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: \( G : X \rightarrow Y \)
Operator learning in a nutshell

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

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

Operator between function spaces: \( \mathcal{G} : \mathcal{X} \rightarrow \mathcal{Y} \)

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

E.g., \( \hat{\mathcal{G}}_\theta = Q \circ \sigma(\mathcal{K}_L) \circ \cdots \circ \sigma(\mathcal{K}_1) \circ P \)
Operator learning in a nutshell

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

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

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

Decoder
Operator learning in a nutshell

Operator between function spaces: \( \mathcal{G} : \mathcal{X} \rightarrow \mathcal{Y} \)

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

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

Decoder

Encoder
Operator learning in a nutshell

Operator between function spaces: \( \mathcal{G} : \mathcal{X} \rightarrow \mathcal{Y} \)

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

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

Decoder

Nonlinear activation function

Encoder
Operator learning in a nutshell

Operator between function spaces: \( \mathcal{G} : \mathcal{X} \rightarrow \mathcal{Y} \)

Approx. \( \mathcal{G} \) by building a parametric map \( \mathcal{G}_\theta \)

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

Decoding

Nonlinear activation function

Encoding

\[(\mathcal{K}_1v)(x) = \int_{D_1} \kappa_1(x, y)v(y)dy + b_1(x)\]
Operator learning in a nutshell

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

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

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

Decoder
Nonlinear activation function
Encoder

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

Want to find $\theta$ such that $\mathcal{G} \approx \hat{\mathcal{G}}_\theta$ in some sense.
Operator learning in a nutshell

Operator between function spaces: \( \mathcal{G} : \mathcal{X} \rightarrow \mathcal{Y} \)

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

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

Decoder

Nonlinear activation function

Encoder

\[(\mathcal{H}_1 v)(x) = \int_{D_1} \kappa_1(x, y)v(y)dy + b_1(x)\]

Want to find \( \theta \) such that \( \mathcal{G} \approx \hat{\mathcal{G}}_\theta \) 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]
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:
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:**

What are the \( \mathcal{G} \)'s of interest?
Neural operator learning

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

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

**Questions:**

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

How big does \( N \) need to be for a certain accuracy?
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:**

What are the \( \mathcal{G} \)'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?
**Question:**
What are the $G$'s of interest?

**My focus for this talk:**
Solution operators associated with PDEs
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 $\mathcal{N}(u_j) = f_j, \quad \mathcal{B}(u_j) = 0.$
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 $\mathcal{N}(u_j) = f_j$, $\mathcal{B}(u_j) = 0$.

Forcing functions $f_j$ $\rightarrow$ $\mathcal{G}$ $\rightarrow$ PDE solutions $u_j$
Question: What are the $G$'s of interest?

My focus for this talk: Solution operators associated with PDEs

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

Forcing functions $f_j$ $\xrightarrow{\hat{G}_\theta} \approx \hat{G}_\theta$ PDE solutions $u_j$
Data-efficient solution operator learning

2D Poisson equation
\[ \nabla^2 u = f, \quad u\big|_{[0,1]^2} = 0 \]

Accuracy of the approx. solution operator

Training pairs
Forcing term
Solution
Green's function associated with linear PDEs

**Linear PDE**

\[ \mathcal{L}u = f \text{ on } \Omega \subseteq \mathbb{R}^d \]

\[ u|_{\partial\Omega} = 0 \]

**Solution operator**

\[ u(x) = \int_{\Omega} G(x, y)f(y)dy \]

\[ =(\mathcal{G}f)(x) \]
Green’s function associated with linear PDEs

**Linear PDE**

\[ \mathcal{L}u = f \quad \text{on} \quad \Omega \subseteq \mathbb{R}^d \]

\[ u \big|_{\partial \Omega} = 0 \]

**Solution operator**

\[ u(x) = \int_{\Omega} G(x, y)f(y)dy \]

\[ = (\mathcal{G}f)(x) \]

**Poisson equation**

\[ -\nabla^2 u = f \]

\[ u(0) = u(1) = 0 \]

Green’s function for PDEs in \( d > 1 \) are unbounded functions
**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 $??$ input-output pairs $(f_j, u_j)$ such that

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

with high probability.

| PDE class | $??$ | $p$ |
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. $\hat{G}$ of $G$ for PDE class with $??$ input-output pairs $(f_j, u_j)$ such that

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

with high probability.

<table>
<thead>
<tr>
<th>PDE class</th>
<th>$??$</th>
<th>$p$</th>
</tr>
</thead>
<tbody>
<tr>
<td>uniformly self-adjoint elliptic in $d = 1,2,3$</td>
<td>$\mathcal{O}(\log^{d+2}(1/\epsilon)/\Gamma_\epsilon)$</td>
<td>2</td>
</tr>
</tbody>
</table>
**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 ?? input-output pairs $(f_j, u_j)$ such that

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

with high probability.

<table>
<thead>
<tr>
<th>PDE class</th>
<th>$p$</th>
</tr>
</thead>
<tbody>
<tr>
<td>uniformly self-adjoint elliptic in $d = 1,2,3$</td>
<td>$O(\log^{d+2}(1/\epsilon)/\Gamma_\epsilon)$</td>
</tr>
<tr>
<td>uniformly parabolic in $d \geq 1$ (and uni. self-adjoint elliptic in $d \geq 4$.)</td>
<td>$O(\log^{d+4}(1/\epsilon)/\Gamma_\epsilon)$</td>
</tr>
</tbody>
</table>
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 input-output pairs $(f_j, u_j)$ such that

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

with high probability.

<table>
<thead>
<tr>
<th>PDE class</th>
<th>$|$</th>
<th>$p$</th>
</tr>
</thead>
<tbody>
<tr>
<td>uniformly self-adjoint elliptic in $d = 1,2,3$</td>
<td>$\mathcal{O}(\log^{d+2}(1/\epsilon)/\Gamma_\epsilon)$</td>
<td>2</td>
</tr>
<tr>
<td>uniformly parabolic in $d \geq 1$ (and uni. self-adjoint elliptic in $d \geq 4$)</td>
<td>$\mathcal{O}(\log^{d+4}(1/\epsilon)/\Gamma_\epsilon)$</td>
<td>1</td>
</tr>
<tr>
<td>uniformly self-adjoint hyperbolic in $d = 1$</td>
<td>$\mathcal{O}(e^{-(6+1/r)} \log^3(1/\epsilon)/\Gamma_\epsilon)$</td>
<td>2</td>
</tr>
</tbody>
</table>
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]
We can recover a symmetric low-rank matrix with matrix-vector products $v \mapsto Av$:

**Randomized SVD:**

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

$$Y = \text{Tall-skinny Gaussian matrix with iid indep. entries}$$

[Halko, Martinsson, & Tropp, 2011], [Martinsson & Tropp, 2020]
We can recover a symmetric low-rank matrix with matrix-vector products $v \mapsto Av$:

**Randomized SVD:**

1. $n \times (k + 5)$
   
   $Y = \text{Tall-skinny Gaussian matrix with iid indep. entries}$

2. $Z = AY$
   
   Input-output data

[Halko, Martinsson, & Tropp, 2011], [Martinsson & Tropp, 2020]
We can recover a symmetric low-rank matrix with matrix-vector products $v \mapsto Av$:

**Randomized SVD:**

1. $n \times (k + 5)$
   - Tall-skinny Gaussian matrix with iid indep. entries

2. $Z = AY$
   - Input-output data

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

[Halko, Martinsson, & Tropp, 2011], [Martinsson & Tropp, 2020]
We can recover a symmetric low-rank matrix with matrix-vector products $v \mapsto Av$:

**Randomized SVD:**

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

   $$Y = \begin{bmatrix} \vdots \\ \vdots \end{bmatrix}$$

   Tall-skinny Gaussian matrix with iid indep. entries

2. $Z = AY$

   Input-output data

3. $Q = \text{orth}(Z)$

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

   $$A_k = QQ^*A$$

**Theorem** (Halko, Martinsson, Tropp, 2011).

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

$$
\mathbb{P} \left[ \| A - A_k \|_F \leq (1 + 15\sqrt{k + 5})\epsilon_k \right] \geq 0.999
$$
We can recover a symmetric low-rank matrix with matrix-vector products \( v \mapsto Av \):

**Randomized SVD:**

1. \( n \times (k+5) \) tall-skinny Gaussian matrix with iid indep. entries
2. \( Z = AY \) input-output data
3. \( Q = \text{orth}(Z) \) orthonormal basis for \( \text{col}(Z) \)
   
   \[ A_k = QQ^*A \]

Theorem (Halko, Martinsson, & Tropp, 2011).

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

\[
\mathbb{P} \left[ \| A - A_k \|_F \leq (1 + 15\sqrt{k+5})\varepsilon_k \right] \geq 0.999
\]
Generalization of the randomized SVD

Theorem [Bouillé & T., 2021]

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

$$\mathbb{P} \left[ \|A - A_k\|_F \leq (1 + 18 \sqrt{k/\gamma_k})\epsilon_k \right] \geq 0.999$$
Generalization of the randomized SVD

Prior knowledge about $A$ helps:

**Theorem** [Bouillé & T., 2021]

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

$$\mathbb{P} \left[ \|A - A_k\|_F \leq (1 + 18\sqrt{k/\gamma_k})\epsilon_k \right] \geq 0.999$$
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. $\Omega \times (k + 5)$

\[
Y = \begin{bmatrix}
\end{bmatrix}
\]

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 $f \mapsto \int_{\Omega} G(x, y)f(y)dy$:

\[ Y = \text{Cols are drawn from Gaussian process } GP(0,C) \]

Randomized SVD for HS operators:

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

$Y = \ldots$

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. \( \Omega \times (k + 5) \)

\[
Y = \begin{bmatrix}
Y_1 & \cdots & Y_k
\end{bmatrix}
\]

2. \( Z_i(x) = \int_{\Omega} G(x, y) Y_i(y) dy \)

Input-output data

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. $\Omega \times (k + 5)$
2. $Z_i(x) = \int_\Omega G(x, y)Y_i(y)dy$
3. $Q = \text{orth}(Z)$ orthonormal basis for $\text{col}(Z)$

"$G_k = QQ^*G$"

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

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. $\Omega \times (k + 5)$
2. $Y = Z_i(x) = \int_{\Omega} G(x, y)Y_i(y)dy$
3. $Q = \text{orth}(Z)$ orthonormal basis for $\text{col}(Z)$
   
   “$G_k = QQ^*G$”

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

**Theorem** [Boullé & T., 2022]

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

$$\mathbb{P} \left[ \|G - G_k\|_{L^2} \leq O \left( \sqrt{k^2/\gamma_k} \right) \epsilon_k \right] \geq 0.999$$
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. \( \Omega \times (k + 5) \)
   - \( Y = \)\[\begin{array}{c}
   \vdots \\
   \end{array}\]
   
2. \( Z_i(x) = \int_{\Omega} G(x, y)Y_i(y)dy \)
   - Input-output data

3. \( Q = \text{orth}(Z) \)
   - Orthonormal basis for \( \text{col}(Z) \)

   "\( G_k = QQ^*G \)"

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

<table>
<thead>
<tr>
<th>Theorem [Boullé &amp; T., 2022]</th>
</tr>
</thead>
<tbody>
<tr>
<td>We can construct an approximation ( G_k ) of ( G ) from ( k+5 ) random input functions ( f ) such that</td>
</tr>
<tr>
<td>( \mathbb{P} \left[ | G - G_k |_{L^2} \leq O \left( \sqrt{k^2/\gamma_k} \right) \epsilon_k \right] \geq 0.999 )</td>
</tr>
</tbody>
</table>

**Problem:**
Green’s functions typically do not have rapidly decaying singular values.

\( \epsilon_k \) decays very slowly with \( k \)
Green’s functions are low rank on separated blocks

One dimension

Very slow decaying singular values

Rapidly decaying singular values
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

Three dimensions:
- Low-rank structure on well separated domains.
  [Bebendorf, Hackbush, 2003]

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

Hierarchical structure

Level 2

Level 3

Level 4

Three dimensions

Low-rank structure on well separated domains.
[Bebendorf, Hackbusch, 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)

Hierarchical structure

(Pictures are in 1D for illustration purposes.)
Green’s functions associated with 1D hyperbolic PDEs

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

2D slice through the 4D Green’s function
Solution operators for 1D hyperbolic PDEs have Green’s functions with jumps along characteristics.

2D slice through the 4D Green’s function
Solution operators for 1D hyperbolic PDEs have Green’s functions with jumps along characteristics.

2D slice through the 4D Green’s function
Solution operators for 1D hyperbolic PDEs have Green’s functions with jumps along characteristics.

Using input-output data to:
1. Adaptively partition domain to isolate characteristics in tiny regions
2. Recover Green’s function off the characteristics
**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 $??$ input-output pairs $(f_j, u_j)$ such that

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

with high probability.

<table>
<thead>
<tr>
<th>PDE class</th>
<th>$??$</th>
<th>$p$</th>
</tr>
</thead>
<tbody>
<tr>
<td>uniformly self-adjoint elliptic in $d = 1, 2, 3$</td>
<td>$\mathcal{O}(\log^{d+2}(1/\epsilon)/\Gamma_\epsilon)$</td>
<td>2</td>
</tr>
<tr>
<td>uniformly parabolic in $d \geq 1$ (and uni. self-adjoint elliptic in $d \geq 4$)</td>
<td>$\mathcal{O}(\log^{d+4}(1/\epsilon)/\Gamma_\epsilon)$</td>
<td>1</td>
</tr>
<tr>
<td>uniformly self-adjoint hyperbolic in $d = 1$</td>
<td>$\mathcal{O}(\epsilon^{-(6+1/r)} \log^3(1/\epsilon)/\Gamma_\epsilon)$</td>
<td>2</td>
</tr>
</tbody>
</table>
Quality of training data

In our theoretical results, $\Gamma_\epsilon$ 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

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

**Definition:**

$$\gamma_k = \frac{k}{(\lambda_1 \text{Tr}(C^{-1}))}$$

$$C_{ij} = \int_{\Omega \times \Omega} v_i(x) K(x, y) v_j(y) \, dx \, dy$$

where $v_i$ is the $i$th right singular vectors of $G$.

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

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

- $0 < \gamma_k \leq 1$
- 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 \( \{f_i, G(f_i)\}_{i=1}^N \) data?
Consider

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

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

\[(\mathcal{G}f) = \int_{0}^{1} G(x, y)f(y)dy, \text{ where } G \text{ is a 1-Lipschitz smooth function} \]

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

Then, \[(\mathcal{G}f)(x) = \left( \int_{0}^{1} h(y)f(y)dy \right) g(x)\]

The adjoint is \[(\mathcal{G}^*f)(x) = \left( \int_{0}^{1} g(y)f(y)dy \right) h(x)\]
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,

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

The adjoint is

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

Training dataset size
to achieve $\epsilon$ accuracy
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 \((\mathcal{G}^*f)(x) = \left(\int_0^1 g(y)f(y)dy\right) h(x)\)

<table>
<thead>
<tr>
<th>Training dataset size to achieve (\epsilon) accuracy</th>
<th>With the adjoint</th>
<th>Without the adjoint</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Operator learning with and without the adjoint

Consider

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

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

<table>
<thead>
<tr>
<th>Training dataset size to achieve (\epsilon) accuracy</th>
<th>With the adjoint</th>
<th>Without the adjoint</th>
</tr>
</thead>
<tbody>
<tr>
<td>(O(1))</td>
<td>(O(1))</td>
<td>(O(1))</td>
</tr>
<tr>
<td>Input-output pairs</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

17
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,

\[(\mathcal{G}f)(x) = \left(\int_0^1 h(y)f(y)dy\right)g(x)\]

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

<table>
<thead>
<tr>
<th>Training dataset size to achieve (\epsilon) accuracy</th>
<th>With the adjoint (\mathcal{O}(1)) Input-output pairs</th>
<th>Without the adjoint (\mathcal{O}(1/\epsilon)) Input-output pairs</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>[Halikias &amp; T., 22]</td>
<td></td>
</tr>
</tbody>
</table>
Forcing terms: \( N \) input-output functions drawn from a Gaussian process.

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

$$\frac{d}{dt}c = \frac{d}{dt}c = 0$$

$\Rightarrow c = 10$
The adjoint mystery
[Boullé, Halikias, Otto & T., 2024], [Levitt & Martinsson, 2024]

Forcing terms: $N$

\[
\frac{dN}{dt} = c \frac{d^2}{dx^2} N + f(x)
\]

$c = 0$
$c = 5$
$c = 10$

A Gaussian process.

$[0, 1]$. $c = 10$

PDE class
With adjoint
Without adjoint
The adjoint mystery

Forcing terms: \( N \)

\[-\frac{d^4}{dx^4} + c \frac{d^2}{dx^2} \]

\( c = 0, 5, 10 \)

\[ \mathcal{O}(1) \]

\[ \mathcal{O}(1) \]

PDE class

Constant coeff., elliptic \( d = 1, 2, 3 \)

With adjoint

Without adjoint
The adjoint mystery
[Boullé, Halikias, Otto & T., 2024], [Levitt & Martinsson, 2024]

Forcing terms: \( N \)

\[-\frac{\partial c}{\partial t} + \frac{\partial}{\partial x} \left( \frac{d}{\partial x} \right) + c = 0\]

\( c = 5 \)

\( c = 10 \)

Forcing terms drawn from a Gaussian process.

\( \mathcal{O}(1) \)

\( \mathcal{O}(\log^{d+2}(1/\epsilon)) \)

\( \mathcal{O}(e^{-d/2}) \)

<table>
<thead>
<tr>
<th>PDE class</th>
<th>With adjoint</th>
<th>Without adjoint</th>
</tr>
</thead>
<tbody>
<tr>
<td>Constant coeff., elliptic</td>
<td>( d = 1, 2, 3 )</td>
<td>( \mathcal{O}(1) )</td>
</tr>
<tr>
<td>General 2nd order uniform</td>
<td></td>
<td></td>
</tr>
<tr>
<td>elliptic ( d = 1, 2, 3 )</td>
<td>( \mathcal{O}(\log^{d+2}(1/\epsilon)) )</td>
<td>( \mathcal{O}(e^{-d/2}) )</td>
</tr>
</tbody>
</table>
Summary

1. Theory for learning Green’s functions

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

2. Generalization of the randomized SVD

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