Elliptic PDE learning is provably data-efficient

Alex Townsend
Cornell University
townsend@cornell.edu

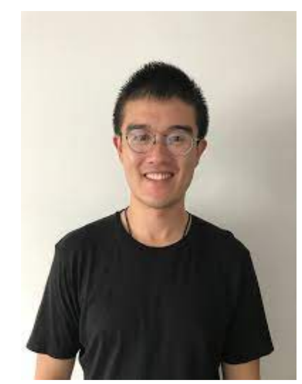
Joint work with

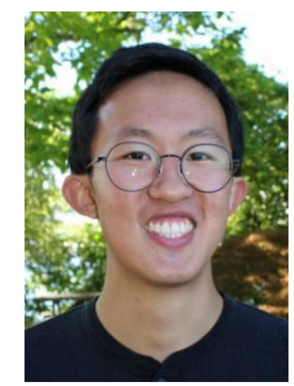

Nicolas Boullé


Diana Halikias


Seick Kim


Sam Otto


Tianyi Shi


Chris Wang

Related papers:

"Learning elliptic partial differential equations with randomized linear algebra" by Boullé and T. in FoCM, 2022

"Learning Green's functions associated with time-dependent partial differential equations" by Boullé, Kim, Shi, and T. in JMLR, 2022

"Elliptic PDE learning is provably data-efficient" by Boullé, Halikias and T. in PNAS, 2023

"Operator learning for hyperbolic partial differential equations" by Wang and T., on ArXiv, 2024

Operator learning in a nutshell

Operator between function spaces: $\mathscr{G} : \mathscr{X} \to \mathscr{Y}$

Operator learning in a nutshell

Operator between function spaces: $\mathscr{G} : \mathscr{X} \to \mathscr{Y}$

Approx. $\mathscr{G}$ by building a parametric map $\hat{\mathscr{G}}_\theta$

Operator learning in a nutshell

Operator between function spaces: $\mathscr{G} : \mathscr{X} \to \mathscr{Y}$

Approx. $\mathscr{G}$ by building a parametric map $\hat{\mathscr{G}}_\theta$

E.g., $\hat{\mathscr{G}}_\theta = \mathscr{Q} \circ \sigma(\mathscr{K}_L) \circ \cdots \circ \sigma(\mathscr{K}_1) \circ \mathscr{P}$

Operator learning in a nutshell

Operator between function spaces: $\mathscr{G} : \mathscr{X} \to \mathscr{Y}$

Approx. $\mathscr{G}$ by building a parametric map $\hat{\mathscr{G}}_\theta$

E.g., $\hat{\mathscr{G}}_\theta = \mathscr{Q} \circ \sigma(\mathscr{K}_L) \circ \cdots \circ \sigma(\mathscr{K}_1) \circ \mathscr{P}$

Decoder

Operator learning in a nutshell

Operator between function spaces: $\mathscr{G} : \mathscr{X} \to \mathscr{Y}$

Approx. $\mathscr{G}$ by building a parametric map $\hat{\mathscr{G}}_\theta$

E.g., $\hat{\mathscr{G}}_\theta = \mathscr{Q} \circ \sigma(\mathscr{K}_L) \circ \cdots \circ \sigma(\mathscr{K}_1) \circ \mathscr{P}$

Decoder

Encoder

Operator learning in a nutshell

Operator between function spaces: $\mathscr{G} : \mathscr{X} \to \mathscr{Y}$

Approx. $\mathscr{G}$ by building a parametric map $\hat{\mathscr{G}}_\theta$

$$\text{E.g., } \hat{\mathscr{G}}_\theta = \underbrace{\mathscr{Q}}_{\text{Decoder}} \circ \underbrace{\sigma}_{\text{Nonlinear activation function}}(\mathscr{K}_L) \circ \cdots \circ \sigma(\mathscr{K}_1) \circ \underbrace{\mathscr{P}}_{\text{Encoder}}$$

Operator learning in a nutshell

Operator between function spaces: $\mathcal{G} : \mathcal{X} \to \mathcal{Y}$

Approx. $\mathcal{G}$ by building a parametric map $\hat{\mathcal{G}}_\theta$

E.g., $\hat{\mathcal{G}}_\theta = \mathcal{Q} \circ \sigma(\mathcal{K}_L) \circ \cdots \circ \sigma(\mathcal{K}_1) \circ \mathcal{P}$

Decoder

Nonlinear activation function

Encoder

$$(\mathcal{K}_1 v)(x) = \int_{D_1} \kappa_1(x, y) v(y) dy + b_1(x)$$

Operator learning in a nutshell

Operator between function spaces: $\mathcal{G} : \mathcal{X} \to \mathcal{Y}$

Approx. $\mathcal{G}$ by building a parametric map $\hat{\mathcal{G}}_\theta$

E.g., $\hat{\mathcal{G}}_\theta = \mathcal{Q} \circ \sigma(\mathcal{K}_L) \circ \cdots \circ \sigma(\mathcal{K}_1) \circ \mathcal{P}$

Decoder

Nonlinear activation function

Encoder

$$(\mathcal{K}_1 v)(x) = \int_{D_1} \kappa_1(x, y) v(y) dy + b_1(x)$$

Want to find θ such that $\mathcal{G} \approx \hat{\mathcal{G}}_\theta$ in some sense.

Operator learning in a nutshell

Operator between function spaces: $\mathscr{G} : \mathscr{X} \to \mathscr{Y}$

Approx. $\mathscr{G}$ by building a parametric map $\hat{\mathscr{G}}_\theta$

E.g., $\hat{\mathscr{G}}_\theta = \mathscr{Q} \circ \sigma(\mathscr{K}_L) \circ \cdots \circ \sigma(\mathscr{K}_1) \circ \mathscr{P}$

Decoder

Nonlinear activation function

Encoder

$$(\mathscr{K}_1 v)(x) = \int_{D_1} \kappa_1(x, y) v(y) dy + b_1(x)$$

Want to find θ such that $\mathscr{G} \approx \hat{\mathscr{G}}_\theta$ in some sense.

FNO, GNO [Li, Kovachki, Azizzadenesheli, Liu, Bhattacharya, Stuart, & Anandkumar, 20], MgNO [He, Liu, Xu 23], DeepGreen [Gin, Shea, Brunton & Kutz, 21], DeepONet [Lu, Jin & Karniadakis, 19] IAE-net [Ong, Shen, Yang, 2022], DIMON [Yin, Charon, Brody, Lu, Trayanova, Maggioni, 2024]

Neural operator learning

Usually, we collect input-output data $\{f_i, \mathcal{G}(f_i)\}_{i=1}^N$ and try to solve

$$\inf_\theta \frac{1}{N} \sum_{i=1}^{N} \|\mathcal{G}(f_i) - \hat{\mathcal{G}}_\theta(f_i)\|_{\mathcal{Y}}^2$$

Questions:

Neural operator learning

Usually, we collect input-output data $\{f_i, \mathscr{G}(f_i)\}_{i=1}^N$ and try to solve

$$\inf_\theta \frac{1}{N} \sum_{i=1}^N \|\mathscr{G}(f_i) - \hat{\mathscr{G}}_\theta(f_i)\|_{\mathscr{Y}}^2$$

Questions:

What are the $\mathscr{G}$'s of interest?

Neural operator learning

Usually, we collect input-output data $\{f_i, \mathscr{G}(f_i)\}_{i=1}^N$ and try to solve

$$\inf_\theta \frac{1}{N}\sum_{i=1}^N \|\mathscr{G}(f_i) - \hat{\mathscr{G}}_\theta(f_i)\|_{\mathscr{Y}}^2$$

Questions:

What are the $\mathscr{G}$'s of interest?

How big does N need to be for a certain accuracy?

Neural operator learning

Usually, we collect input-output data $\{f_i, \mathscr{G}(f_i)\}_{i=1}^N$ and try to solve

$$\inf_{\theta} \frac{1}{N} \sum_{i=1}^{N} \|\mathscr{G}(f_i) - \hat{\mathscr{G}}_\theta(f_i)\|_{\mathscr{Y}}^2$$

<u>Questions:</u>

What are the $\mathscr{G}$'s of interest?

How big does N need to be for a certain accuracy?

If N is big enough, then how do I generate the $f_i's$?

Solution operators associated with PDEs

Question:

What are the $\mathscr{G}$'s of interest?

My focus for this talk:

Solution operators associated with PDEs

Solution operators associated with PDEs

Question:

What are the $\mathscr{G}$'s of interest?

My focus for this talk:

Solution operators associated with PDEs

Input-output data: $\{(f_j, u_j)\}_{j=1}^N$ such that $\mathscr{N}(u_j) = f_j, \quad \mathscr{B}(u_j) = 0.$

Solution operators associated with PDEs

Question:

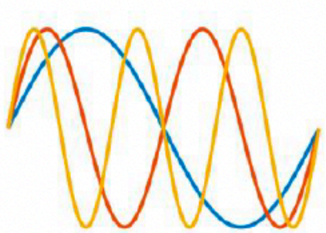
What are the $\mathscr{G}$'s of interest?

My focus for this talk:

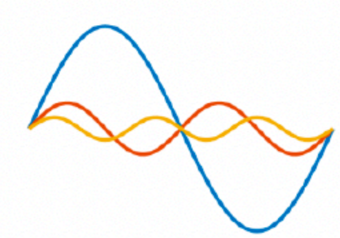
Solution operators associated with PDEs

Input-output data: $\{(f_j, u_j)\}_{j=1}^{N}$ such that $\mathscr{N}(u_j) = f_j, \quad \mathscr{B}(u_j) = 0.$

$$f_j \longrightarrow \boxed{\mathscr{G}} \longrightarrow u_j$$



Forcing functions



PDE solutions

Solution operators associated with PDEs

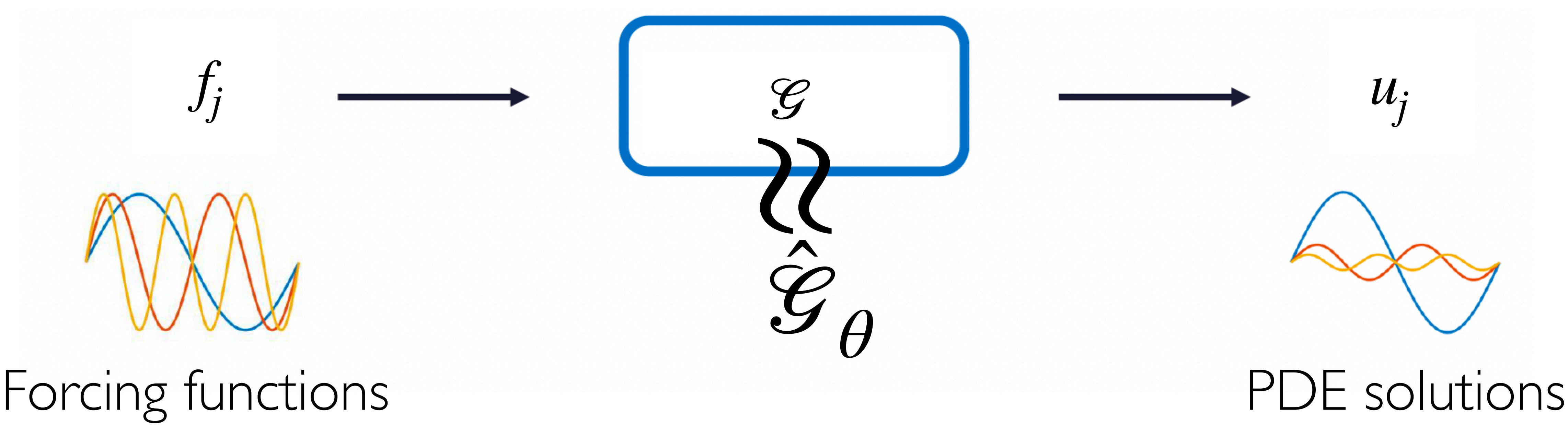
Question:

What are the $\mathcal{G}$'s of interest?

My focus for this talk:

Solution operators associated with PDEs

Input-output data: $\{(f_j, u_j)\}_{j=1}^{N}$ such that $\mathcal{N}(u_j) = f_j, \quad \mathcal{B}(u_j) = 0.$

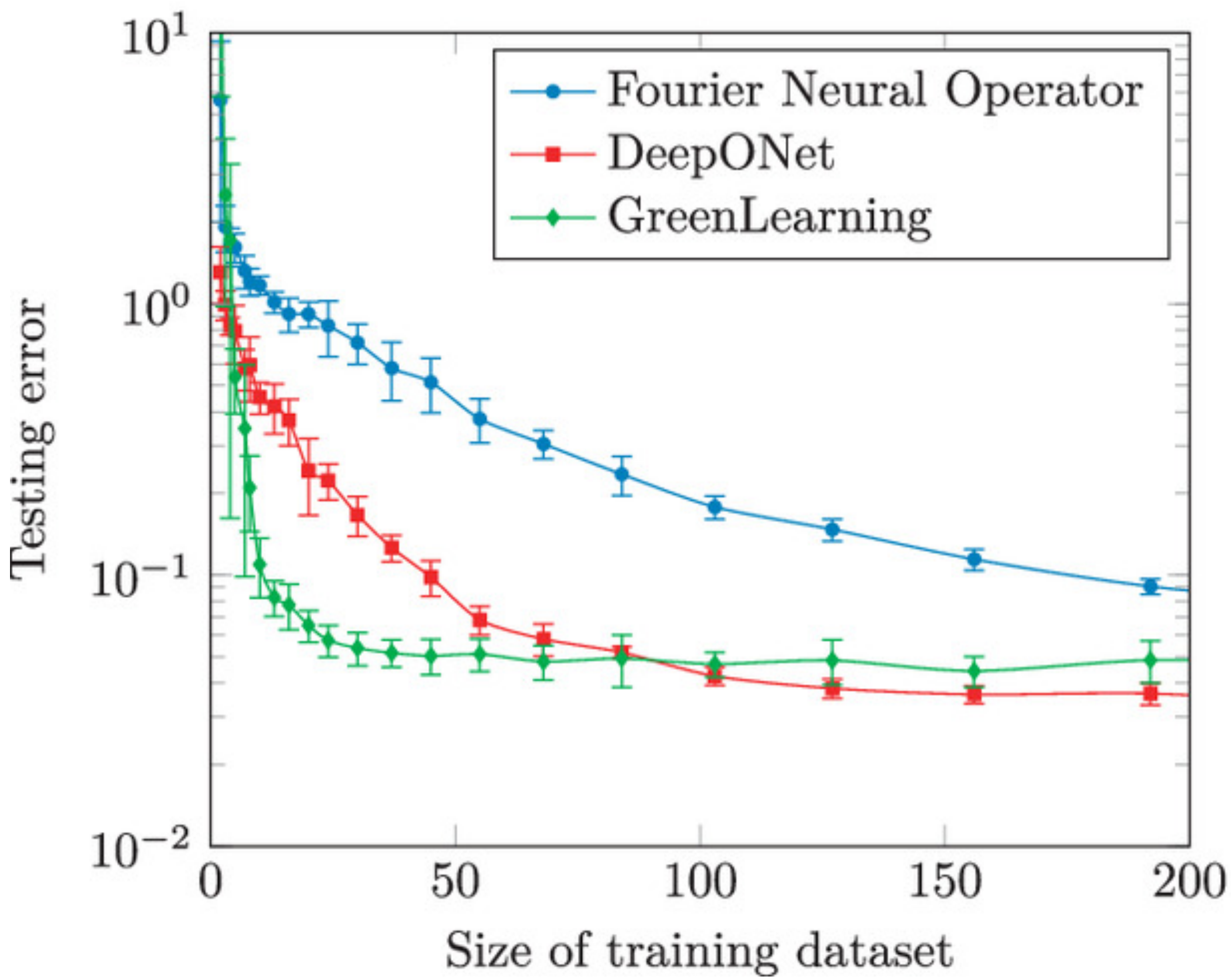


Data-efficient solution operator learning

2D Poisson equation

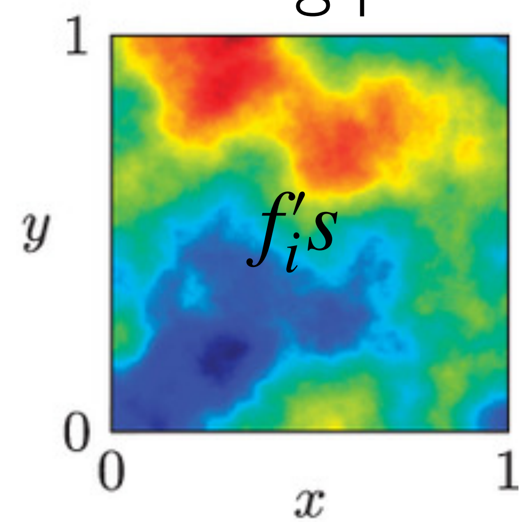
$$\nabla^2 u = f, \qquad u|_{[0,1]^2} = 0$$

Accuracy of the approx. solution operator

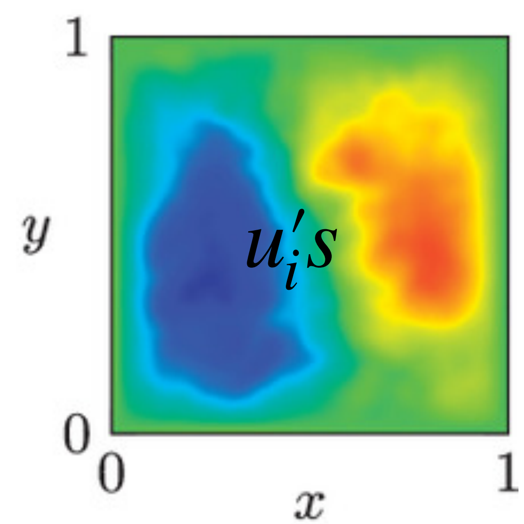


N

Training pairs



Forcing term



Solution

Green's function associated with linear PDEs

Linear PDE

$$\mathscr{L}u = f \text{ on } \Omega \subseteq \mathbb{R}^d$$

$$u|_{\partial\Omega} = 0$$

$\longrightarrow$

Solution operator

$$u(x) = \underbrace{\int_{\Omega} G(x,y) f(y) dy}_{=(\mathscr{G}f)(x)}$$

Green's function associated with linear PDEs

Linear PDE

$$\mathscr{L}u = f \text{ on } \Omega \subseteq \mathbb{R}^d$$

$$u|_{\partial\Omega} = 0$$

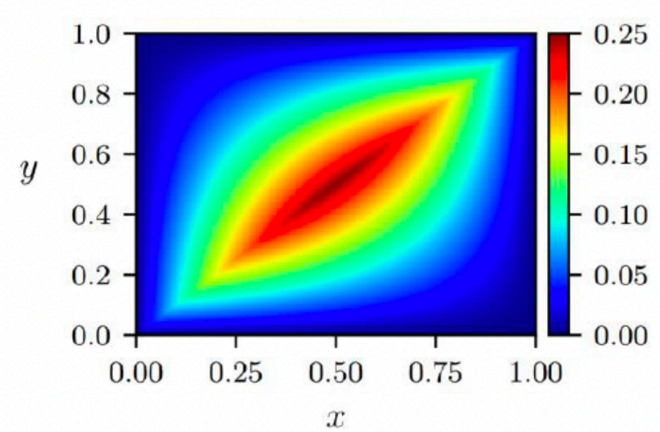
Solution operator

$$u(x) = \underbrace{\int_\Omega G(x,y)f(y)dy}_{=(\mathscr{G}f)(x)}$$

Poisson equation

$$-\nabla^2 u = f$$

$$u(0) = u(1) = 0$$



Green's function for PDEs in $d > 1$ are unbounded functions

Green's function recovery

Theorem: [Boullé & T., 2021], [Boullé, Kim, Tianyi & T., 2022], [Boullé, Hailikas & T., 2023] [Wang & T., 2024]
There is a randomized algorithm that, for any $\epsilon > 0$, can construct an approx. G of $\hat{G}$ for PDE class with $\boxed{??}$ input-output pairs (f_j, u_j) such that

$$\|G - \hat{G}\|_{L^p} \leq \epsilon \|G\|_{L^p}$$

with high probability.

PDE class	$\boxed{??}$	p

Green's function recovery

Theorem: [Boullé & T., 2021], [Boullé, Kim, Tianyi & T., 2022], [Boullé, Hailikas & T., 2023] [Wang & T., 2024]
There is a randomized algorithm that, for any $\epsilon > 0$, can construct an approx. G of $\hat{G}$ for PDE class with $\boxed{??}$ input-output pairs (f_j, u_j) such that

$$\|G - \hat{G}\|_{L^p} \leq \epsilon \|G\|_{L^p}$$

with high probability.

PDE class	$\boxed{??}$	p
uniformly self-adjoint elliptic in $d = 1,2,3$	$\mathcal{O}(\log^{d+2}(1/\epsilon)/\Gamma_\epsilon)$	2

Green's function recovery

Theorem: [Boullé & T., 2021], [Boullé, Kim, Tianyi & T., 2022], [Boullé, Hailikas & T., 2023] [Wang & T., 2024]
There is a randomized algorithm that, for any $\epsilon > 0$, can construct an approx. G of $\hat{G}$ for PDE class with $\boxed{\;??\;}$ input-output pairs (f_j, u_j) such that

$$\|G - \hat{G}\|_{L^p} \leq \epsilon \|G\|_{L^p}$$

with high probability.

PDE class	$\boxed{\;??\;}$	p
uniformly self-adjoint elliptic in $d = 1,2,3$	$\mathcal{O}(\log^{d+2}(1/\epsilon)/\Gamma_\epsilon)$	2
uniformly parabolic in $d \geq 1$ (and uni. self-adjoint elliptic in $d \geq 4$.)	$\mathcal{O}(\log^{d+4}(1/\epsilon)/\Gamma_\epsilon)$	1

Green's function recovery

Theorem: [Boullé & T., 2021], [Boullé, Kim, Tianyi & T., 2022], [Boullé, Hailikas & T., 2023] [Wang & T., 2024]
There is a randomized algorithm that, for any $\epsilon > 0$, can construct an approx. G of $\hat{G}$ for PDE class with $\boxed{\quad ?? \quad}$ input-output pairs (f_j, u_j) such that

$$\|G - \hat{G}\|_{L^p} \leq \epsilon \|G\|_{L^p}$$

with high probability.

PDE class	$\boxed{\quad ?? \quad}$	p
uniformly self-adjoint elliptic in $d = 1,2,3$	$\mathcal{O}(\log^{d+2}(1/\epsilon)/\Gamma_\epsilon)$	2
uniformly parabolic in $d \geq 1$ (and uni. self-adjoint elliptic in $d \geq 4$.)	$\mathcal{O}(\log^{d+4}(1/\epsilon)/\Gamma_\epsilon)$	1
uniformly self-adjoint hyperbolic in $d = 1$	$\mathcal{O}(\epsilon^{-(6+1/r)}\log^{3}(1/\epsilon)/\Gamma_\epsilon)$	2

Recovering a matrix with matrix-vector products

We can recover a symmetric low-rank matrix with matrix-vector products $v \mapsto Av$:

Randomized SVD:

[Halko, Martinsson, & Tropp, 2011], [Martinsson & Tropp, 2020]

Recovering a matrix with matrix-vector products

We can recover a symmetric low-rank matrix with matrix-vector products $v \mapsto Av$:

Randomized SVD:

1 $n \times (k+5)$

$Y =$ 

Tall-skinny Gaussian matrix with iid indep. entries

[Halko, Martinsson, & Tropp, 2011], [Martinsson & Tropp, 2020]

Recovering a matrix with matrix-vector products

We can recover a symmetric low-rank matrix with matrix-vector products $v \mapsto Av$:

Randomized SVD:

1 $n \times (k+5)$

$Y =$ 

Tall-skinny Gaussian matrix
with iid indep. entries

2

$Z = AY$
Input-output data

[Halko, Martinsson, & Tropp, 2011], [Martinsson & Tropp, 2020]

Recovering a matrix with matrix-vector products

We can recover a symmetric low-rank matrix with matrix-vector products $v \mapsto Av$:

Randomized SVD:

1 $n \times (k+5)$

$Y =$ 

Tall-skinny Gaussian matrix with iid indep. entries

2

$Z = AY$

Input-output data

3

$Q = \mathrm{orth}(Z)$

orthonormal basis for $\mathrm{col}(Z)$

$A_k = QQ^*A$

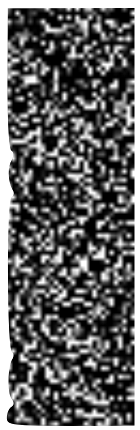
[Halko, Martinsson, & Tropp, 2011], [Martinsson & Tropp, 2020]

Recovering a matrix with matrix-vector products

We can recover a symmetric low-rank matrix with matrix-vector products $v \mapsto Av$:

Randomized SVD:

1. $n \times (k+5)$

$Y =$ 

Tall-skinny Gaussian matrix
with iid indep. entries

2. $Z = AY$
Input-output data

3. $Q = \mathrm{orth}(Z)$
orthonormal basis
for $\mathrm{col}(Z)$
$A_k = QQ^*A$

[Halko, Martinsson, & Tropp, 2011], [Martinsson & Tropp, 2020]

Theorem (Halko, Martinsson, Tropp, 2011).

We can construct an approximation A_k of A from k+5 random input vectors such that

$$\mathbb{P}\left[\|A - A_k\|_\mathrm{F} \leq (1 + 15\sqrt{k+5})\epsilon_k\right] \geq 0.999$$

Recovering a matrix with matrix-vector products

We can recover a symmetric low-rank matrix with matrix-vector products $v \mapsto Av$:

Randomized SVD:

1 $n \times (k+5)$

$Y =$ 

Tall-skinny Gaussian matrix
with iid indep. entries

2

$Z = AY$
Input-output data

3

$Q = \mathrm{orth}(Z)$
orthonormal basis
for $\mathrm{col}(Z)$

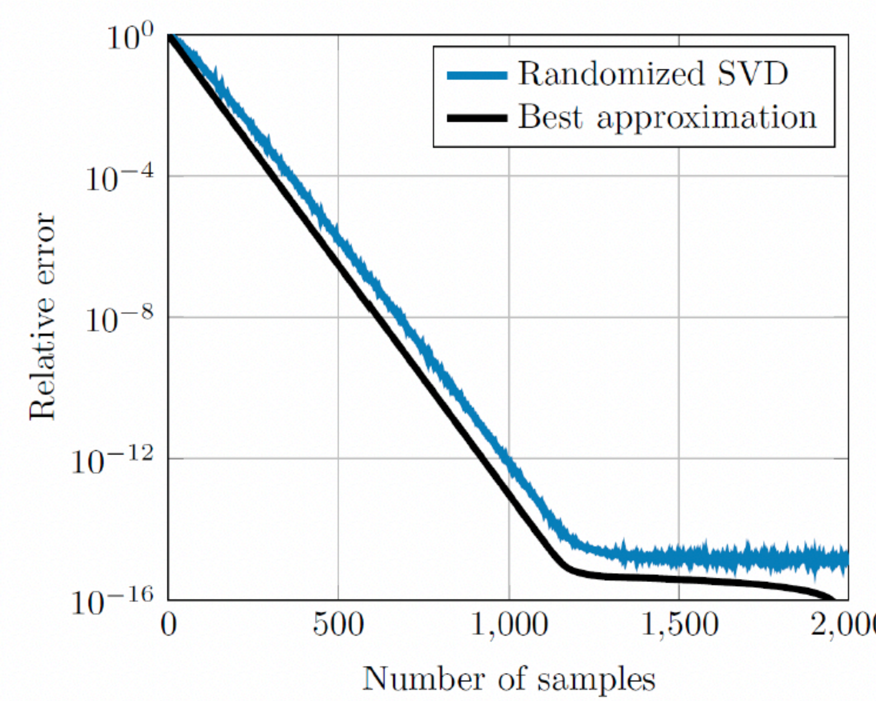
$A_k = QQ^*A$

[Halko, Martinsson, & Tropp, 2011], [Martinsson & Tropp, 2020]

Theorem (Halko, Martinsson, Tropp, 2011).

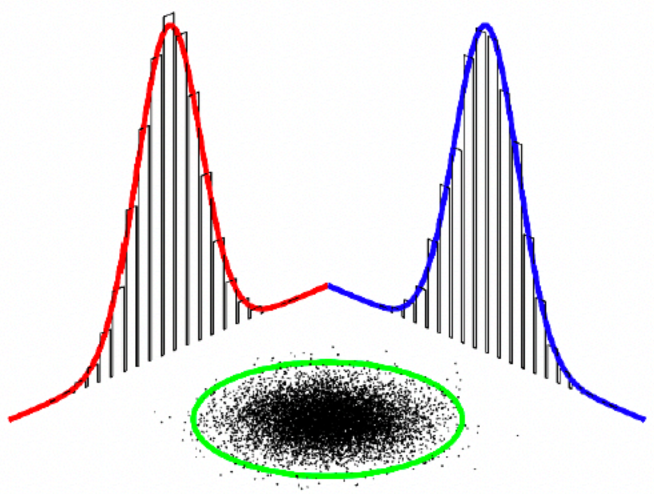
We can construct an approximation A_k of A from k+5 random input vectors such that

$$\mathbb{P}\left[\|A - A_k\|_{\mathrm{F}} \leq (1 + 15\sqrt{k+5})\epsilon_k\right] \geq 0.999$$

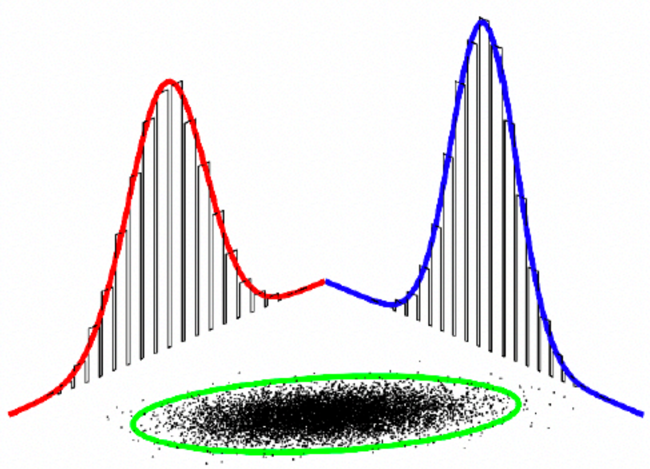


Generalization of the randomized SVD

Standard Gaussian vectors



Correlated Gaussian vectors



Theorem [Boullé & T., 2021]

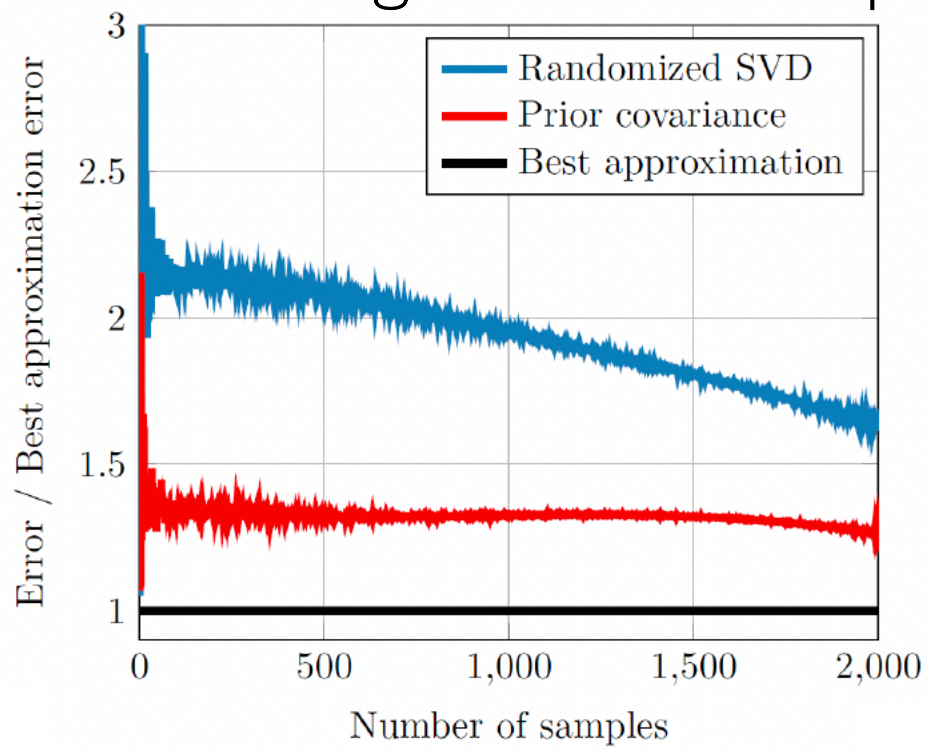
We can construct an approximation A_k of A from k+5 correlated random input vectors such that

$$\mathbb{P}\left[\|A - A_k\|_{\mathrm{F}} \leq (1 + 18\sqrt{k/\gamma_k})\epsilon_k\right] \geq 0.999$$

Generalization of the randomized SVD

Standard Gaussian vectors → **Correlated Gaussian vectors**

Prior knowledge about A helps:



Theorem [Boullé & T., 2021]

We can construct an approximation A_k of A from k+5 correlated random input vectors such that

$$\mathbb{P}\left[\|A - A_k\|_{\mathrm{F}} \leq (1 + 18\sqrt{k/\gamma_k})\epsilon_k\right] \geq 0.999$$

Randomized SVD for Green's functions

We can learn kernel in a self-adjoint HS integral operator $f \mapsto \int_\Omega G(x,y)f(y)dy$:

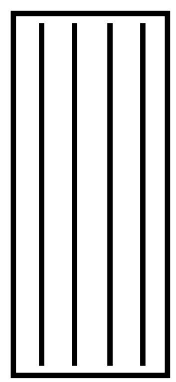
Randomized SVD for HS operators:

Randomized SVD for Green's functions

We can learn kernel in a self-adjoint HS integral operator $f \mapsto \int_\Omega G(x,y)f(y)dy$:

Randomized SVD for HS operators:

1 $\quad \Omega \times (k+5)$

$Y =$ 

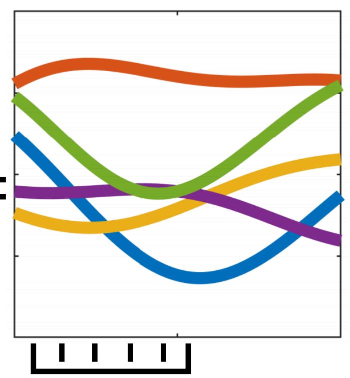
Cols are drawn from
Gaussian process $GP(0,C)$

Randomized SVD for Green's functions

We can learn kernel in a self-adjoint HS integral operator $f \mapsto \int_{\Omega} G(x,y)f(y)dy$:

Randomized SVD for HS operators:

1 $\Omega \times (k+5)$

$Y =$ 

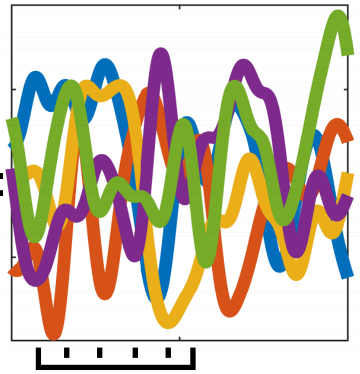
Cols are drawn from
Gaussian process $GP(0,C)$

Randomized SVD for Green's functions

We can learn kernel in a self-adjoint HS integral operator $f \mapsto \int_\Omega G(x,y)f(y)dy$:

Randomized SVD for HS operators:

1 $\Omega \times (k+5)$

$Y =$ 

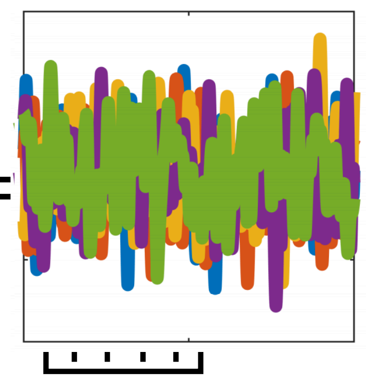
Cols are drawn from
Gaussian process $GP(0,C)$

Randomized SVD for Green's functions

We can learn kernel in a self-adjoint HS integral operator $f \mapsto \int_\Omega G(x,y)f(y)dy$:

Randomized SVD for HS operators:

1 $\Omega \times (k+5)$

$Y =$ 

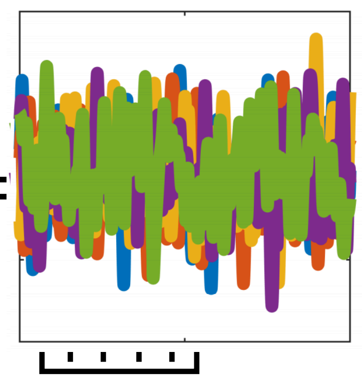
Cols are drawn from
Gaussian process $GP(0,C)$

Randomized SVD for Green's functions

We can learn kernel in a self-adjoint HS integral operator $f \mapsto \int_\Omega G(x, y)f(y)dy$:

Randomized SVD for HS operators:

1 $\Omega \times (k+5)$

$Y =$ 

Cols are drawn from Gaussian process $GP(0, C)$

2 $Z_i(x) = \int_\Omega G(x, y)Y_i(y)dy$

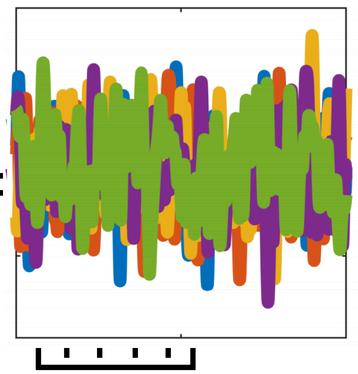
Input-output data

Randomized SVD for Green's functions

We can learn kernel in a self-adjoint HS integral operator $f \mapsto \int_\Omega G(x,y)f(y)dy$:

Randomized SVD for HS operators:

1 $\Omega \times (k+5)$

$Y =$ 

Cols are drawn from Gaussian process $GP(0,C)$

2 $Z_i(x) = \int_\Omega G(x,y)Y_i(y)dy$

Input-output data

3 $Q = \mathrm{orth}(Z)$

orthonormal basis for $\mathrm{col}(Z)$

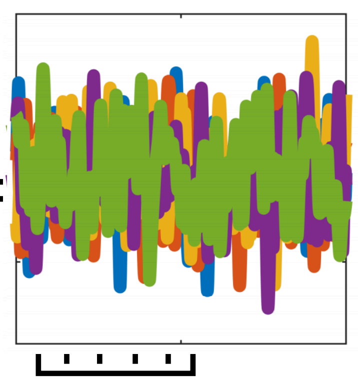
"$G_k = QQ^*G$"

Randomized SVD for Green's functions

We can learn kernel in a self-adjoint HS integral operator $f \mapsto \int_\Omega G(x,y)f(y)dy$:

Randomized SVD for HS operators:

1 $\Omega \times (k+5)$

$Y =$ 

Cols are drawn from Gaussian process $GP(0,C)$

2 $Z_i(x) = \int_\Omega G(x,y)Y_i(y)dy$

Input-output data

3 $Q = \mathrm{orth}(Z)$

orthonormal basis for $\mathrm{col}(Z)$

"$G_k = QQ^*G$"

Theorem [Boullé & T., 2022]

We can construct an approximation G_k of G from k+5 random input functions f such that

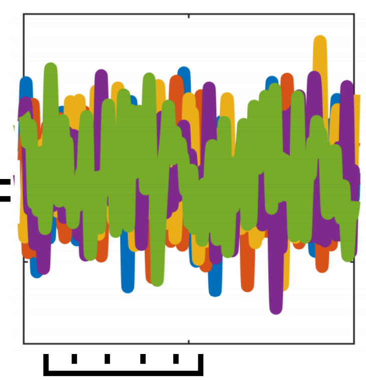
$$\mathbb{P}\left[\|G - G_k\|_{L^2} \leq \mathcal{O}\left(\sqrt{k^2/\gamma_k}\right)\epsilon_k\right] \geq 0.999$$

Randomized SVD for Green's functions

We can learn kernel in a self-adjoint HS integral operator $f \mapsto \int_\Omega G(x,y)f(y)dy$:

Randomized SVD for HS operators:

1 $\Omega \times (k+5)$

$Y =$ 

Cols are drawn from Gaussian process $GP(0,C)$

2 $Z_i(x) = \int_\Omega G(x,y)Y_i(y)dy$

Input-output data

3 $Q = \mathrm{orth}(Z)$

orthonormal basis for $\mathrm{col}(Z)$

"$G_k = QQ^*G$"

Theorem [Boullé & T., 2022]

We can construct an approximation G_k of G from k+5 random input functions f such that

$$\mathbb{P}\left[\|G - G_k\|_{L^2} \leq \mathcal{O}\left(\sqrt{k^2/\gamma_k}\right)\epsilon_k\right] \geq 0.999$$

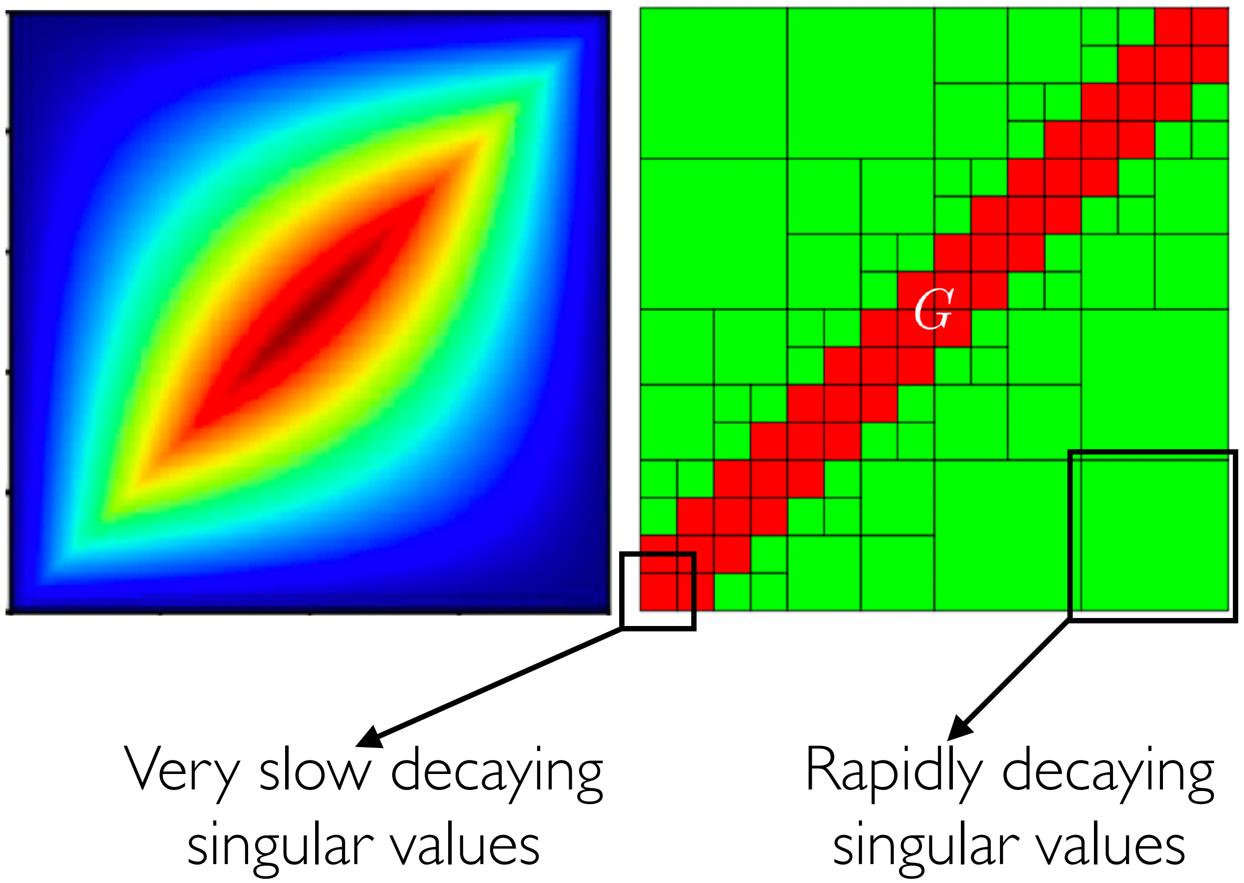
Problem:

Green's functions typically do not have rapidly decaying singular values.

ϵ_k decays very slowly with k

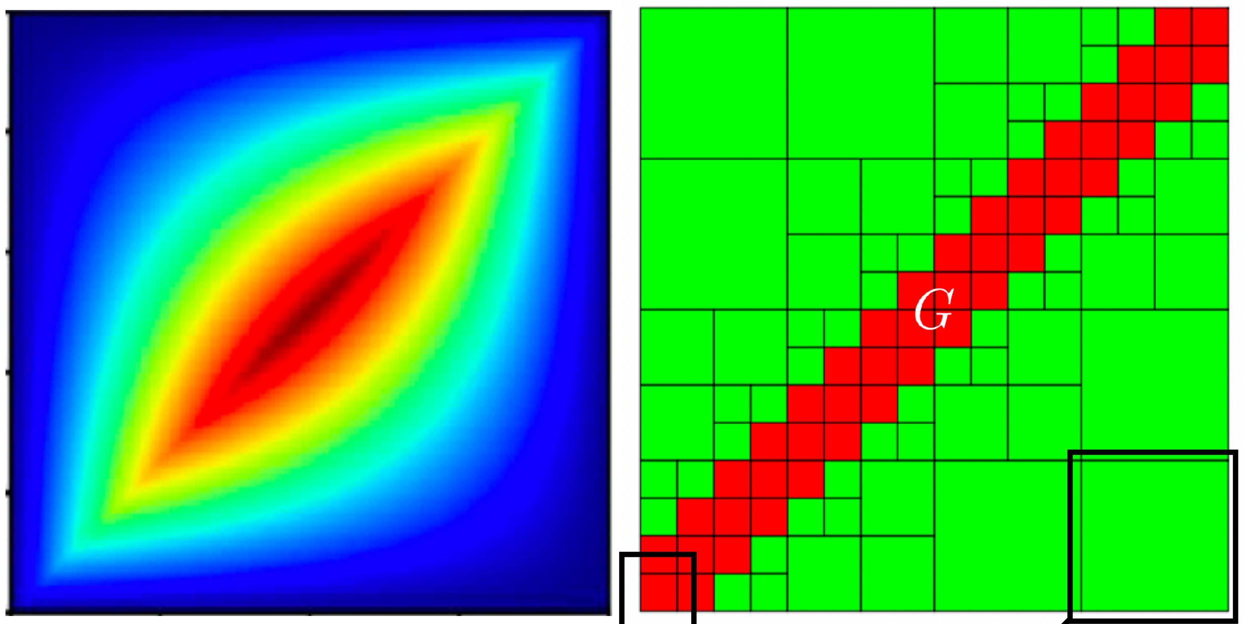
Green's functions are low rank on separated blocks

One dimension



Green's functions are low rank on separated blocks

One dimension

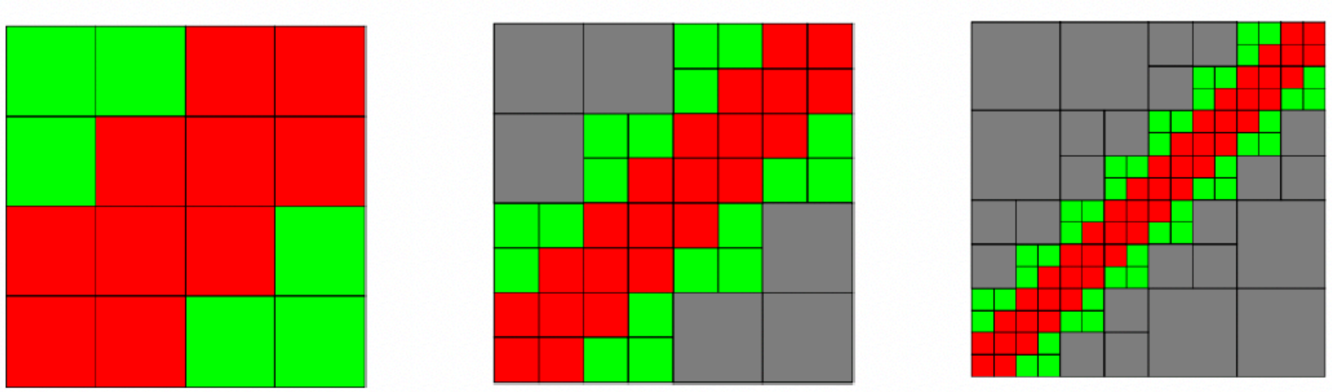


Very slow decaying singular values

Rapidly decaying singular values

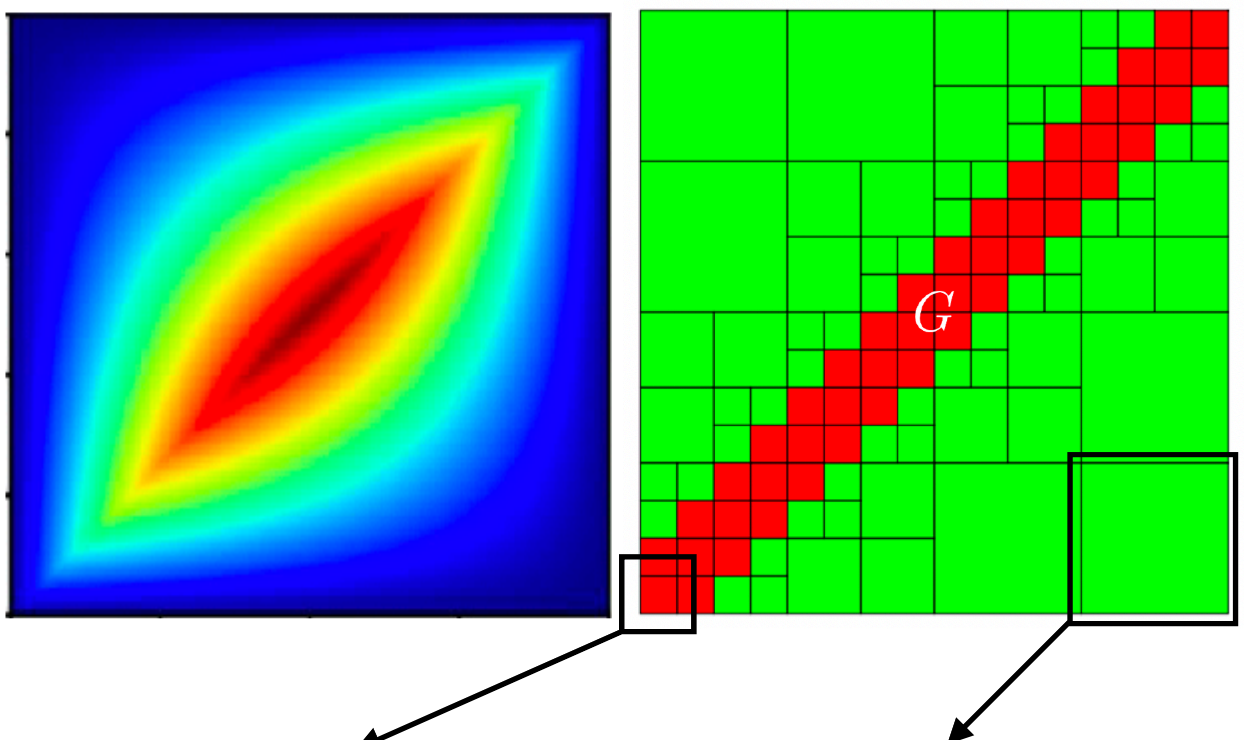
Hierarchical structure

Level 2 Level 3 Level 4



Green’s functions are low rank on separated blocks

One dimension

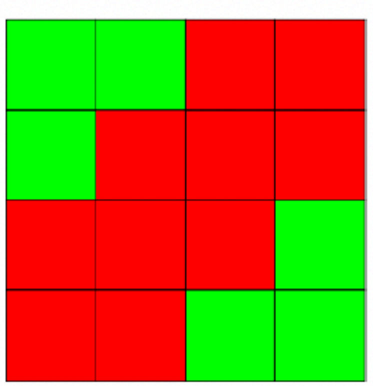


Very slow decaying singular values

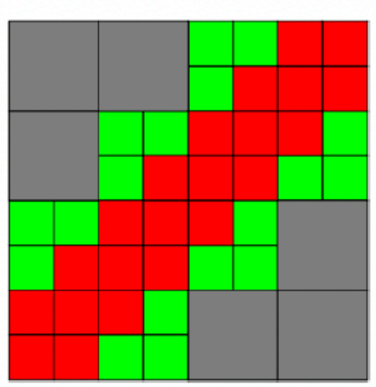
Rapidly decaying singular values

<u>Hierarchical structure</u>

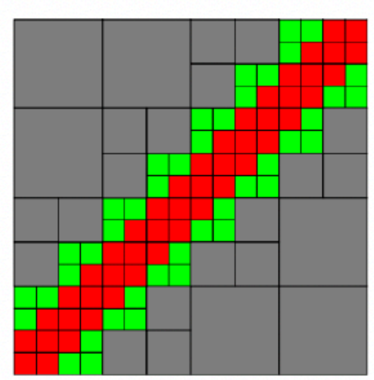
Level 2



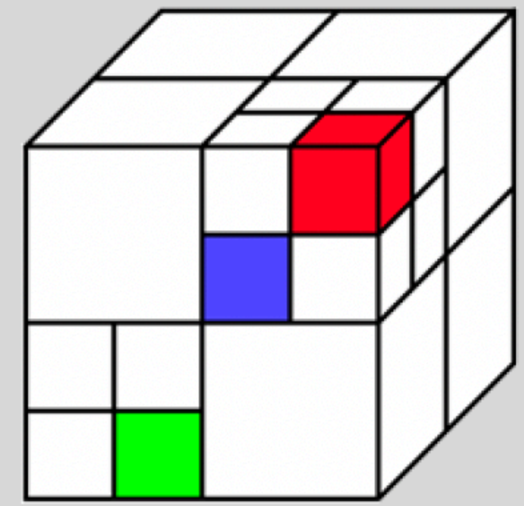
Level 3



Level 4



Three dimensions

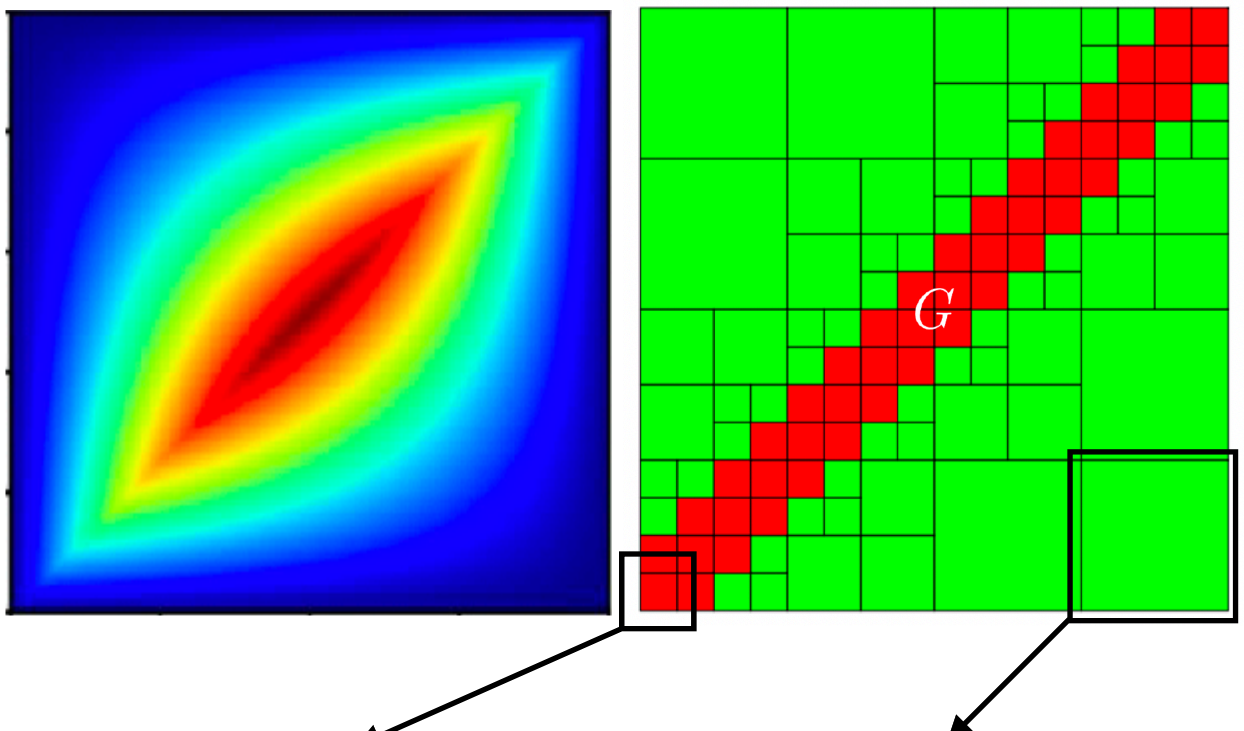


Low-rank structure on well separated domains.

[Bebendorf, Hackbush, 2003]

Green’s functions are low rank on separated blocks

One dimension

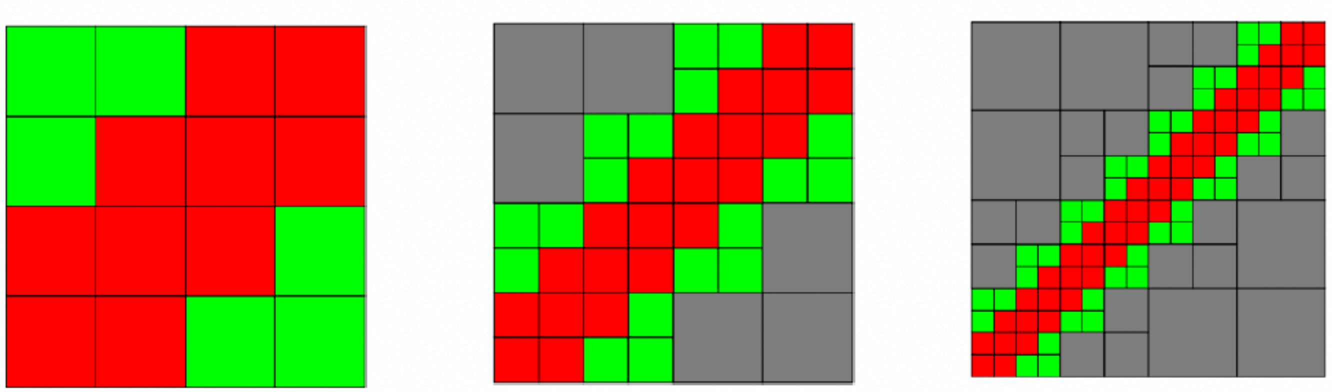


Very slow decaying singular values

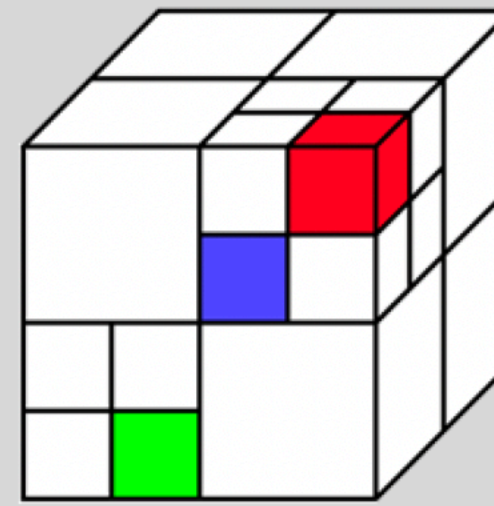
Rapidly decaying singular values

Hierarchical structure

Level 2 Level 3 Level 4

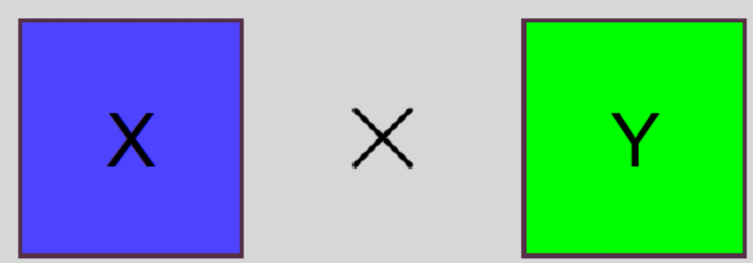


Three dimensions



Low-rank structure on well separated domains.

[Bebendorf, Hackbush, 2003]



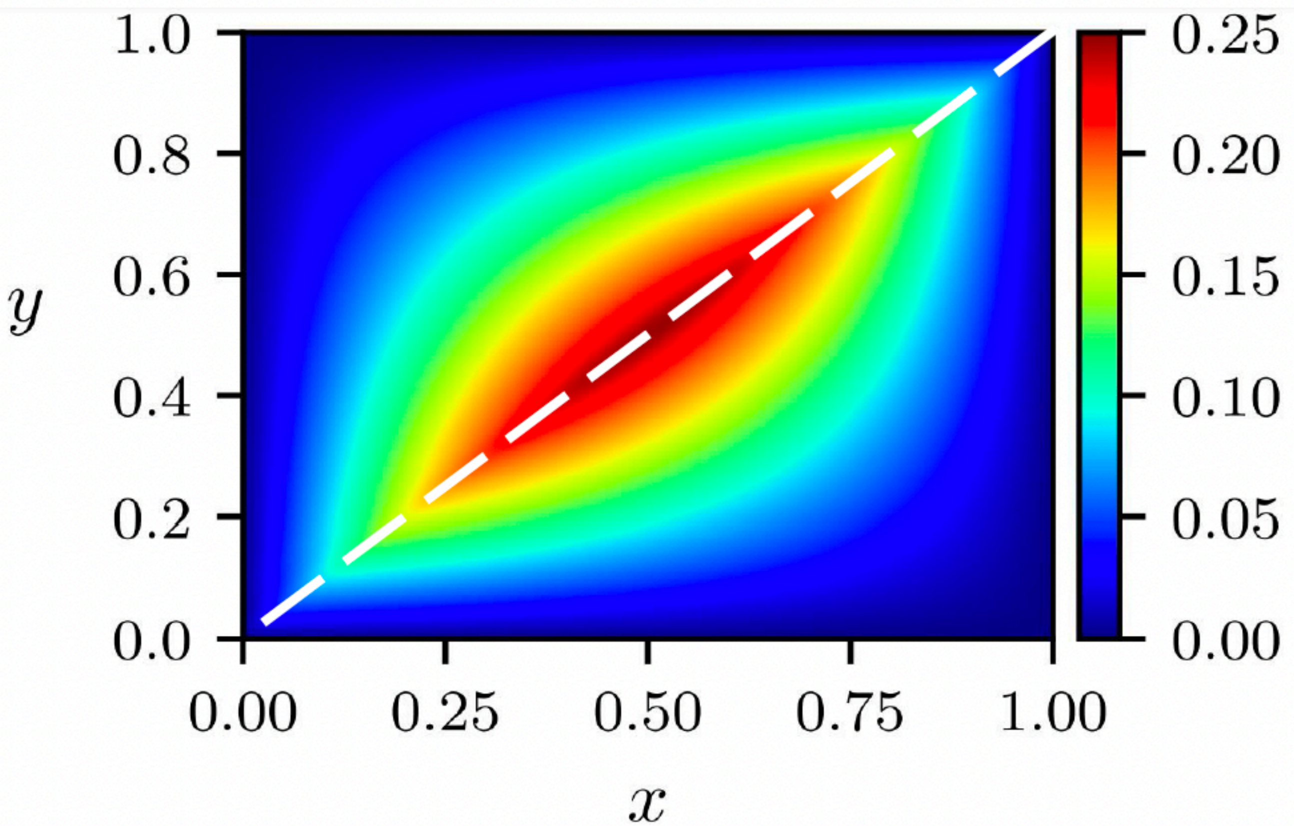
Related approaches for matrices:

[Martinsson, 2008], [Lin, Lu, & Ying, 2010], [Martinsson, 2016], [Levitt & Martinsson, 2022]

Off-diagonal decay

Green's function of the Laplace operator:

$$-\nabla^2 u = f$$

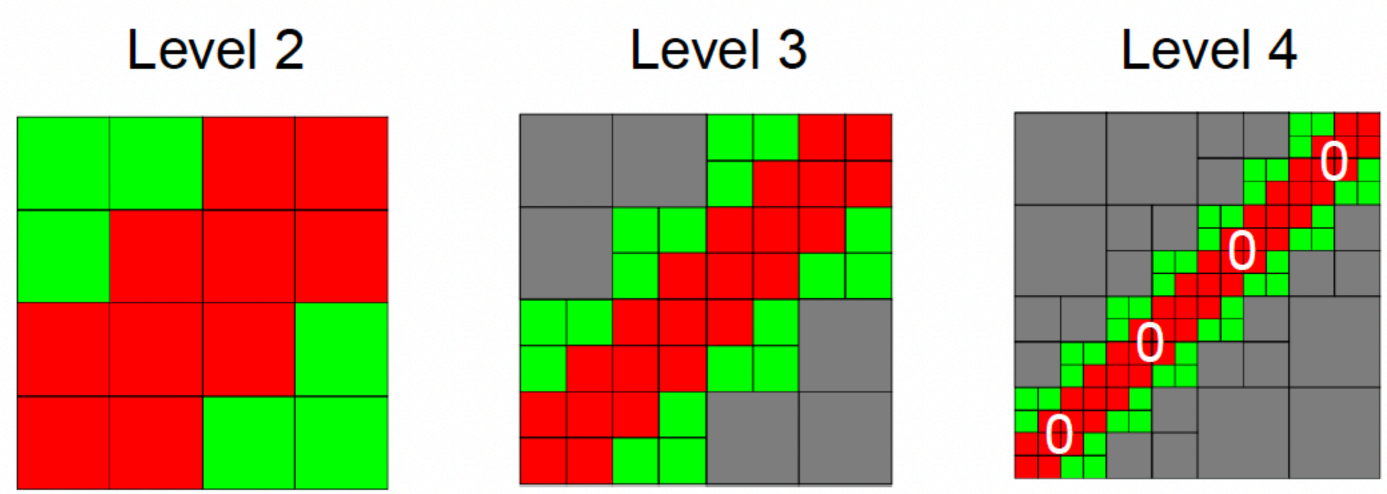


Green's functions are smooth and decay off the diagonal. [Grüter, Widman, 1982]

$$G(x,y) \leq \frac{1}{\|x-y\|}$$

(for 3D elliptic PDEs)

<u>Hierarchical structure</u>

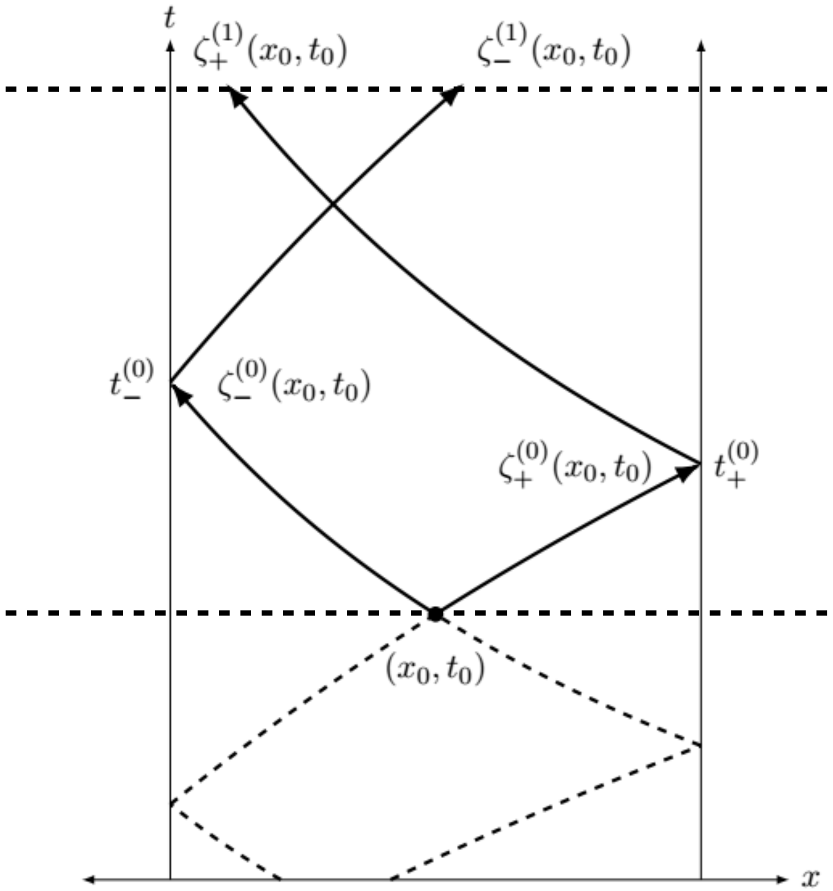


(Pictures are in 1D for illustration purposes.)

Green's functions associated with 1D hyperbolic PDEs

[Wang & T., 2024]

Solution operators for 1D hyperbolic PDEs have Green's functions with jumps along characteristics.



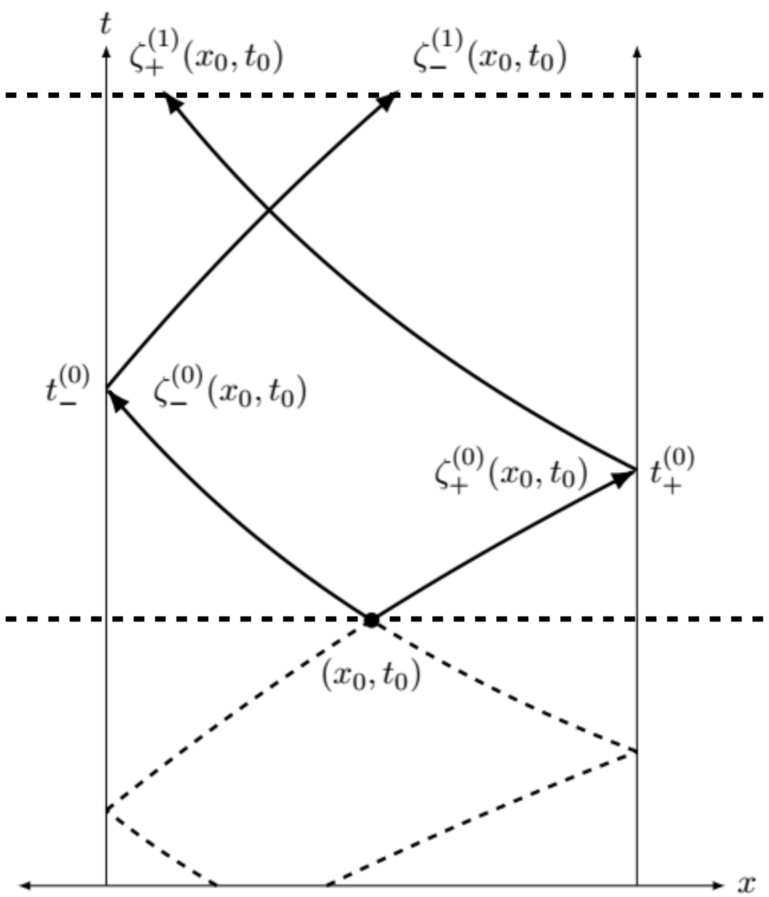
2D slice through the 4D Green's function

Chris Wang

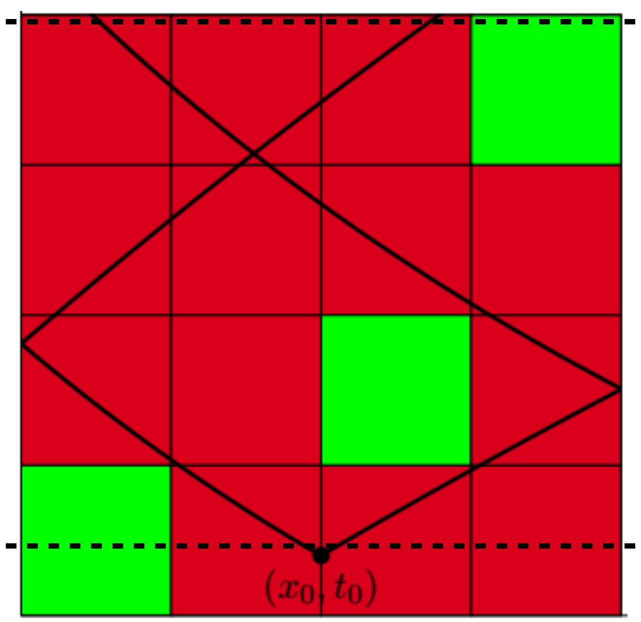
Green's functions associated with 1D hyperbolic PDEs

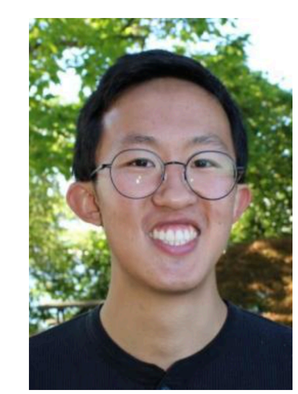
[Wang & T., 2024]

Solution operators for 1D hyperbolic PDEs have Green's functions with jumps along characteristics.



2D slice through the 4D Green's function



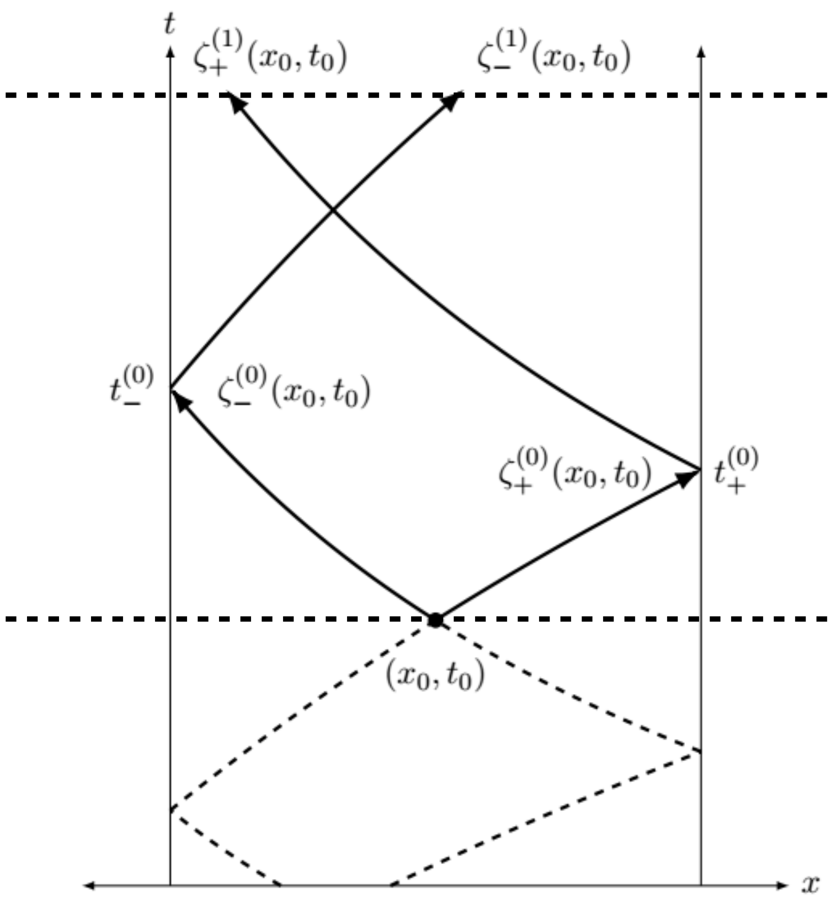


Chris Wang

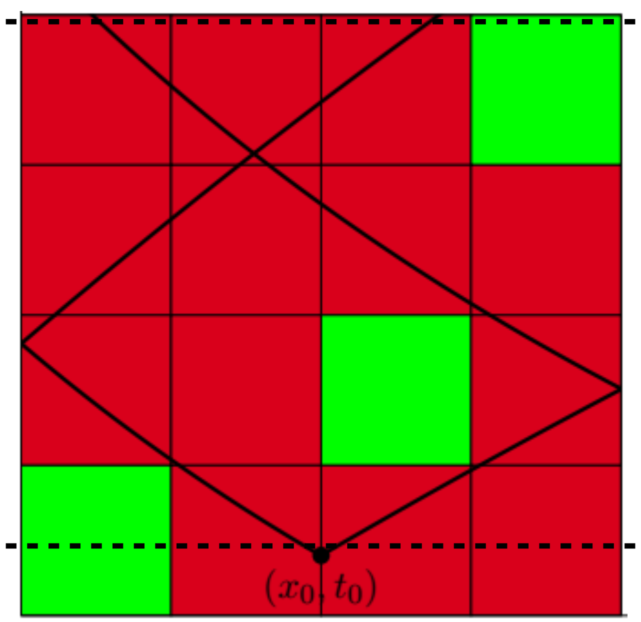
Green's functions associated with 1D hyperbolic PDEs

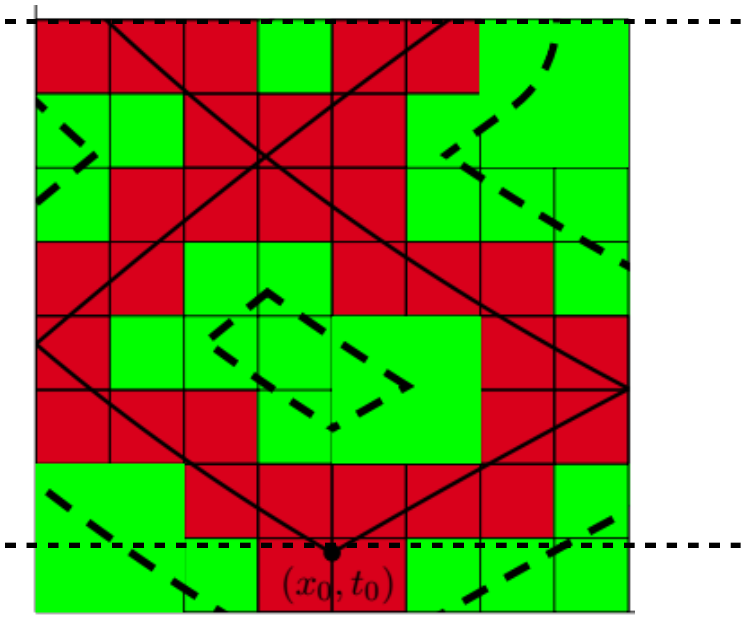
[Wang & T., 2024]

Solution operators for 1D hyperbolic PDEs have Green's functions with jumps along characteristics.



2D slice through the 4D Green's function



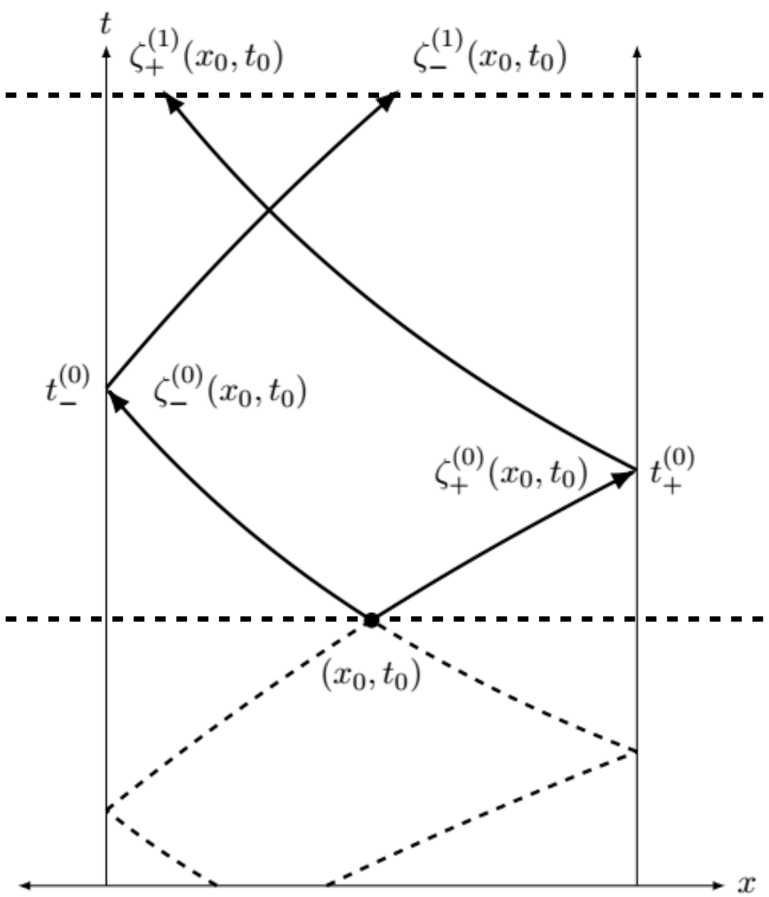


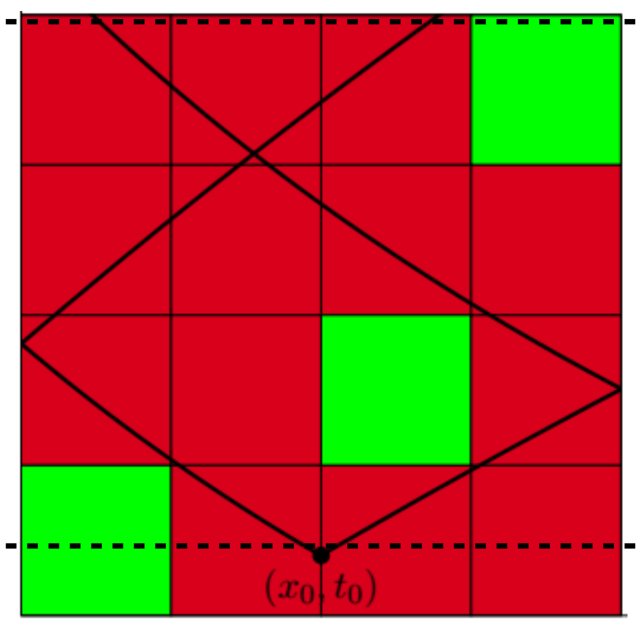
Chris Wang

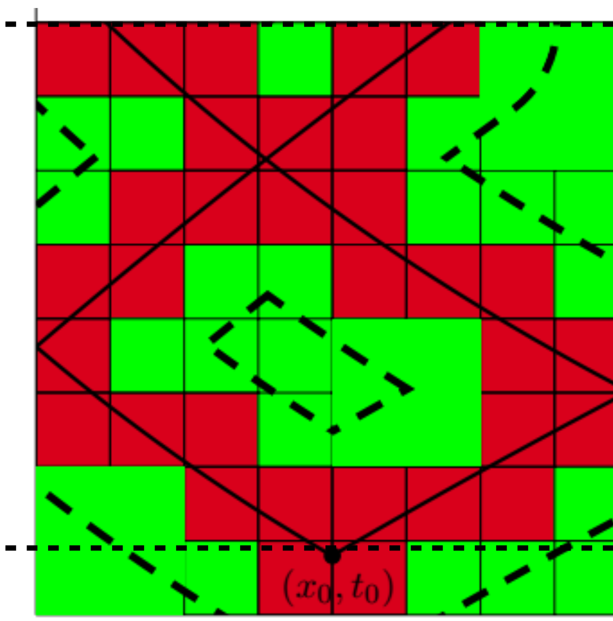
Green's functions associated with 1D hyperbolic PDEs

[Wang & T., 2024]

Solution operators for 1D hyperbolic PDEs have Green's functions with jumps along characteristics.



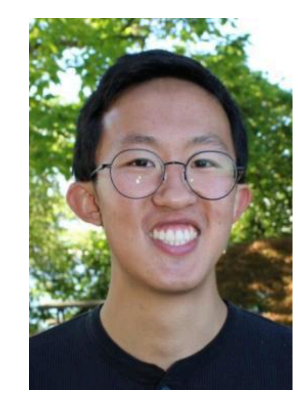




2D slice through the 4D Green's function

Using input-output data to:

1. Adaptively partition domain to isolate characteristics in tiny regions
2. Recover Green's function off the characteristics



Chris Wang

Green's function recovery

Theorem: [Boullé & T., 2021], [Boullé, Kim, Tianyi & T., 2022], [Boullé, Hailikas & T., 2023] [Wang & T., 2024]
There is a randomized algorithm that, for any $\epsilon > 0$, can construct an approx. G of $\hat{G}$ for PDE class with $\boxed{??}$ input-output pairs (f_j, u_j) such that

$$\|G - \hat{G}\|_{L^p} \leq \epsilon \|G\|_{L^p}$$

with high probability.

PDE class	$\boxed{??}$	p
uniformly self-adjoint elliptic in $d = 1,2,3$	$\mathcal{O}(\log^{d+2}(1/\epsilon)/\Gamma_\epsilon)$	2
uniformly parabolic in $d \geq 1$ (and uni. self-adjoint elliptic in $d \geq 4$.)	$\mathcal{O}(\log^{d+4}(1/\epsilon)/\Gamma_\epsilon)$	1
uniformly self-adjoint hyperbolic in $d = 1$	$\mathcal{O}(\epsilon^{-(6+1/r)}\log^3(1/\epsilon)/\Gamma_\epsilon)$	2

Quality of training data

In our theoretical results, Γ_ϵ is a measure of the quality of the training data.

Theorem

We can construct an approximation G_k of G from k+5 random input functions f such that

$$\mathbb{P}\left[\|G - G_k\|_{L^2} \leq \mathcal{O}\left(\sqrt{k^2/\gamma_k}\right)\epsilon_k\right] \geq 0.999$$

$$f \sim \mathcal{GP}(0, K)$$

where $K(x, y)$ is the covariance kernel

Definition: $\gamma_k = k/(\lambda_1 \mathrm{Tr}(\mathbf{C}^{-1}))$

$$\mathbf{C}_{ij} = \int_{\Omega\times\Omega} v_i(x)K(x,y)v_j(y)\,\mathrm{d}x\,\mathrm{d}y$$

where v_i is the ith right singular vectors of G.

- $0 < \gamma_k \leq 1$
- We can impose prior knowledge on the covariance kernel
- Explicit bounds for the covariance quality factor are available

Operator learning without the adjoint

Question:

Can operator learning be data-efficient with only input-output $\{f_i, \mathcal{G}(f_i)\}_{i=1}^N$ data?

Operator learning with and without the adjoint

Consider

$$(\mathcal{G}f) = \int_0^1 G(x, y)f(y)dy, \text{ where } G \text{ is a 1-Lipschitz smooth function}$$

$$\ldots\text{and } G(x, y) = g(x)h(y)$$

Operator learning with and without the adjoint

Consider

$$(\mathcal{G}f) = \int_0^1 G(x,y)f(y)dy,$$ where G is a 1-Lipschitz smooth function

…and $G(x,y) = g(x)h(y)$

Then, $$(\mathcal{G}f)(x) = \left(\int_0^1 h(y)f(y)dy\right)g(x)$$

Operator learning with and without the adjoint

Consider

$$(\mathscr{G}f) = \int_0^1 \boldsymbol{G}(x,y)f(y)dy, \text{ where } \boldsymbol{G} \text{ is a 1-Lipschitz smooth function}$$

$$\ldots\text{and } \boldsymbol{G}(x,y) = g(x)h(y)$$

Then, $(\mathscr{G}f)(x) = \left(\int_0^1 h(y)f(y)dy\right)g(x)$

The adjoint is $(\mathscr{G}^*f)(x) = \left(\int_0^1 g(y)f(y)dy\right)h(x)$

Operator learning with and without the adjoint

Consider

$$(\mathcal{G}f) = \int_0^1 G(x,y)f(y)dy, \text{ where } G \text{ is a 1-Lipschitz smooth function}$$

…and $G(x,y) = g(x)h(y)$

Then, $(\mathcal{G}f)(x) = \left(\int_0^1 h(y)f(y)dy\right)g(x)$

The adjoint is $(\mathcal{G}^*f)(x) = \left(\int_0^1 g(y)f(y)dy\right)h(x)$

Training dataset size to achieve ϵ accuracy

Operator learning with and without the adjoint

Consider

$$(\mathscr{G}f) = \int_0^1 G(x,y)f(y)dy, \text{ where } \boldsymbol{G} \text{ is a 1-Lipschitz smooth function}$$

$$\ldots\text{and } \boldsymbol{G}(x,y) = g(x)h(y)$$

Then, $$(\mathscr{G}f)(x) = \left(\int_0^1 h(y)f(y)dy\right) g(x)$$

The adjoint is $$(\mathscr{G}^*f)(x) = \left(\int_0^1 g(y)f(y)dy\right) h(x)$$

	With the adjoint	Without the adjoint
Training dataset size to achieve ϵ accuracy		

Operator learning with and without the adjoint

Consider

$$(\mathscr{G}f) = \int_0^1 G(x,y)f(y)dy,$$ where G is a 1-Lipschitz smooth function

…and $G(x,y) = g(x)h(y)$

Then, $(\mathscr{G}f)(x) = \left(\int_0^1 h(y)f(y)dy\right)g(x)$

The adjoint is $(\mathscr{G}^*f)(x) = \left(\int_0^1 g(y)f(y)dy\right)h(x)$

	With the adjoint	Without the adjoint
Training dataset size to achieve ϵ accuracy	$\mathscr{O}(1)$ Input-output pairs	

Operator learning with and without the adjoint

Consider

$$(\mathscr{G}f) = \int_0^1 G(x, y)f(y)dy, \text{ where } \boldsymbol{G} \text{ is a 1-Lipschitz smooth function}$$

$$\ldots\text{and } \boldsymbol{G}(x, y) = g(x)h(y)$$

$$\text{Then, } (\mathscr{G}f)(x) = \left(\int_0^1 h(y)f(y)dy\right)g(x)$$

$$\text{The adjoint is } (\mathscr{G}^*f)(x) = \left(\int_0^1 g(y)f(y)dy\right)h(x)$$

	With the adjoint	Without the adjoint
Training dataset size to achieve ϵ accuracy	$\mathcal{O}(1)$ Input-output pairs	$\mathcal{O}(1/\epsilon)$ Input-output pairs

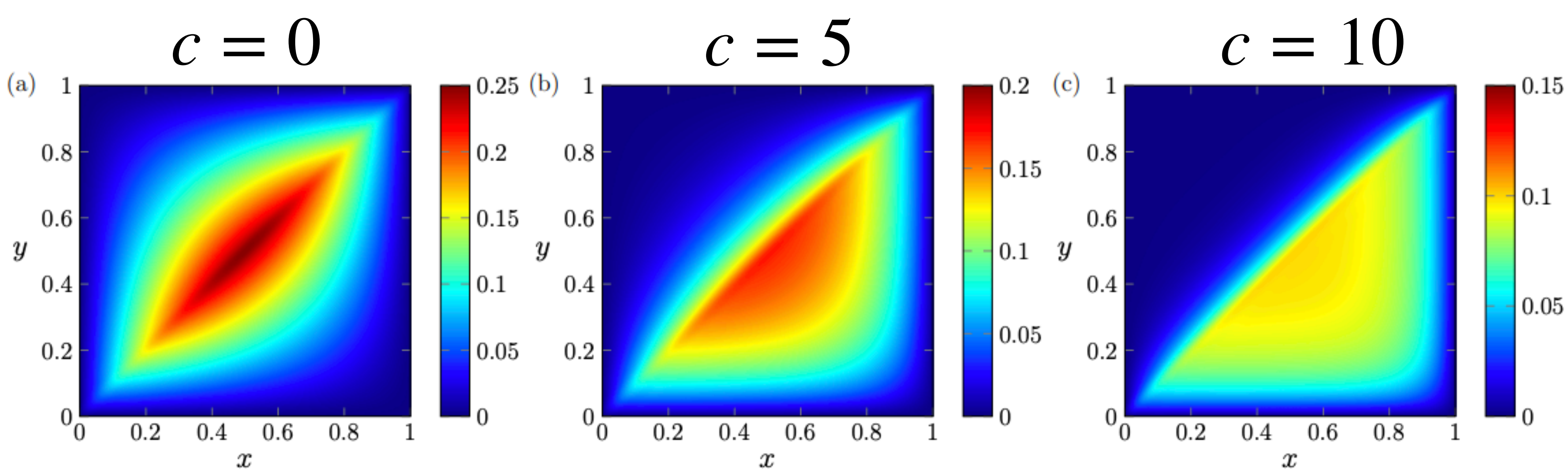
[Halikias & T., 22]

The adjoint mystery

[Boullé, Halikias, Otto & T., 2024], [Levitt & Martinsson, 2024]

Forcing terms: N input-output functions drawn from a Gaussian process.

$$-\frac{d^2u}{dx^2} + c\frac{du}{dx} = f, \quad u(0) = u(1) = 0, \quad x \in [0,1].$$



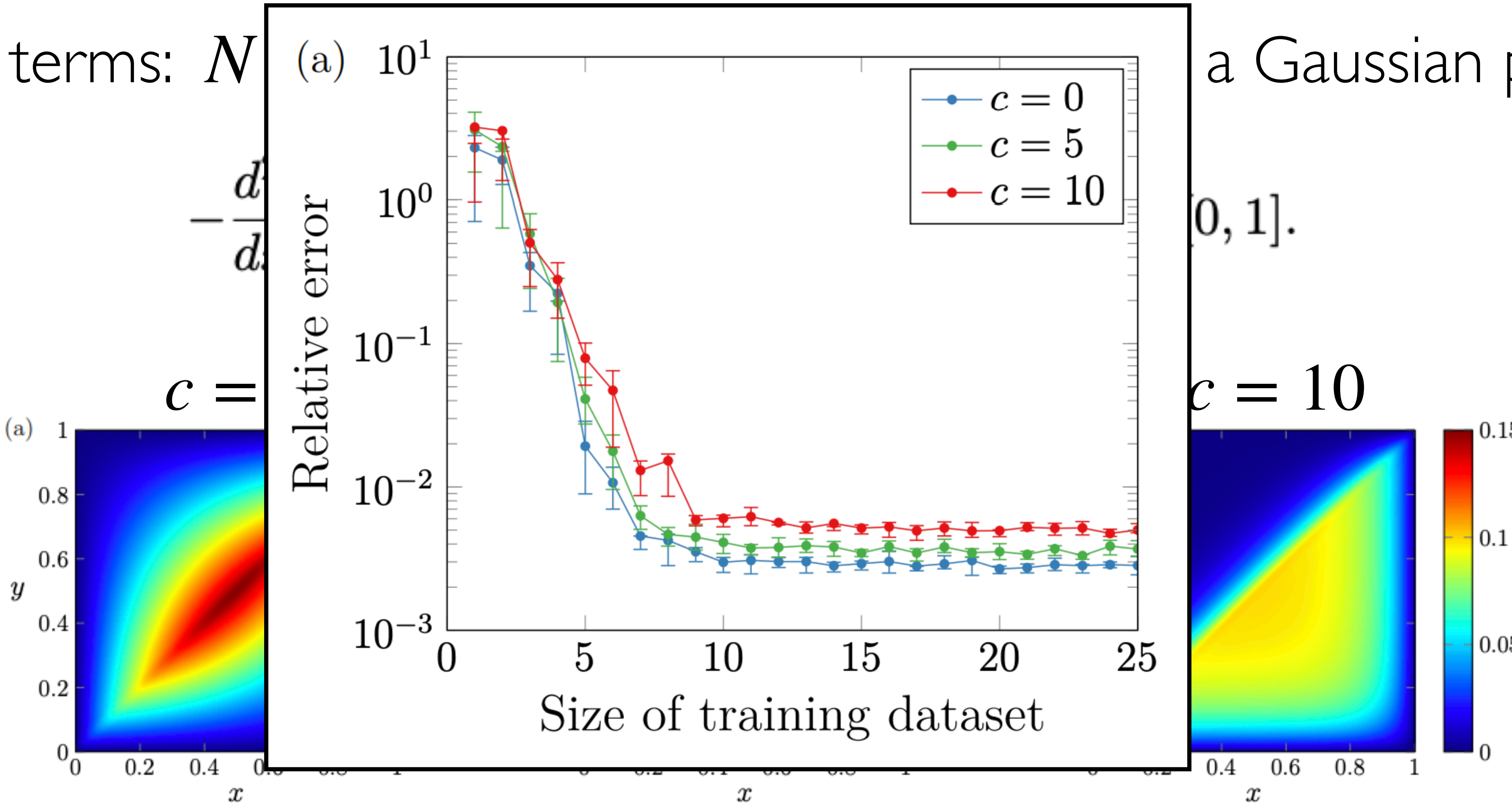
The adjoint mystery

[Boullé, Halikias, Otto & T., 2024], [Levitt & Martinsson, 2024]

Forcing terms: N … a Gaussian process.

$$-\frac{d}{d} \quad \ldots \quad [0,1].$$

$c =$ … $c = 10$



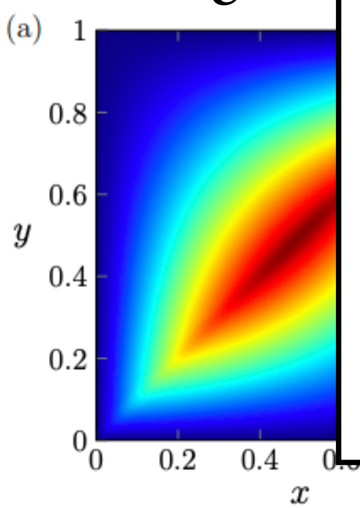
The adjoint mystery

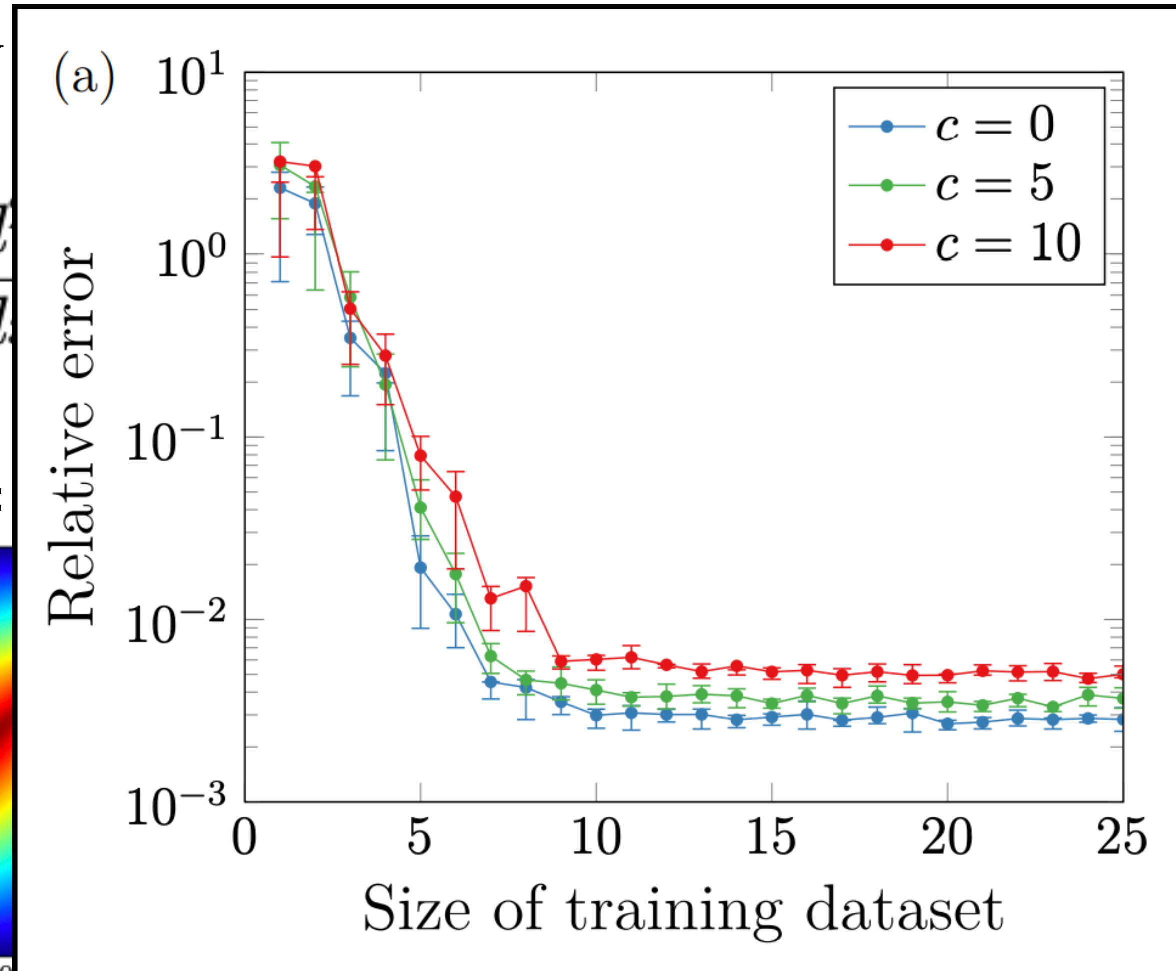
[Boullé, Halikias, Otto & T., 2024], [Levitt & Martinsson, 2024]

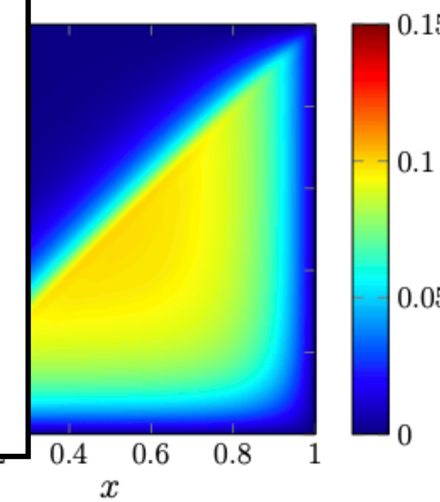
Forcing terms: N … a Gaussian process.

… $[0, 1]$.

$c =$ … $c = 10$







PDE class	With adjoint	Without adjoint

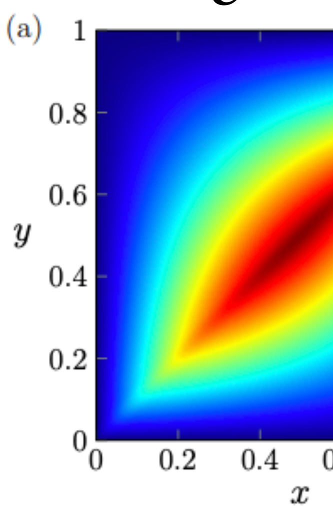
The adjoint mystery

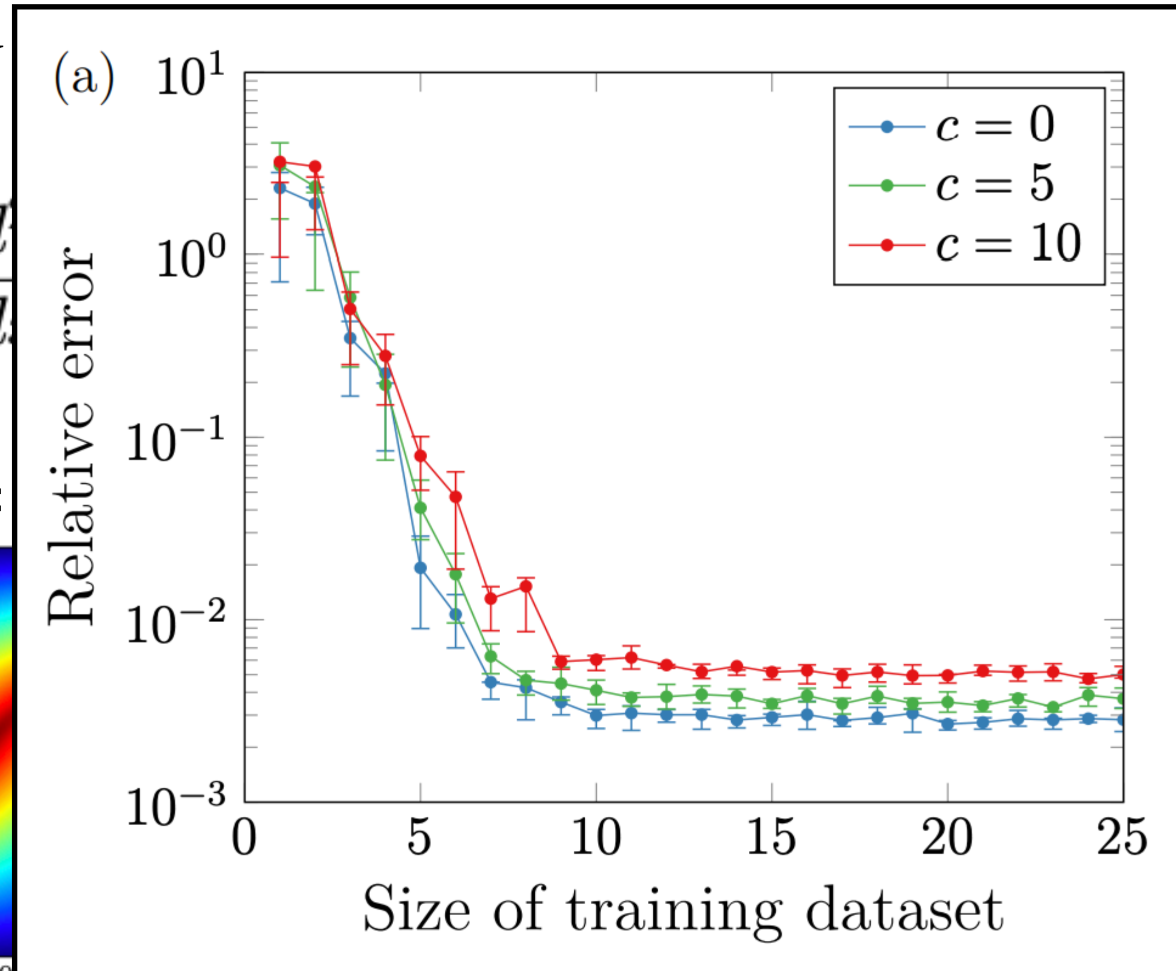
[Boullé, Halikias, Otto & T., 2024], [Levitt & Martinsson, 2024]

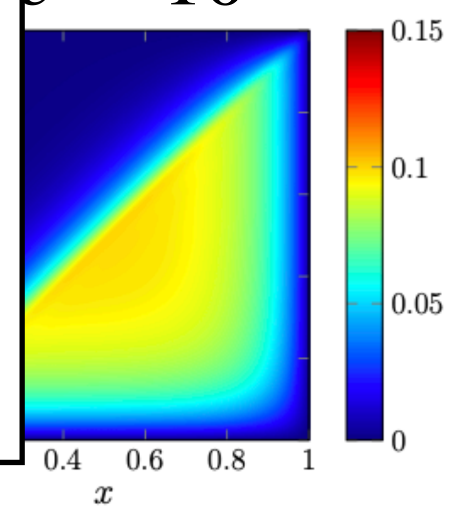
Forcing terms: N … a Gaussian process.

$$-\frac{d}{d} \ldots [0,1].$$

$c =$ … $c = 10$







PDE class	With adjoint	Without adjoint
Constant coeff., elliptic $d = 1,2,3$	$\mathcal{O}(1)$	$\mathcal{O}(1)$

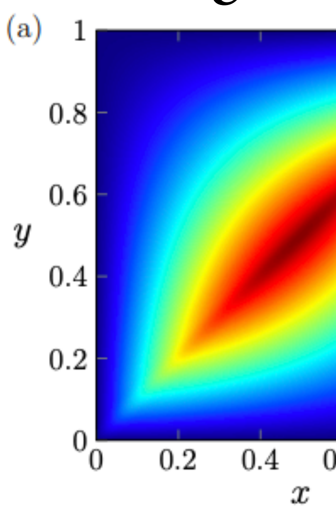
The adjoint mystery

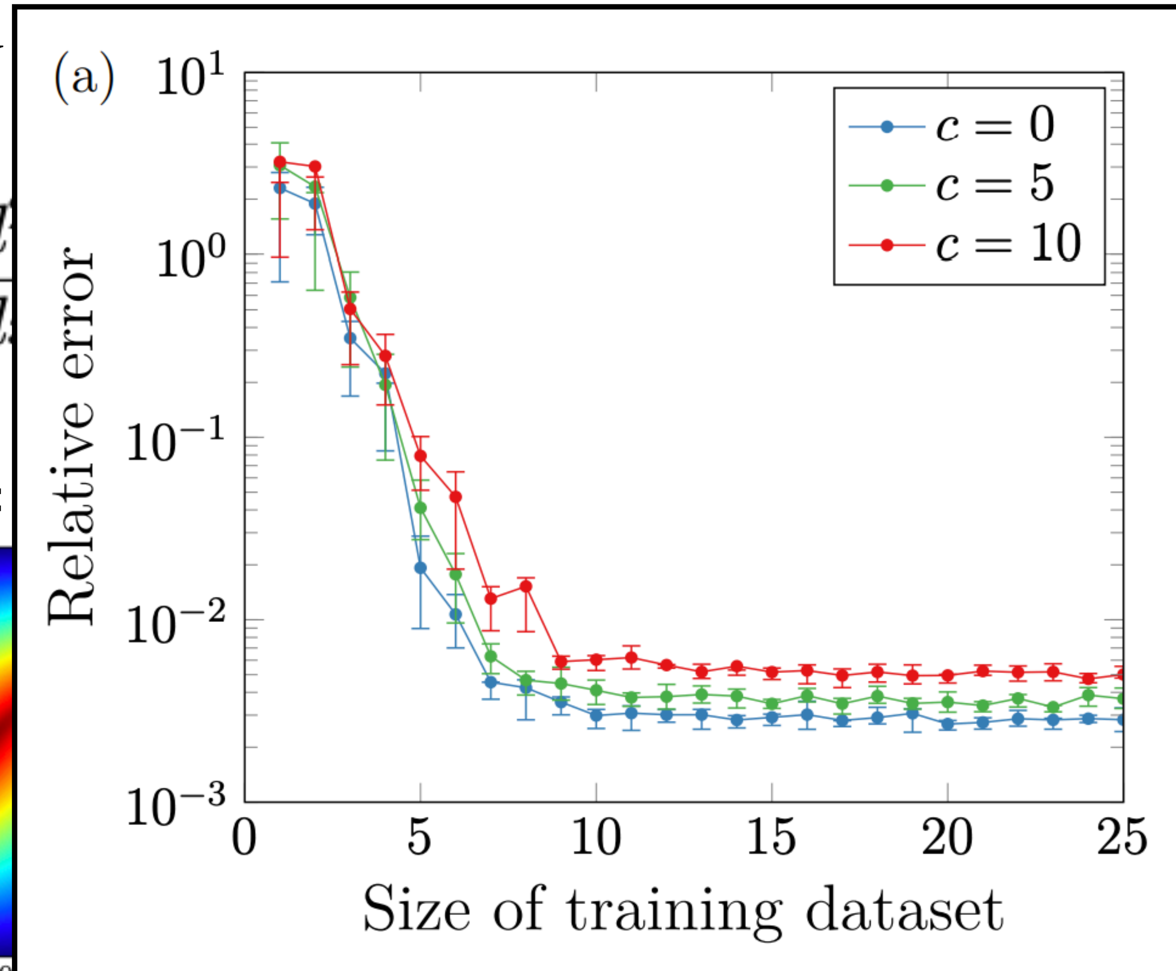
[Boullé, Halikias, Otto & T., 2024], [Levitt & Martinsson, 2024]

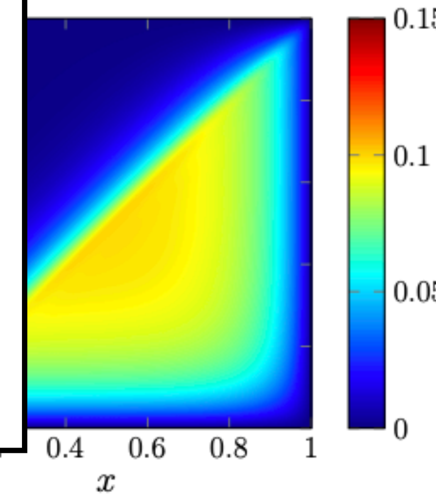
Forcing terms: N … a Gaussian process.

$$-\frac{d}{d} \ldots [0, 1].$$

$c =$ … $c = 10$





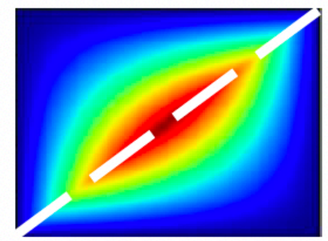


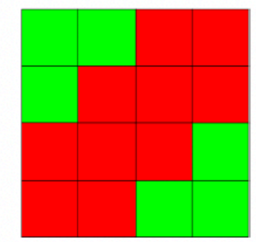
PDE class	With adjoint	Without adjoint
Constant coeff., elliptic $d = 1,2,3$	$\mathcal{O}(1)$	$\mathcal{O}(1)$
General 2nd order uniform elliptic $d = 1,2,3$	$\mathcal{O}(\log^{d+2}(1/\epsilon))$	$\mathcal{O}(\epsilon^{-d/2})$

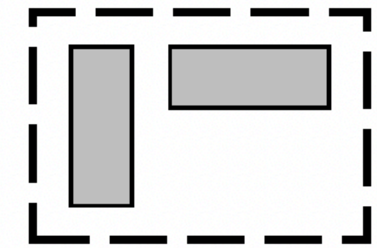
Summary

1. Theory for learning Green's functions

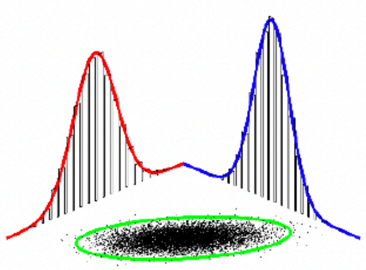
$$\mathcal{L}u = -\nabla \cdot (A(x)\nabla u)$$

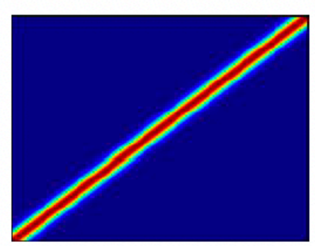


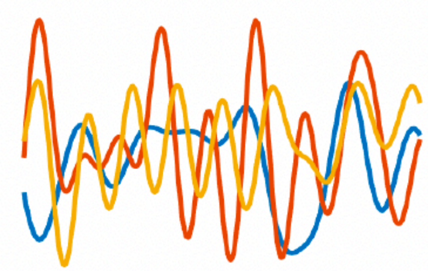


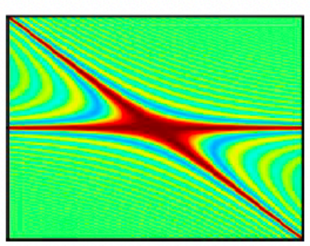


2. Generalization of the randomized SVD









<u>Question:</u>

Can operator learning be data-efficient with only input-output $\{f_i, \mathcal{G}(f_i)\}_{i=1}^{N}$ data?