# An Optimal Automatic, Adaptive Algorithm Employing Continuous Linear Functionals

Yuhan Ding
Misericordia University

Fred Hickernell
Illinois Institute of Technology

Lluís Antoni Jiménez Rugama
Illinois Institute of Technology

yding@misericordia.edu, hickernell@iit.edu, ljimene1@hawk.iit.edu

July 3, 2018

# Outline



- Introduction
- An Automatic Algorithm
  - Assumption
  - Adaptive Algorithm
  - Computational Cost
- Examples
- Future Work

# Problem Setting



- $\mathcal{F}$ = a separable Hilbert input space with basis $\{u_i\}_{i=1}^{\infty}$

$$f = \sum_{i=1}^{\infty} \widehat{f}_i u_i \in \mathcal{F} \qquad \|f\|_{\mathcal{F}} = \left\|\left(\widehat{f}_i\right)_{i=1}^{\infty}\right\|_2$$

- $\mathcal{G}$ = a separable Hilbert output space with basis $\{v_i\}_{i=1}^{\infty}$

$$g = \sum_{i=1}^{\infty} \widehat{g}_i v_i \in \mathcal{G} \qquad \|g\|_{\mathcal{G}} = \left\|\left(\widehat{g}_i\right)_{i=1}^{\infty}\right\|_2$$

- Linear solution operator $S : \mathcal{F} \to \mathcal{G}$, satisfies

$$S(u_i) = \lambda_i v_i, \qquad \lambda_1 \ge \lambda_2 \ge \cdots \ge 0, \quad \lim_{i\to\infty} \lambda_i = 0,$$

$$\|S\|_{\mathcal{F}\to\mathcal{G}} := \sup_{f\neq 0} \frac{\|S(f)\|_{\mathcal{G}}}{\|f\|_{\mathcal{F}}} = \lambda_1.$$

# Challenge



- Set of successful algorithms:

$$\mathcal{A}(\mathcal{C}) := \Big\{\text{algorithms } A : \mathcal{C} \times (0,\infty) \to \mathcal{G} : \\ \big\| S(f) - A(f,\varepsilon) \big\|_{\mathcal{G}} \le \varepsilon \ \forall f \in \mathcal{C},\ \varepsilon > 0 \Big\},$$

  The $A$ are allowed to be adaptive and depend on linear functionals of the input function

- Can we find an adaptive algorithm $\widetilde{A} \in \mathcal{A}(\mathcal{C})$? What should $\mathcal{C}$ be? What is the computational cost of $\widetilde{A}$? Is $\widetilde{A}$ optimal?

# Computational Cost



- $\mathrm{cost}(A, f, \varepsilon)$: the number of linear functional values required to produce $A(f, \varepsilon)$

# Computational Cost



- $\text{cost}(A, f, \varepsilon)$: the number of linear functional values required to produce $A(f, \varepsilon)$
- $\text{cost}(A, \mathcal{C}, \varepsilon) := \sup\{\text{cost}(A, f, \varepsilon) : f \in \mathcal{C}\} \quad \forall \varepsilon > 0$
  Could be infinite if $\sup_{f\in\mathcal{C}} \|f\|_{\mathcal{F}} = \infty$!!!

# Computational Cost



- $\mathrm{cost}(A, f, \varepsilon)$: the number of linear functional values required to produce $A(f, \varepsilon)$
- $\mathrm{cost}(A, \mathcal{C}, \varepsilon) := \sup\{\mathrm{cost}(A, f, \varepsilon) : f \in \mathcal{C}\} \quad \forall \varepsilon > 0$
  Could be infinite if $\sup_{f\in\mathcal{C}} \|f\|_{\mathcal{F}} = \infty$!!!
- $\mathrm{cost}(A, \mathcal{C}, \varepsilon, \rho) := \sup\{\mathrm{cost}(A, f, \varepsilon) : f \in \mathcal{C} \cap \mathcal{B}_\rho\}$
  $\forall \rho > 0, \forall \varepsilon > 0, \quad$ where $\mathcal{B}_\rho := \{f \in \mathcal{F} : \|f\|_{\mathcal{F}} \le \rho\}$

# Nonadaptive Algorithm $\widehat{A} \in \mathcal{A}(\mathcal{B}_\rho)$



- Interpolation based on the first $n$ series coefficients

$$A_n(f) := \sum_{i=1}^{n} \lambda_i \widehat{f}_i v_i$$

- Error of the interpolation

$$\|S(f) - A_n(f)\|_{\mathcal{G}} = \left\| \left( \lambda_i \widehat{f}_i \right)_{i=n+1}^{\infty} \right\|_2 \leq \lambda_{n+1} \|f\|_{\mathcal{F}}$$

- Non-adaptive algorithm $\widehat{A} \in \mathcal{A}(\mathcal{B}_\rho)$: $\widehat{A}(f, \varepsilon) = A_{n^*}(f)$, where $n^* = \min\{n : \lambda_{n+1} \leq \varepsilon/\rho\}$
- The error depends on the coefficients not yet observed!!!

# Cone of Nice Functions



- Define partial sum

$$\sigma_j(f) = \left\| \left(\lambda_i \widehat{f}_i\right)_{i=n_{j-1}+1}^{n_j} \right\|_2, \qquad j = 1, 2, \ldots,$$

$\boldsymbol{n} = \{n_0, n_1, \ldots\}$ is a strictly increasing, unbounded sequence of non-negative integers

- Require the $\sigma_j(f)$ to decay steadily:

$$\begin{aligned} \mathcal{C} &= \left\{ f \in \mathcal{F} : \sigma_{j+r}(f) \le ab^r \sigma_j(f) \ \forall j, r \in \mathbb{N} \right\} \\ &= \left\{ f \in \mathcal{F} : \sigma_j(f) \le \min_{1 \le r < j} \{ ab^r \sigma_{j-r}(f) \} \ \forall j \in \mathbb{N} \right\}. \end{aligned}$$

where $0 < b < 1 < a$

- Decay rate of the $\lambda_i$ not explicitly assumed

# Data-driven Error Bound



For all $j \in \mathbb{N}$:

$$
\begin{aligned}
\left\| S(f) - A_{n_j}(f) \right\|_{\mathcal{G}} &= \left\| \left( \lambda_i \widehat{f}_i \right)_{i=n_j+1}^{\infty} \right\|_2 \\
&= \left\{ \sum_{r=1}^{\infty} \sum_{i=n_{j+r-1}+1}^{n_{j+r}} \left| \lambda_i \widehat{f}_i \right|^2 \right\}^{1/2} \\
&= \left\| \left( \sigma_{j+r}(f) \right)_{r=1}^{\infty} \right\|_2 \\
&\le \left\| \left( ab^r \sigma_j(f) \right)_{r=1}^{\infty} \right\|_2 \\
&= ab \sqrt{\frac{1}{1-b^2}} \sigma_j(f)
\end{aligned}
$$

# The Adaptive Algorithm $\widetilde{A} \in \mathcal{A}(\mathcal{C})$



## Algorithm

Given $a$, $b$, $\boldsymbol{n}$, the cone $\mathcal{C}$, and the input function $f \in \mathcal{C}$ and the absolute error tolerance $\varepsilon$, set $j = 1$.

**Step 1.** Compute $\sigma_j(f)$

**Step 2.** Check whether $j$ is large enough to satisfy

$$\sigma_j(f) \leq \frac{\varepsilon\sqrt{1-b^2}}{ab}$$

If true, then return $\widetilde{A}(f, \varepsilon) = A_{n_j}(f)$, and terminate the algorithm

**Step 3.** Increase $j$ by 1 and return to Step 1

# Computational Cost of $\widetilde{A}$



## Theorem

$\mathrm{cost}(\widetilde{A}, f, \varepsilon) = n_{j^*}$, *where* $j^*$ *is defined implicitly by*

$$\sigma_{j^*}(f) \le \frac{\varepsilon\sqrt{1-b^2}}{ab} < \sigma_j(f), \quad 1 \le j < j^*.$$

*Moreover,* $\mathrm{cost}(\widetilde{A}, \rho, \varepsilon) \le n_{j^\dagger}$, *where*

$$j^\dagger \le \min\Bigg\{ j \in \mathbb{N} : \frac{\rho^2}{\varepsilon^2} \le \frac{(1-b^2)}{a^2 b^2} \left[ \sum_{k=1}^{j-1} \frac{b^{2(k-j)}}{a^2 \lambda_{n_{k-1}+1}^2} + \frac{1}{\lambda_{n_{j-1}+1}^2} \right] \Bigg\}.$$



MISERICORDIA UNIVERSITY

# Simpler, Looser Upper Bounds on Computational Cost



## Corollary

$\text{cost}(\widetilde{A}, \rho, \varepsilon) \le n_{j^\dagger}$, *where* $j^\dagger$ *satisfies*

$$j^\dagger \le \left\lceil \log\left(\frac{\rho a^2 \lambda_{n_0+1}}{\varepsilon\sqrt{1-b^2}}\right) \div \log\left(\frac{1}{b}\right) \right\rceil .$$

*Moreover, if the* $\lambda_{n_{j-1}+1}$ *decay as*

$$\lambda_{n_{j-1}+1} \le \alpha\beta^j, \quad j \in \mathbb{N}, \qquad \text{for some } \alpha > 0,\ 0 < \beta < 1,$$

*then* $j^\dagger$ *also satisfies*

$$j^\dagger \le \left\lceil \log\left(\frac{\rho a \alpha b}{\varepsilon\sqrt{1-b^2}}\right) \div \log\left(\frac{1}{\beta}\right) \right\rceil .$$

# Lower Bound on the Complexity



A work in progress

- Know

$$\begin{aligned}\operatorname{comp}(\mathcal{A}(\mathcal{B}_\rho),\varepsilon) &:= \inf_{A\in\mathcal{A}(\mathcal{B}_\rho)} \operatorname{cost}(A,\mathcal{B}_\rho,\varepsilon)\\ &= \min\{n : \lambda_{n+1} \le \varepsilon/\rho\}\\ &= \operatorname{cost}(\widehat{A},\varepsilon)\\ &\ge \operatorname{cost}(\widetilde{A},\alpha\varepsilon,\rho)\end{aligned}$$

- But $\mathcal{C} \cap \mathcal{B}_\rho \subset \mathcal{B}_\rho$, so

$$\begin{aligned}\operatorname{comp}(\mathcal{A}(\mathcal{C}),\varepsilon,\rho) &:= \inf_{A\in\mathcal{A}(\mathcal{C})} \operatorname{cost}(A,\mathcal{C},\varepsilon,\rho)\\ &\le \operatorname{comp}(\mathcal{A}(\mathcal{B}_\rho),\varepsilon)\end{aligned}$$

# Example

- Approximating $\partial f / \partial x_1$:

$$f = \sum_{\boldsymbol{k} \in \mathbb{Z}^d} \widehat{f}(\boldsymbol{k}) u_{\boldsymbol{k}}(\boldsymbol{x}) \qquad \text{periodic}$$

$$u_{\boldsymbol{k}}(\boldsymbol{x}) := \prod_{j=1}^{d} \frac{2^{(1-\delta_{k_j,0})/2} \cos(2\pi k_j x_j + \mathbb{1}_{(-\infty,0)}(k_j)\pi/2)}{\max(1, \gamma_j k_j)}$$

$$v_{\boldsymbol{k}}(\boldsymbol{x}) := \cdots$$

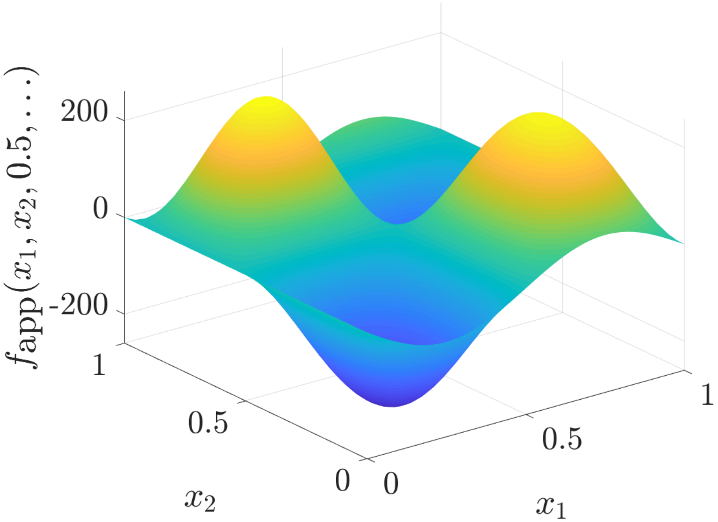
$$S(f) := \frac{\partial f}{\partial x_1} = \sum_{\boldsymbol{k} \in \mathbb{Z}^d} \widehat{f}(\boldsymbol{k}) \lambda_{\boldsymbol{k}} v_{\boldsymbol{k}}(\boldsymbol{x})$$

$$\lambda_{\boldsymbol{k}} := \frac{2\pi \left| k_1 \right|}{\prod_{j=1}^{d} \max(1, \gamma_j k_j)}$$

- Numerical example: $d = 3$, $\varepsilon = 0.1$, and $f$ generated randomly

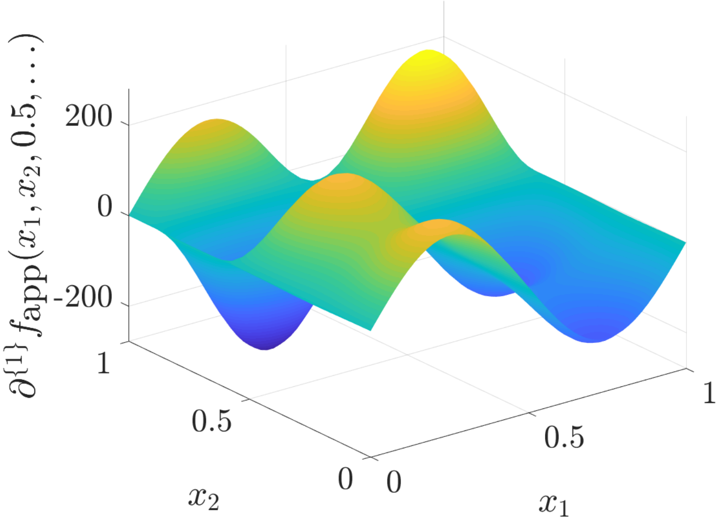
# Input Function Approximation





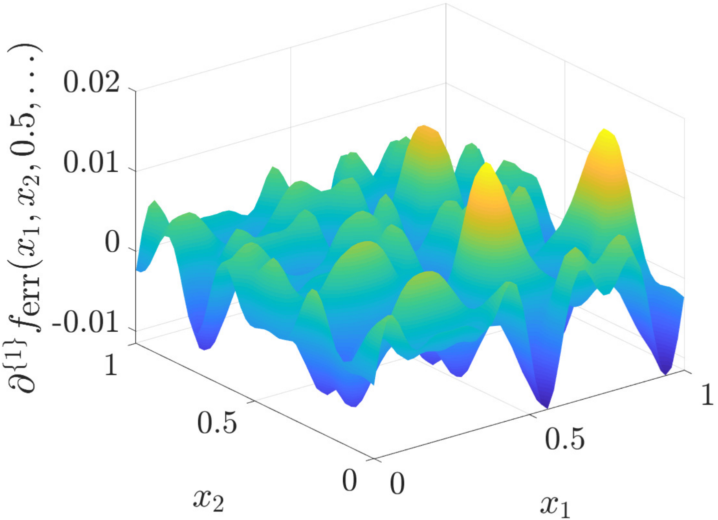
# Solution Approximation





# Approximation Error





# Summary & Future Work



## Summary

- General linear problem setting
- Define computational cost and complexity, in particular, for sets of unbounded functions

## Future Work

- Optimality of the algorithm
- Extension from Hilbert spaces to Banach spaces
- Tractability
- Algorithms that only use function values

# Thank You!

yding@misericordia.edu, hickernell@iit.edu,
ljimene1@hawk.iit.edu

MISERICORDIA
UNIVERSITY

MISERICORDIA
UNIVERSITY