbol{x},\tau,\boldsymbol{a})$$

- **index set:** $I \subset \mathbb{N}_0^{d+1+n}$ unknown but s-sparse ($|I| = s$)
- **coefficients:** $\hat{u}_{\boldsymbol{k}} \in \mathbb{C}$ approximations of true basis coefficients $c_{\boldsymbol{k}}$

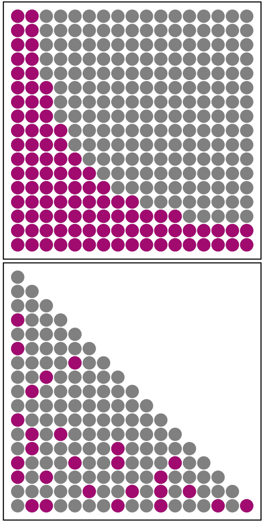
Benefits: The structure of the detected index set I contains various information about the solution and its dependence on $\boldsymbol{x}$, τ and $\boldsymbol{a}$ (which we identified with u_0)!
Problem: How to find a good, s-sparse index set $I \subset \mathbb{N}_0^{d+1+n}$?

- first idea:
  - choose a large search space $\Gamma \supset \mathrm{I}, |\Gamma| \gg |\mathrm{I}|$
  - compute all coefficients $\hat{u}_{\boldsymbol{k}}$
  - choose indices corresponding to the s largest coefficients (absolute value)
- but: unfeasible in practice for large Γ ($\rightarrow$ "curse of dimensionality"), e.g., $\Gamma = [0,64]^{10} \rightarrow |\Gamma| \approx 10^{18}$
- better idea: use a dimension-incremental approach

Benefits: The structure of the detected index set I contains various information about the solution and its dependence on $\boldsymbol{x}$, τ and $\boldsymbol{a}$ (which we identified with u_0)!

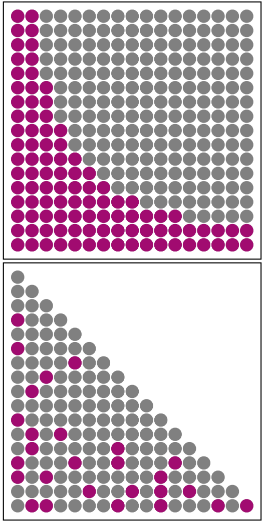
Problem: How to find a good, s-sparse index set $I \subset \mathbb{N}_0^{d+1+n}$?

- first idea:
  - choose a large search space $\Gamma \supset \mathrm{I}, |\Gamma| \gg |\mathrm{I}|$
  - compute all coefficients $\hat{u}_{\boldsymbol{k}}$
  - choose indices corresponding to the s largest coefficients (absolute value)
- but: unfeasible in practice for large Γ ($\to$ "curse of dimensionality"), e.g., $\Gamma = [0, 64]^{10} \to |\Gamma| \approx 10^{18}$
- better idea: use a dimension-incremental approach



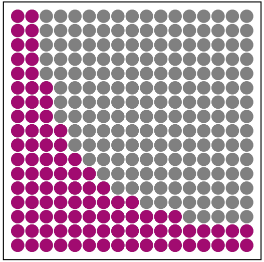
Benefits: The structure of the detected index set I contains various information about the solution and its dependence on $\boldsymbol{x}$, τ and $\boldsymbol{a}$ (which we identified with u_0)!
Problem: How to find a good, s-sparse index set $I \subset \mathbb{N}_0^{d+1+n}$?

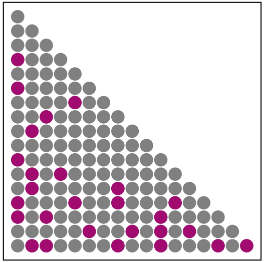
- first idea:
  - choose a large search space $\Gamma \supset \mathrm{I}, |\Gamma| \gg |\mathrm{I}|$
  - compute all coefficients $\hat{u}_{\boldsymbol{k}}$
  - choose indices corresponding to the s largest coefficients (absolute value)
- but: unfeasible in practice for large Γ ($\rightarrow$ "curse of dimensionality"), e.g., $\Gamma = [0, 64]^{10} \rightarrow |\Gamma| \approx 10^{18}$
- better idea: use a dimension-incremental approach



Benefits: The structure of the detected index set I contains various information about the solution and its dependence on $\boldsymbol{x}$, τ and $\boldsymbol{a}$ (which we identified with u_0)!
Problem: How to find a good, s-sparse index set $I \subset \mathbb{N}_0^{d+1+n}$?

- first idea:
  - choose a large search space $\Gamma \supset \mathrm{I}, |\Gamma| \gg |\mathrm{I}|$
  - compute all coefficients $\hat{u}_{\boldsymbol{k}}$
  - choose indices corresponding to the s largest coefficients (absolute value)
- but: unfeasible in practice for large Γ ($\rightarrow$ "curse of dimensionality"), e.g., $\Gamma = [0, 64]^{10} \rightarrow |\Gamma| \approx 10^{18}$
- better idea: use a dimension-incremental approach





Differential equation setting

- approximate $u(\boldsymbol{x}, \tau, \boldsymbol{a})$ with $(\boldsymbol{x}, \tau, \boldsymbol{a}) \in \mathcal{D}$
- compute $I \subset \mathbb{N}_0^{d+1+n}$ and $\hat{u}_{\boldsymbol{k}}$ for $\boldsymbol{k} \in I$

$\leftrightarrow$

Function approximation setting

- approximate $g(\boldsymbol{y})$ with $\boldsymbol{y} \in \mathcal{D} \subset \mathbb{R}^d$
- compute $I \subset \mathbb{N}_0^d$ and $\hat{g}_{\boldsymbol{k}}$ for $\boldsymbol{k} \in I$

Input

- target function g (black box)
- search space $\Gamma \subset \mathbb{N}_0^d$
- sparsity $s \in \mathbb{N}$
- detection threshold $\delta_+ > 0$
- number of detection iterations $r \in \mathbb{N}$

$\longrightarrow$

Output

- detected index set $I \subset \mathbb{N}_0^d$ with $|I| = s$
- approximated coefficients $\hat{g}_{\boldsymbol{k}}$ for all $\boldsymbol{k} \in I$

Differential equation setting

- approximate $u(\boldsymbol{x}, \tau, \boldsymbol{a})$ with $(\boldsymbol{x}, \tau, \boldsymbol{a}) \in \mathcal{D}$
- compute $I \subset \mathbb{N}_0^{d+1+n}$ and $\hat{u}_{\boldsymbol{k}}$ for $\boldsymbol{k} \in I$

$\leftrightarrow$

Function approximation setting

- approximate $g(\boldsymbol{y})$ with $\boldsymbol{y} \in \mathcal{D} \subset \mathbb{R}^d$
- compute $I \subset \mathbb{N}_0^d$ and $\hat{g}_{\boldsymbol{k}}$ for $\boldsymbol{k} \in I$

Input

- target function g (black box)
- search space $\Gamma \subset \mathbb{N}_0^d$
- sparsity $s \in \mathbb{N}$
- detection threshold $\delta_+ > 0$
- number of detection iterations $r \in \mathbb{N}$

$\longrightarrow$

Output

- detected index set $I \subset \mathbb{N}_0^d$ with $|I| = s$
- approximated coefficients $\hat{g}_{\boldsymbol{k}}$ for all $\boldsymbol{k} \in I$

Differential equation setting

- approximate $u(\boldsymbol{x}, \tau, \boldsymbol{a})$ with $(\boldsymbol{x}, \tau, \boldsymbol{a}) \in \mathcal{D}$
- compute $I \subset \mathbb{N}_0^{d+1+n}$ and $\hat{u}_{\boldsymbol{k}}$ for $\boldsymbol{k} \in I$

$\leftrightarrow$

Function approximation setting

- approximate $g(\boldsymbol{y})$ with $\boldsymbol{y} \in \mathcal{D} \subset \mathbb{R}^d$
- compute $I \subset \mathbb{N}_0^d$ and $\hat{g}_{\boldsymbol{k}}$ for $\boldsymbol{k} \in I$

Input

- target function g (black box)
- search space $\Gamma \subset \mathbb{N}_0^d$
- sparsity $s \in \mathbb{N}$
- detection threshold $\delta_+ > 0$
- number of detection iterations $r \in \mathbb{N}$

$\longrightarrow$

Output

- detected index set $I \subset \mathbb{N}_0^d$ with $|I| = s$
- approximated coefficients $\hat{g}_{\boldsymbol{k}}$ for all $\boldsymbol{k} \in I$

Projected coefficients

$$c_{\{1,\dots,t\},\boldsymbol{k}}(\tilde{\boldsymbol{y}}) := \int_{\mathcal{D}_{\{1,\dots,t\}}} g(\boldsymbol{\xi},\tilde{\boldsymbol{y}})\,\overline{\Phi_{\{1,\dots,t\},\boldsymbol{k}}(\boldsymbol{\xi})}\,\mathrm{d}\mu_{\{1,\dots,t\}}(\boldsymbol{\xi}) \qquad \forall \boldsymbol{k}\in\mathbb{N}_0^t,\ \forall\tilde{\boldsymbol{y}}\in\mathcal{D}_{\{t+1,\dots,d\}}$$

Projected coefficients are similar to basis coefficients $c_{\boldsymbol{k}}$, but:

- fix some dimensions via a random anchor $\tilde{\boldsymbol{y}}$
- integrate only over the remaining dimensions

Projected coefficients indicate the "importance" of the indices $(\boldsymbol{k},*,*,*,\dots)$, since

$$c_{\{1,\dots,t\},\boldsymbol{k}}(\tilde{\boldsymbol{y}}) = \dots = \sum_{\boldsymbol{h}\in\mathbb{N}_0^{d-t}} c_{(\boldsymbol{k},\boldsymbol{h})}\,\Phi_{\{t+1,\dots,d\},(\boldsymbol{k},\boldsymbol{h})}(\tilde{\boldsymbol{y}}).$$

Example: Fourier setting with $d=3$ and $t=2$

$$c_{\{1,2\},\boldsymbol{k}}(\tilde{y}) := \int_{\mathbb{T}^2} g(\boldsymbol{\xi},\tilde{y})\,\mathrm{e}^{-2\pi\mathrm{i}\boldsymbol{k}\cdot\boldsymbol{\xi}}\,\mathrm{d}\boldsymbol{\xi} = \sum_{h\in\mathbb{Z}} c_{(\boldsymbol{k},h)}\mathrm{e}^{2\pi\mathrm{i}h\tilde{y}} \qquad \forall\boldsymbol{k}\in\mathbb{Z}^2,\ \forall\tilde{y}\in\mathbb{T}$$

Projected coefficients

$$c_{\{1,\dots,t\},\boldsymbol{k}}(\tilde{\boldsymbol{y}}) := \int_{\mathcal{D}_{\{1,\dots,t\}}} g(\boldsymbol{\xi},\tilde{\boldsymbol{y}})\,\overline{\Phi_{\{1,\dots,t\},\boldsymbol{k}}(\boldsymbol{\xi})}\,\mathrm{d}\mu_{\{1,\dots,t\}}(\boldsymbol{\xi}) \qquad \forall \boldsymbol{k}\in\mathbb{N}_0^t,\ \forall\tilde{\boldsymbol{y}}\in\mathcal{D}_{\{t+1,\dots,d\}}$$

Projected coefficients are similar to basis coefficients $c_{\boldsymbol{k}}$, but:

- fix some dimensions via a random anchor $\tilde{\boldsymbol{y}}$
- integrate only over the remaining dimensions

Projected coefficients indicate the “importance” of the indices $(\boldsymbol{k},*,*,*,\dots)$, since

$$c_{\{1,\dots,t\},\boldsymbol{k}}(\tilde{\boldsymbol{y}}) = \dots = \sum_{\boldsymbol{h}\in\mathbb{N}_0^{d-t}} c_{(\boldsymbol{k},\boldsymbol{h})}\,\Phi_{\{t+1,\dots,d\},(\boldsymbol{k},\boldsymbol{h})}(\tilde{\boldsymbol{y}}).$$

Example: Fourier setting with $d=3$ and $t=2$

$$c_{\{1,2\},\boldsymbol{k}}(\tilde{y}) := \int_{\mathbb{T}^2} g(\boldsymbol{\xi},\tilde{y})\,\mathrm{e}^{-2\pi\mathrm{i}\boldsymbol{k}\cdot\boldsymbol{\xi}}\,\mathrm{d}\boldsymbol{\xi} = \sum_{h\in\mathbb{Z}} c_{(\boldsymbol{k},h)}\mathrm{e}^{2\pi\mathrm{i}h\tilde{y}} \qquad \forall\boldsymbol{k}\in\mathbb{Z}^2,\ \forall\tilde{y}\in\mathbb{T}$$

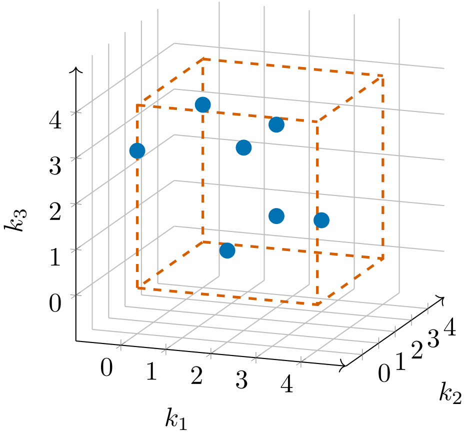


Figure: The desired but unknown index set I and the search space $\Gamma = \{0, 1, 2, 3, 4\}^3$.

The algorithm

The dimension-incremental method

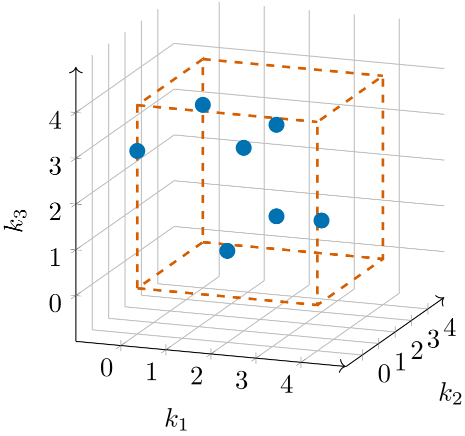


Figure: The desired but unknown index set I and the search space $\Gamma = \{0, 1, 2, 3, 4\}^3$.

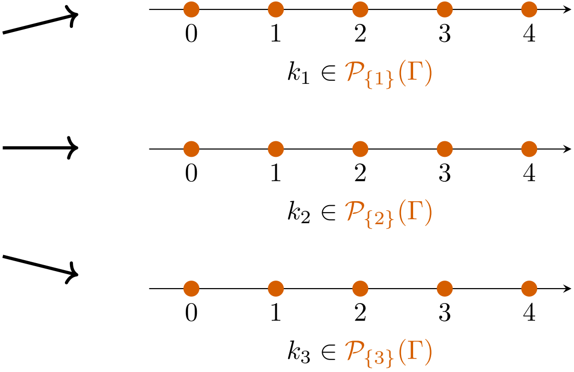


Figure: The one-dimensional candidate sets $\mathcal{P}_{\{j\}}(\Gamma)$.

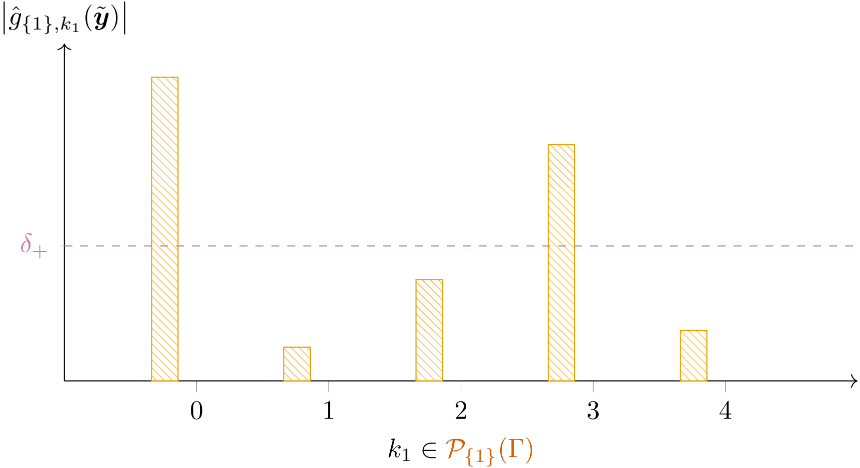


Figure: The one-dimensional detection step in the first dimension with $r = 3$ detection iterations.

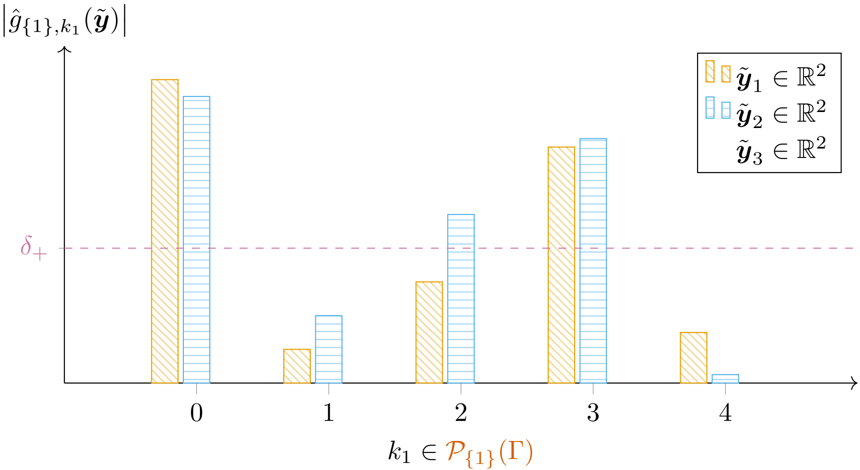


Figure: The one-dimensional detection step in the first dimension with $r = 3$ detection iterations.

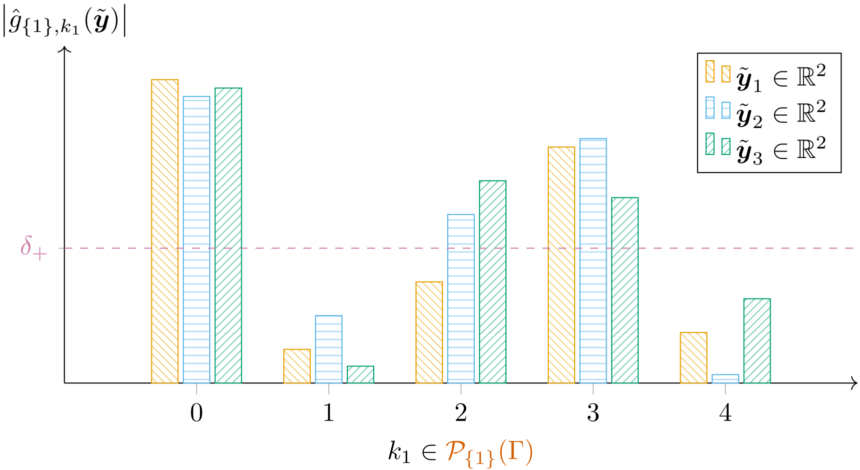


Figure: The one-dimensional detection step in the first dimension with $r = 3$ detection iterations.

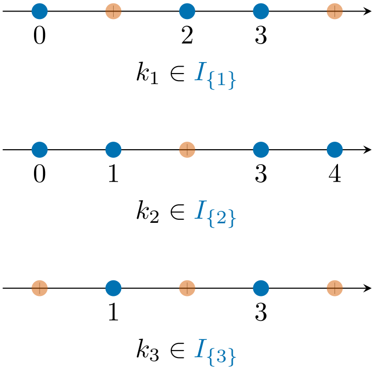


Figure: The one-dimensional index sets $I_{\{j\}}$.

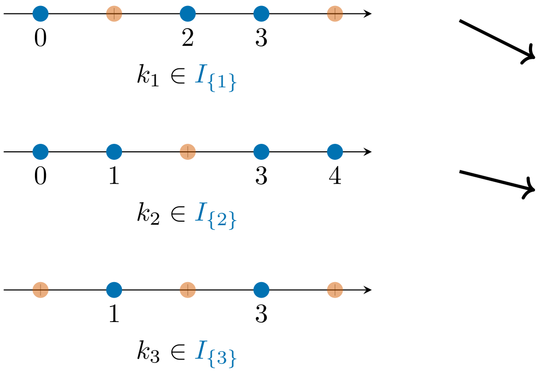


Figure: The one-dimensional index sets $I_{\{j\}}$.

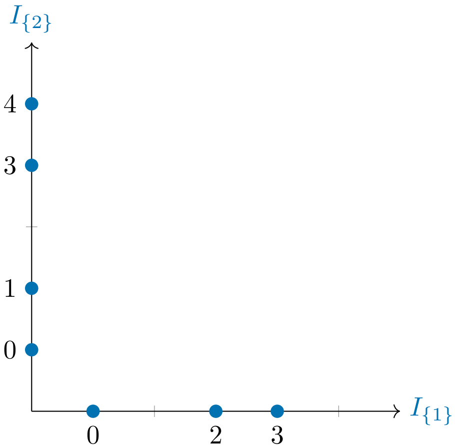


Figure: The two-dimensional candidate set $I_{\{1\}} \times I_{\{2\}}$.

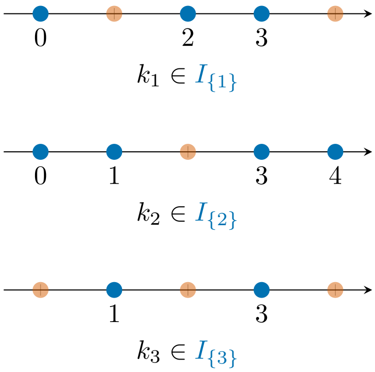


Figure: The one-dimensional index sets $I_{\{j\}}$.

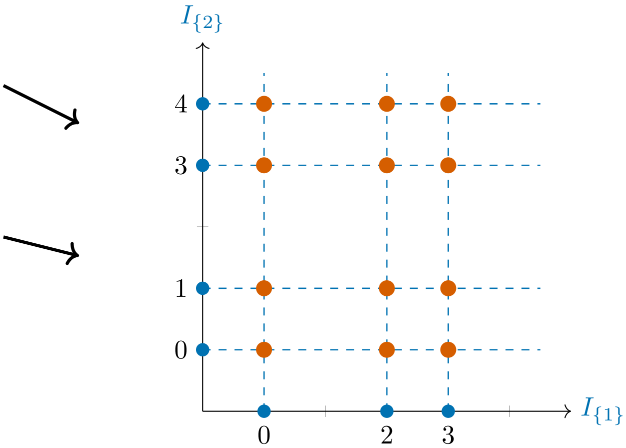


Figure: The two-dimensional candidate set $I_{\{1\}} \times I_{\{2\}}$.

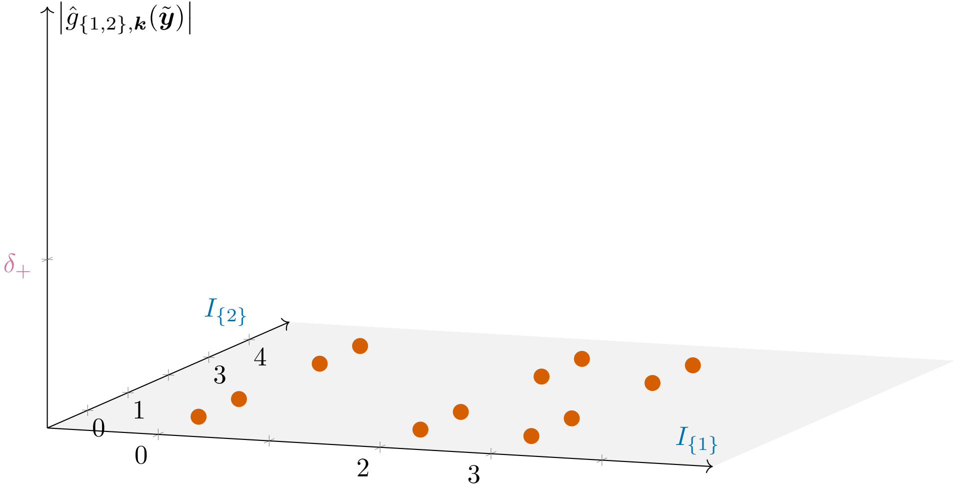


Figure: The two-dimensional detection step in the dimensions $\{1,2\}$. (Only $r=1$ detection iteration shown.)

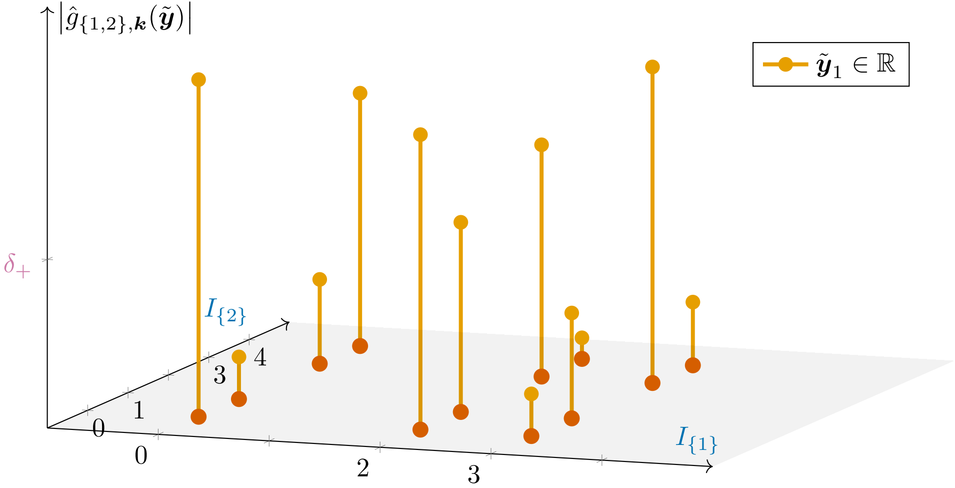


Figure: The two-dimensional detection step in the dimensions $\{1,2\}$. (Only $r=1$ detection iteration shown.)

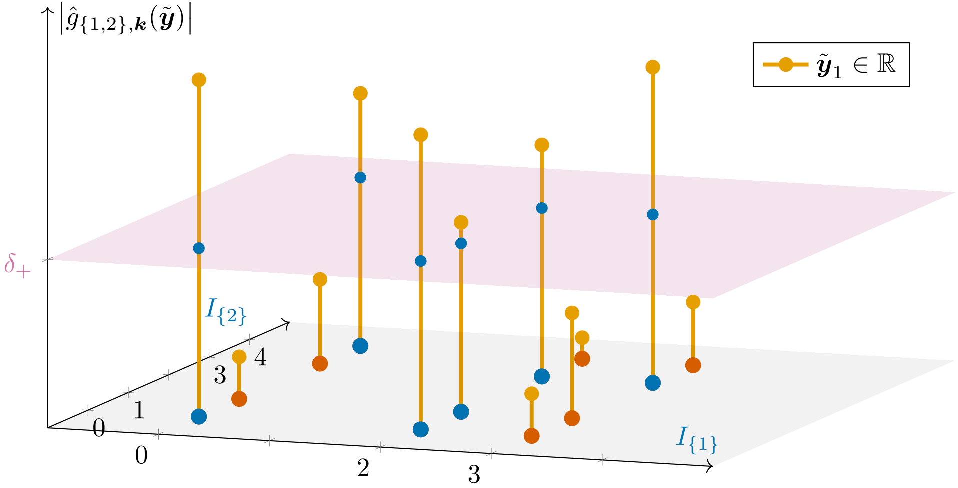


Figure: The two-dimensional detection step in the dimensions $\{1, 2\}$. (Only $r = 1$ detection iteration shown.)

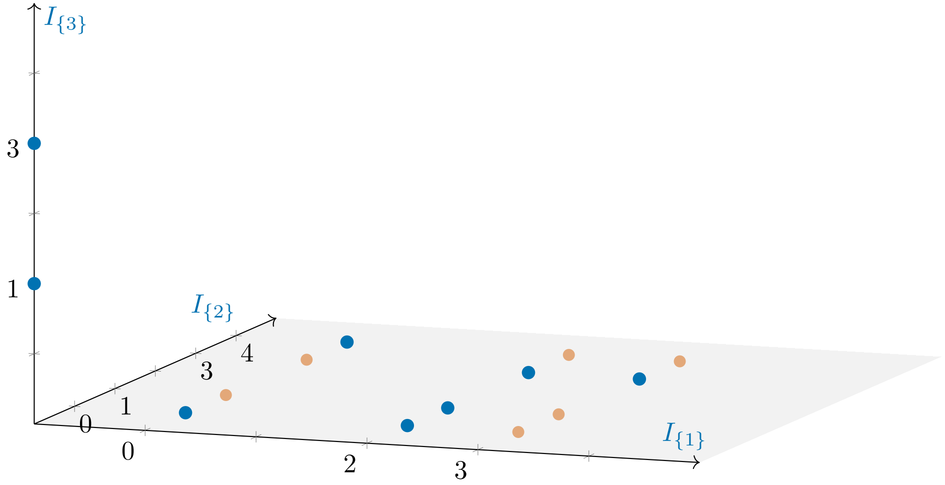


Figure: The two-dimensional index set $I_{\{1,2\}}$ and the three-dimensional candidate set $I_{\{1,2\}} \times I_{\{3\}}$.

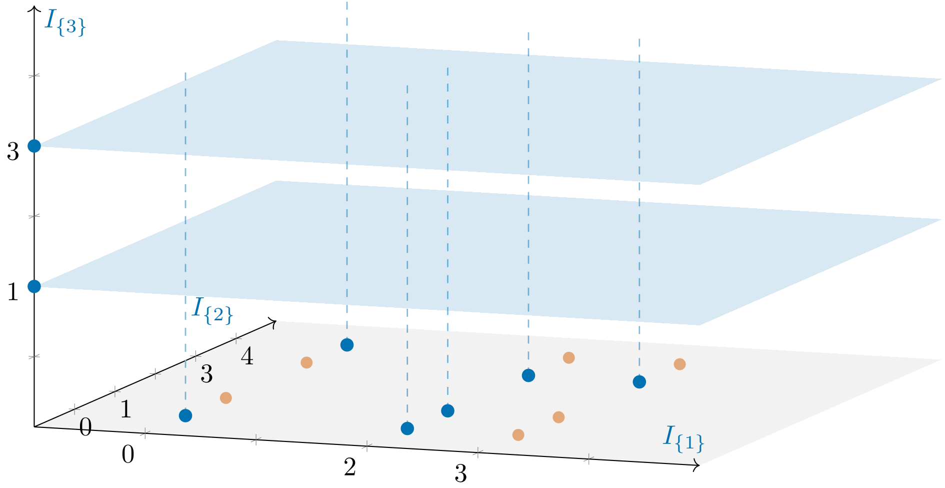


Figure: The two-dimensional index set $I_{\{1,2\}}$ and the three-dimensional candidate set $I_{\{1,2\}} \times I_{\{3\}}$.

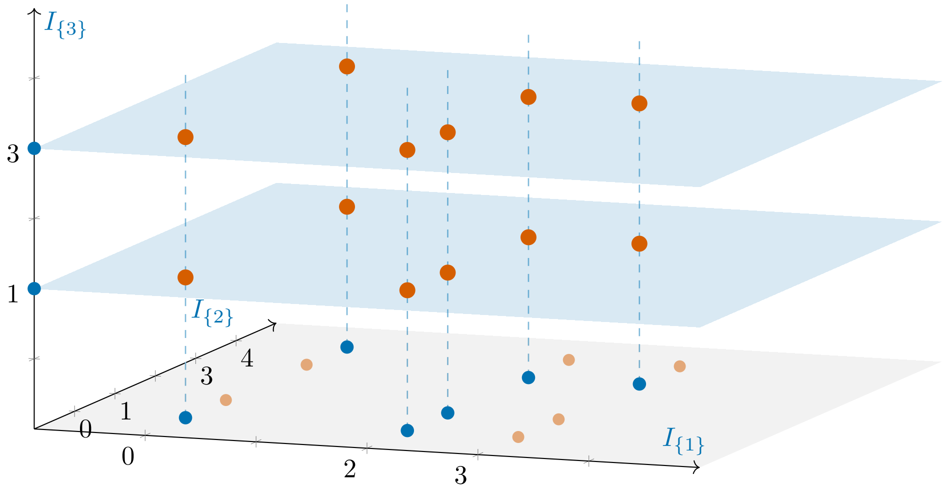


Figure: The two-dimensional index set $I_{\{1,2\}}$ and the three-dimensional candidate set $I_{\{1,2\}} \times I_{\{3\}}$.

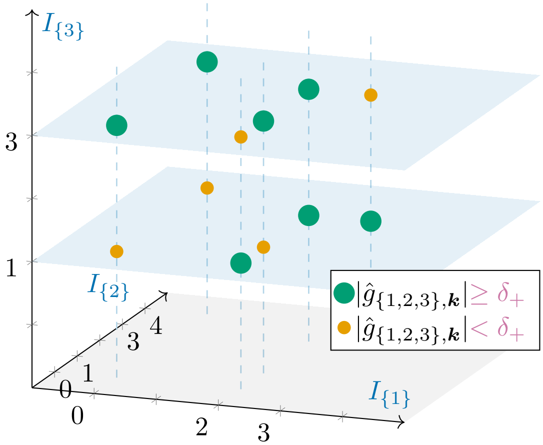


Figure: The three-dimensional detection step with the detected index set $I_{\{1,2,3\}}$.

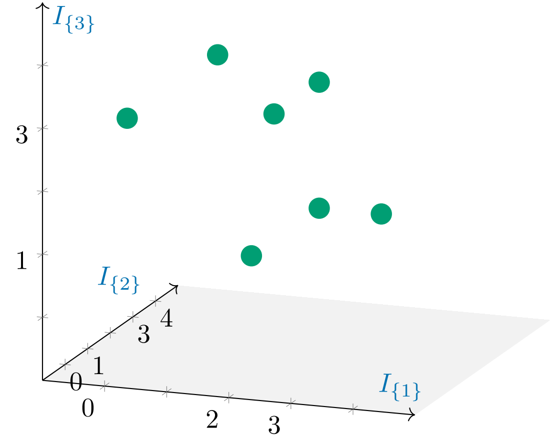


Figure: The three-dimensional detection step with the detected index set $I_{\{1,2,3\}}$.

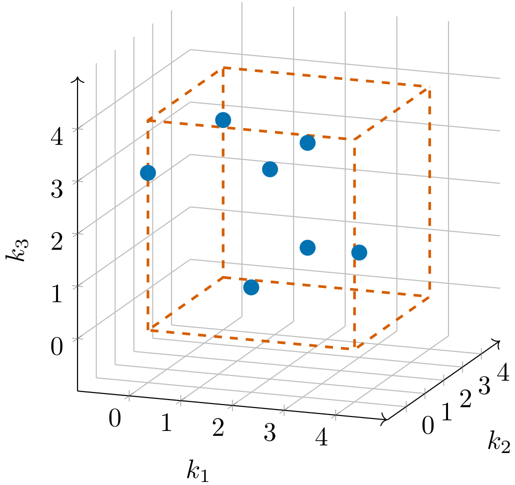


Figure: The correctly detected index set $I = I_{\{1,2,3\}}$.

- Projected coefficients: $\boldsymbol{c}(\tilde{\boldsymbol{y}}) =: c_{\{1,\ldots,t\},\boldsymbol{k}}(\tilde{\boldsymbol{y}}) = \int_{\ldots} g(\boldsymbol{\xi}, \tilde{\boldsymbol{y}}) \overline{\Phi_{\{1,\ldots,t\},\boldsymbol{k}}(\boldsymbol{\xi})} \,\mathrm{d}\boldsymbol{\xi} := \boldsymbol{\Phi}^* \boldsymbol{g}(\tilde{\boldsymbol{y}})$

Numerical Integration

- consider the integral formula
- Approximation via cubature rules, MC methods, QMC methods (e.g. rank-1 lattices), ...

Pros & Cons

- typically stable, reliable and efficient
- accurate approximations for all $\boldsymbol{k} \in K = I_{\{1,\ldots,t-1\}} \times I_{\{t\}}$
- large amount of samples required, e.g., $M \approx |K|^2$ or $\approx |K| \log |K|$
- does not benefit from additional samples

Solving a linear system

- consider the linear system $\boldsymbol{\Phi c} = \boldsymbol{g}$
- Assumption: $\boldsymbol{c}$ is approximately s-sparse.
- Approximation via sparse recovery methods (e.g. OMP, CoSaMP, ...)

Pros & Cons

- small amount of samples required, e.g., $M \approx 5s$
- benefits from additional samples
- naturally incorporates the sparsity s
- expensive matrix-vector multiplication

- Projected coefficients: $\boldsymbol{c}(\tilde{\boldsymbol{y}}) =: c_{\{1,\ldots,t\},\boldsymbol{k}}(\tilde{\boldsymbol{y}}) = \int_{\ldots} g(\boldsymbol{\xi},\tilde{\boldsymbol{y}})\,\overline{\Phi_{\{1,\ldots,t\},\boldsymbol{k}}(\boldsymbol{\xi})}\,\mathrm{d}\boldsymbol{\xi} := \boldsymbol{\Phi}^*\boldsymbol{g}(\tilde{\boldsymbol{y}})$

Numerical Integration

- consider the integral formula
- Approximation via cubature rules, MC methods, QMC methods (e.g. rank-1 lattices), ...

Pros & Cons

- typically stable, reliable and efficient
- accurate approximations for all $\boldsymbol{k} \in K = I_{\{1,\ldots,t-1\}} \times I_{\{t\}}$
- large amount of samples required, e.g., $M \approx |K|^2$ or $\approx |K| \log |K|$
- does not benefit from additional samples

Solving a linear system

- consider the linear system $\boldsymbol{\Phi}\boldsymbol{c} = \boldsymbol{g}$
- Assumption: $\boldsymbol{c}$ is approximately s-sparse.
- Approximation via sparse recovery methods (e.g. OMP, CoSaMP, ...)

Pros & Cons

- small amount of samples required, e.g., $M \approx 5s$
- benefits from additional samples
- naturally incorporates the sparsity s
- expensive matrix-vector multiplication

- Projected coefficients: $\boldsymbol{c}(\tilde{\boldsymbol{y}}) =: c_{\{1,\dots,t\},\boldsymbol{k}}(\tilde{\boldsymbol{y}}) = \int_{\dots} g(\boldsymbol{\xi},\tilde{\boldsymbol{y}})\,\overline{\Phi_{\{1,\dots,t\},\boldsymbol{k}}(\boldsymbol{\xi})}\,\mathrm{d}\boldsymbol{\xi} := \boldsymbol{\Phi}^*\boldsymbol{g}(\tilde{\boldsymbol{y}})$

Numerical Integration

- consider the integral formula
- Approximation via cubature rules, MC methods, QMC methods (e.g. rank-1 lattices), ...

Pros & Cons

- typically stable, reliable and efficient
- accurate approximations for all $\boldsymbol{k} \in K = I_{\{1,\dots,t-1\}} \times I_{\{t\}}$
- large amount of samples required, e.g., $M \approx |K|^2$ or $\approx |K| \log |K|$
- does not benefit from additional samples

Solving a linear system

- consider the linear system $\boldsymbol{\Phi c} = \boldsymbol{g}$
- Assumption: $\boldsymbol{c}$ is approximately s-sparse.
- Approximation via sparse recovery methods (e.g. OMP, CoSaMP, ...)

Pros & Cons

- small amount of samples required, e.g., $M \approx 5s$
- benefits from additional samples
- naturally incorporates the sparsity s
- expensive matrix-vector multiplication

- Projected coefficients: $\boldsymbol{c}(\tilde{\boldsymbol{y}}) =: c_{\{1,\dots,t\},\boldsymbol{k}}(\tilde{\boldsymbol{y}}) = \int_{\dots} g(\boldsymbol{\xi},\tilde{\boldsymbol{y}})\,\overline{\Phi_{\{1,\dots,t\},\boldsymbol{k}}(\boldsymbol{\xi})}\,\mathrm{d}\boldsymbol{\xi} := \boldsymbol{\Phi}^{*}\boldsymbol{g}(\tilde{\boldsymbol{y}})$

Numerical Integration

- consider the integral formula
- Approximation via cubature rules, MC methods, QMC methods (e.g. rank-1 lattices), ...

Pros & Cons

- typically stable, reliable and efficient
- accurate approximations for all $\boldsymbol{k} \in K = I_{\{1,\dots,t-1\}} \times I_{\{t\}}$
- large amount of samples required, e.g., $M \approx |K|^2$ or $\approx |K| \log |K|$
- does not benefit from additional samples

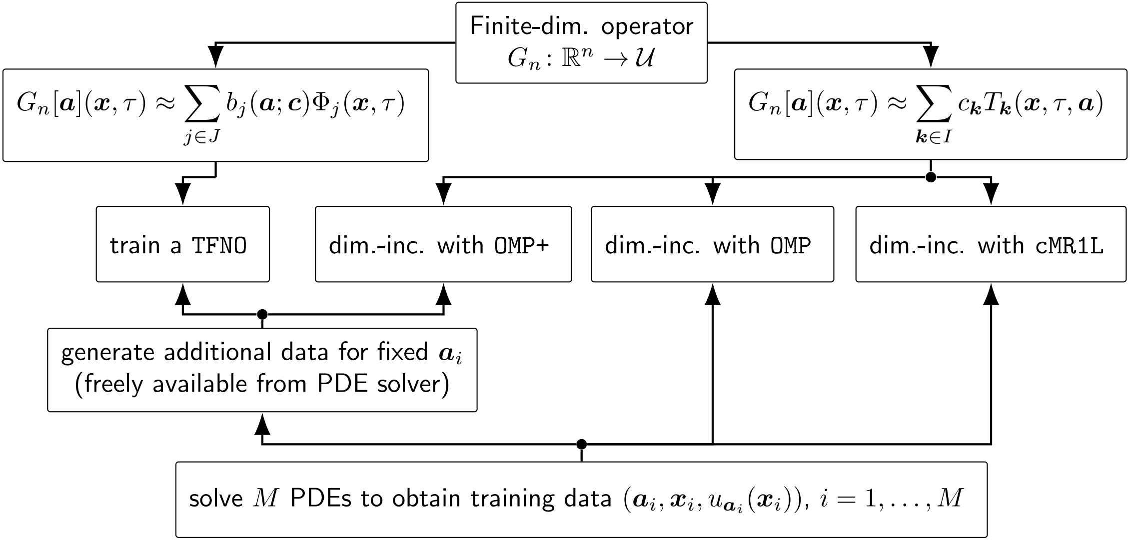
Solving a linear system

- consider the linear system $\boldsymbol{\Phi}\boldsymbol{c} = \boldsymbol{g}$
- Assumption: $\boldsymbol{c}$ is approximately s-sparse.
- Approximation via sparse recovery methods (e.g. OMP, CoSaMP, ...)

Pros & Cons

- small amount of samples required, e.g., $M \approx 5s$
- benefits from additional samples
- naturally incorporates the sparsity s
- expensive matrix-vector multiplication

Comparison of considered strategies



Heat equation (one-dimensional)

$$
\begin{aligned}
\partial_\tau u = \frac{1}{16}\partial_{xx} u, &\quad x,\tau \in (0,1)\\
u(x,0) = u_0(x), &\quad x \in (0,1)\\
u(0,\tau) = u(1,\tau), = 0 &\quad \tau \in (0,1)
\end{aligned}
$$

- sampling: solve for given u_0 with method of lines based solver

Parametrization of u_0

- sine series:

$$u_0(x) \approx \sum_{\ell=1}^{9} a_\ell \sin(\ell\pi x)$$

Exact solution

$$u(x,\tau,\boldsymbol{a}) = \sum_{\ell=1}^{9} a_\ell \sin(\ell\pi x) \exp\left(\frac{1}{16}\ell^2\pi^2\tau\right)$$

Heat equation (one-dimensional)

$$\partial_\tau u = \frac{1}{16}\partial_{xx} u, \quad x, \tau \in (0,1)$$
$$u(x,0) = u_0(x), \quad x \in (0,1)$$
$$u(0,\tau) = u(1,\tau), = 0 \quad \tau \in (0,1)$$

- sampling: solve for given u_0 with method of lines based solver

Parametrization of u_0

- sine series:

$$u_0(x) \approx \sum_{\ell=1}^{9} a_\ell \sin(\ell \pi x)$$

Exact solution

$$u(x,\tau,\boldsymbol{a}) = \sum_{\ell=1}^{9} a_\ell \sin(\ell\pi x) \exp\left(\frac{1}{16}\ell^2\pi^2\tau\right)$$

Heat equation (one-dimensional)

$$\begin{aligned} \partial_\tau u &= \frac{1}{16}\partial_{xx}u, & x,\tau &\in (0,1) \\ u(x,0) &= u_0(x), & x &\in (0,1) \\ u(0,\tau) = u(1,\tau), &= 0 & \tau &\in (0,1) \end{aligned}$$

- sampling: solve for given u_0 with method of lines based solver

Parametrization of u_0

- sine series:

$$u_0(x) \approx \sum_{\ell=1}^{9} a_\ell \sin(\ell\pi x)$$

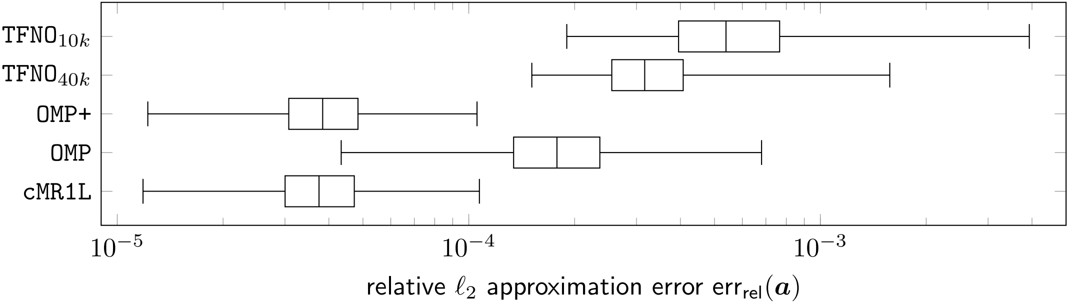
Exact solution

$$u(x,\tau,\boldsymbol{a}) = \sum_{\ell=1}^{9} a_\ell \sin(\ell\pi x) \exp\left(\frac{1}{16}\ell^2\pi^2\tau\right)$$

- sparsity $s = 1000$
- search space $\Gamma = [0, 64]^{11}$
- BOPB: multivariate Chebyshev polynomials $T_{\boldsymbol{k}}$

Transformed solution

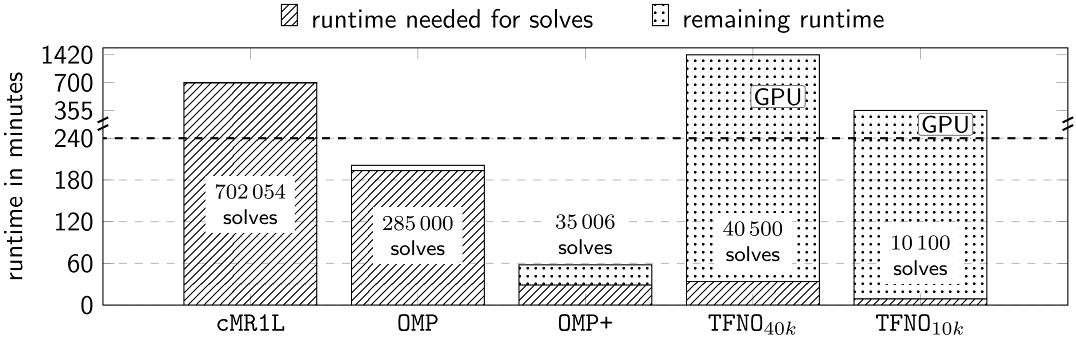
$$\tilde{u}(\tilde{x}, \tilde{\tau}, \boldsymbol{a}) = \sum_{\ell=1}^{9} a_\ell \sin\left(\frac{\ell\pi}{2}(\tilde{x}+1)\right) \exp\left(\frac{\ell^2\pi^2}{32}(\tilde{\tau}+1)\right)$$



- sparsity $s = 1000$
- search space $\Gamma = [0, 64]^{11}$
- BOPB: multivariate Chebyshev polynomials $T_{\boldsymbol{k}}$

Transformed solution

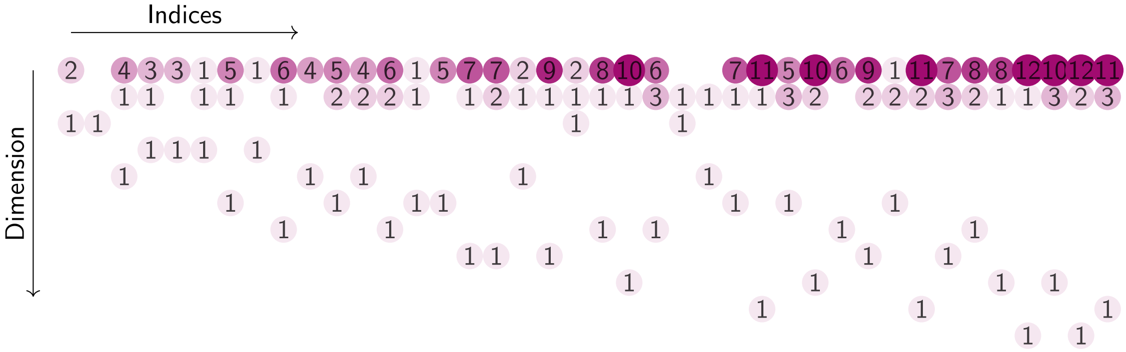
$$\tilde{u}(\tilde{x}, \tilde{\tau}, \boldsymbol{a}) = \sum_{\ell=1}^{9} a_\ell \sin\left(\frac{\ell\pi}{2}(\tilde{x}+1)\right) \exp\left(\frac{\ell^2\pi^2}{32}(\tilde{\tau}+1)\right)$$



- sparsity $s = 1000$
- search space $\Gamma = [0, 64]^{11}$
- BOPB: multivariate Chebyshev polynomials $T_{\boldsymbol{k}}$

Transformed solution

$$\tilde{u}(\tilde{x}, \tilde{\tau}, \boldsymbol{a}) = \sum_{\ell=1}^{9} a_\ell \sin\left(\frac{\ell\pi}{2}(\tilde{x}+1)\right) \exp\left(\frac{\ell^2\pi^2}{32}(\tilde{\tau}+1)\right)$$



Parametric diffusion equation

$$
\begin{aligned}
-\nabla \cdot (a(\boldsymbol{x}, \boldsymbol{y}) \nabla u(\boldsymbol{x}, \boldsymbol{y})) &= 1, \\
u(\boldsymbol{x}, \boldsymbol{y}) &= 0.
\end{aligned}
$$

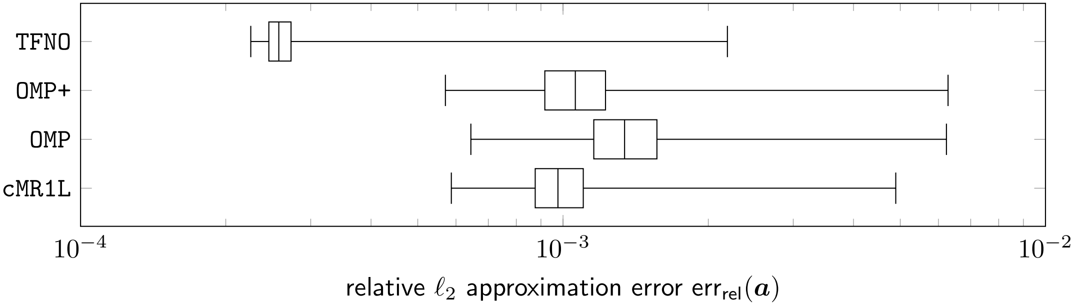
- domains: $\boldsymbol{x} \in [0,1]^2$ and $\boldsymbol{y} \in [-1,1]^{20}$
- sampling: solve for given $\boldsymbol{y}$ using FEM (FEniCS)

Affine coefficient a

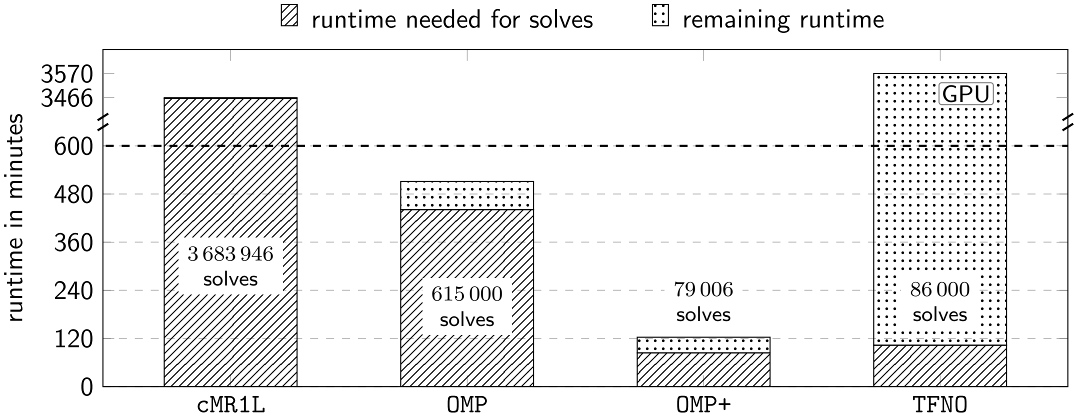
$$
a(\boldsymbol{x}, \boldsymbol{y}) = 1 + \sum_{j=1}^{20} y_j \psi_j(\boldsymbol{x}),
$$

- $\psi_j(\boldsymbol{x}) = c j^{-\mu} \cos(2\pi m_1(j) x_1) \cos(2\pi m_2(j) x_2)$,
- $m_1(j) := j - \frac{k(j)(k(j)+1)}{2}$
- $m_2(j) := k(j) - m_1(j)$
- $k(j) := \lfloor -1/2 + \sqrt{1/4 + 2j} \rfloor$
- $c = 0.9/\zeta(2)$ and $\mu = 2$

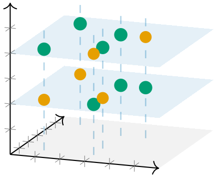
- sparsity $s = 1000$
- search space $\Gamma = [0, 64]^{22}$
- BOPB: multivariate Chebyshev polynomials $T_{\boldsymbol{k}}$

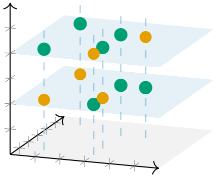


- sparsity $s = 1000$
- search space $\Gamma = [0, 64]^{22}$
- BOPB: multivariate Chebyshev polynomials $T_{\boldsymbol{k}}$



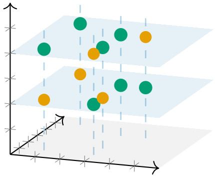
- What did we do?
  - approached the solution operator of differential equations
  - applied high-dimensional approximation methods
  - identified structural information about the solution
  - reduced the computational effort tremendously using sparse recovery



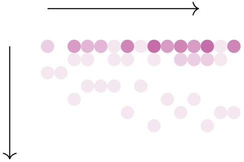


- **What did we do?**
  - approached the solution operator of differential equations
  - applied high-dimensional approximation methods
  - identified structural information about the solution
  - reduced the computational effort tremendously using sparse recovery
- **What did we see?**
  - reasonable numerical results and approximation errors
  - accessible information about the approximation by its index set I
  - results being consistent with analytical solutions for toy examples

- What did we do?
  - approached the solution operator of differential equations
  - applied high-dimensional approximation methods
  - identified structural information about the solution
  - reduced the computational effort tremendously using sparse recovery



- What did we see?
  - reasonable numerical results and approximation errors
  - accessible information about the approximation by its index set I
  - results being consistent with analytical solutions for toy examples
- Open problems?
  - Application of our method to more difficult differential problems?
  - Exploration of alternative basis functions with improved sparsity properties?



- *NABOPB: Nonlinear approximation in bounded orthonormal product bases*
  GitHub Repository: https://github.com/fabiantaubert/nabopb

- *NABOPB: Nonlinear approximation in bounded orthonormal product bases*
  GitHub Repository: https://github.com/fabiantaubert/nabopb

- Sebastian Neumayer, Daniel Potts, Fabian Taubert
  Learning solution operators of PDEs with sparse approximation methods
  ArXiv e-prints, 2026. arXiv:2606.06046 [math.NA]
- Daniel Potts, Fabian Taubert
  An approach to discrete operator learning based on sparse high-dimensional approximation
  Electron. Trans. Numer. Anal., 63, 468-495, 2025.
- Lutz Kämmerer, Daniel Potts, Fabian Taubert
  Nonlinear approximation in bounded orthonormal product bases
  Sampl. Theory Signal Process. Data Anal., 21, Article number: 19, 2023.

- *NABOPB: Nonlinear approximation in bounded orthonormal product bases*
  GitHub Repository: https://github.com/fabiantaubert/nabopb

- Sebastian Neumayer, Daniel Potts, Fabian Taubert
  Learning solution operators of PDEs with sparse approximation methods
  ArXiv e-prints, 2026. arXiv:2606.06046 [math.NA]
- Daniel Potts, Fabian Taubert
  An approach to discrete operator learning based on sparse high-dimensional approximation
  Electron. Trans. Numer. Anal., 63, 468-495, 2025.
- Lutz Kämmerer, Daniel Potts, Fabian Taubert
  Nonlinear approximation in bounded orthonormal product bases
  Sampl. Theory Signal Process. Data Anal., 21, Article number: 19, 2023.

Thank you for your attention!

Questions? Ideas? Suggestions?