AI in the Sciences and Engineering 2024: Lecture 11

Siddhartha Mishra

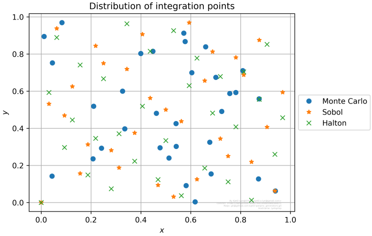
Seminar for Applied Mathematics (SAM), D-MATH (and),
ETH AI Center,
ETH Zürich, Switzerland.

What you learnt so far

- Intro into Operator learning.
- Abstract PDE: $\mathcal{D}_a(u) = f$
- Solution Operator: $\mathcal{G} : \mathcal{X} \mapsto \mathcal{Y}$ with $\mathcal{G}(a, f) = u$
- Simplified Setting: $\dim(\mathrm{Supp}(\mu)) = d_y < \infty$
- Corresponds to Parametrized PDEs with finite parameters.
- Find Soln $u(t, x, y)$ or Observable $\mathcal{L}(y)$ for $y \in Y \subset \mathbb{R}^{d_y}$.
- 1st Challenge: $\mathcal{E} \sim \sqrt{\frac{1}{N}}$
- We need lots of Data.

A Trick: Lye, SM, Ray, 2020

- Use Low discrepancy sequences $\{y_i\}_{i=1}^{N} \in Y$ as Training Set



- These sequences are Equidistributed (better spread out).
- Examples: Sobol, Halton, Owen, Niederreiter ++
- Basis of Quasi-Monte Carlo (QMC) integration.

Training on Low-Discrepancy Sequences

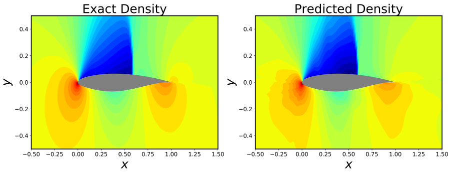
- For $\mathcal{L}$ with Bounded Hardy-Krause variation and smooth σ.
- Generalization Error for Sobol sequences: (SM, Rusch, 2020),

$$\mathcal{E} \leq \mathcal{E}_T + C\left(V_{HK}(\mathcal{L}), V_{HK}(\mathcal{L}^*)\right) \frac{(\log N)^d}{N},$$

Prediction

- Given Hicks-Henne parameter: Predict Drag, Lift, Flow
- DNN with $10^3 - 10^4$ parameters and 128 training samples :

	Run time (1 sample)	Training	Evaluation	Error
Lift	2400 s	700 s	10^{-5} s	0.78%
Drag	2400 s	840 s	10^{-5} s	1.87%
Field	2400 s	1 hr	0.2 s	1.9%

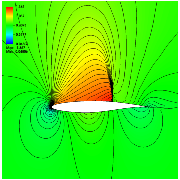


- Errors with Random Training pts: Lift 8.2%, Drag: 23.4%

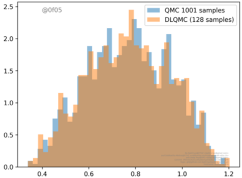
Forward UQ

- DL-UQ algorithm of Lye,SM,Ray, 2020 is

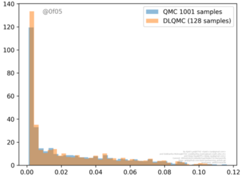
$$\mathcal{L}\#\mu \approx \frac{1}{M}\sum_{i=1}^{M}\delta_{\mathcal{L}(y_i)} \approx \frac{1}{M}\sum_{i=1}^{M}\delta_{\mathcal{L}^*(y_i)}$$



Sample



Lift PDF



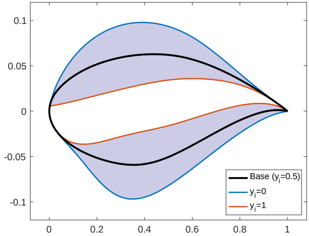
Drag PDF

Observable	Speedup (MC)	Speedup (QMC)
Lift	246.02	6.64
Drag	179.54	8.56

Application: PDE constrained Optimization

- Example: Shape optimization of airfoils
- Parametrize airfoil shape with Hicks-Henne basis functions:

$$S = S_{ref} + \sum_{i=1}^{d} \phi_i, \quad \phi_i = \phi(y_i), \; y = \{y_i\} \in Y \subset \mathbb{R}^{\bar{d}}.$$



- Change airfoil shape to Minimize Drag for constant Lift.

Airfoil shape optimization

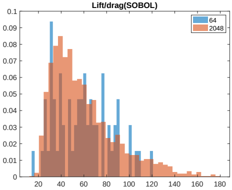
- Solve the minimization problem: Find $y^* = \arg \min_{y \in Y} J(y)$,

$$J(y) = C_D(y) + P_1(C_L(y) - C_L^{ref}) + P_2(G(y) - G^{ref}(y)).$$

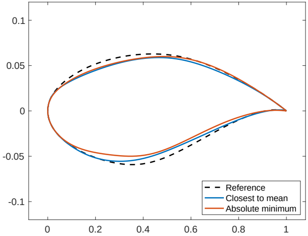
- With penalization parameters $P_1, P_2 >> 1$
- Standard Shape optimization algorithms require Gradients $\nabla_y J(y)$ at each iteration.
- Multiple calls to PDE (and Adjoint) solvers.
- Can be very expensive, even infeasible for optimization under uncertainty.

A DNN based surrogate optimization algorithm

- Choose Training Set $\mathcal{S} \subset Y$ (Random, Sobol,...)
- Train Neural Networks to obtain $C_D^* \approx C_D$, $C_L^* \approx C_L$
- For each step of optimization algorithm:
  - Evaluate objective function as $J^*(y) = J(C_D^*(y), C_L^*(y))$.
  - Evaluate Gradients $\nabla_y J^*$, Hessians $\nabla_y^2 J^*$.
- Run optimization algorithm till minimum is found.
- Significantly faster as DNNs are cheap to evaluate !
- Issue: Training points may not represent Extrema well.
- Addressed in ISMO algorithm of Lye et. al, 2020.

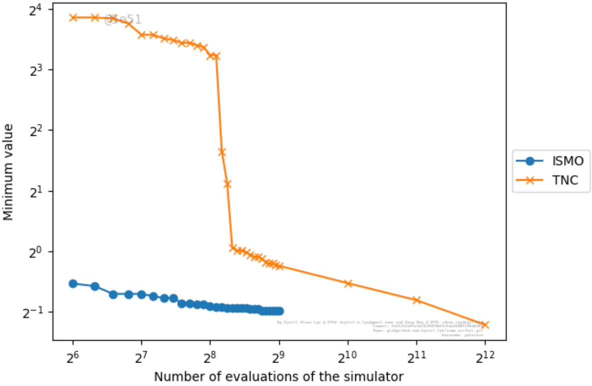


Shape Optimization of airfoils: summary

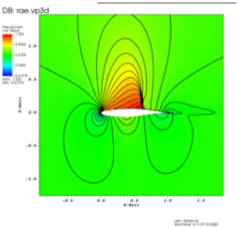


- Reference Drag: 0.0115, Mean optimized Drag: 0.0058
- Reference Lift: 0.876, Mean optimized Lift: 0.887
- Almost 50% Drag reduction on average.

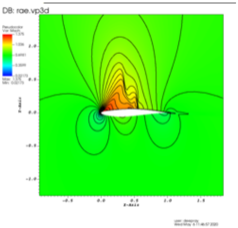
Comparison with Standard Optimizer



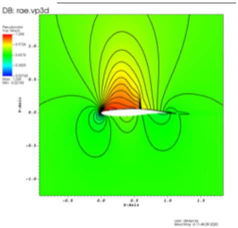
Flow around optimized shapes



Reference



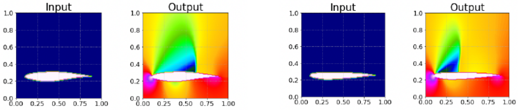
Mean



Best

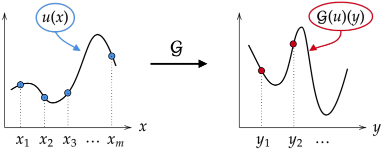
A Bigger Challenge with Parametrization

- Difficult to come up with a suitable one.
- Not unique.
- $f : x \in [0,1] \mapsto x^2$ same as
  $F : (y_1, y_2) \in [0,1]^2 \mapsto \frac{1}{2}\left[\left(\frac{y_1 - b_1}{a_1}\right)^2 + \left(\frac{y_2 - b_2}{a_2}\right)^2\right]$
- Does not Generalize well



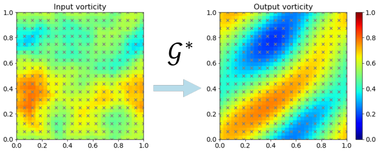
$d = 20$ $\qquad$ $d = 30$

Back to Operator Learning

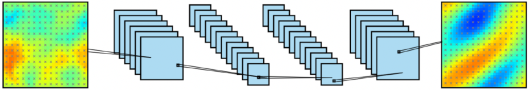


- Underlying Solution Operator: $\mathcal{G}(a) = u$ for PDE $\mathcal{D}u = a$
- Task: Find a Surrogate (based on DNNs) $\mathcal{G}^* \approx \mathcal{G}$ from data.
- Inputs+Outputs for $\mathcal{G}^*$ are Functions.

Solution I: Just use DNN + Interpolation

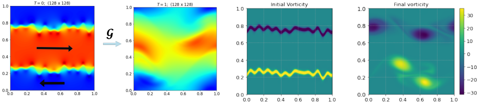


- Uniform Sampling $\mapsto$ CNN $\mapsto$ Interpolation



Does this work ?

- Shear flow with Navier-Stokes with $Re >> 1$

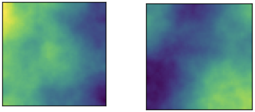


- CNN + Interpolation Results:

- Consistent with Zhu,Zabaras, 2019.
- Desiderata for Operator Learning:
  - Input + Output are functions.
  - Some notion of Continuous-Discrete Equivalence
- Learn underlying Operator, not just a discrete Representation

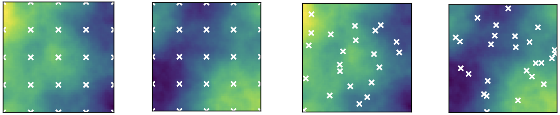
Why is this a challenge ?

- In principle, Operator maps functions to functions.



Input Output

- In practice, both inputs and outputs are Discrete



Input Output Input Output

- Multiple Discrete Representations !!
- Only discrete operations on Digital Computers.
- A proper notion of Continuous-Discrete Equivalence (CDE)

On Discrete-Continuous Equivalence

$$
\begin{tikzcd}
\mathcal{X} \arrow[r, "\mathcal{G}"] \arrow[d, bend right, "?"'] & \mathcal{Y} \arrow[d, dashed, bend right, "?"'] \\
X \arrow[r, red, "\color{red}G"] \arrow[u, dashed, bend right, "?"'] & Y \arrow[u, dashed, bend right, "?"']
\end{tikzcd}
$$

Discrete-Continuous Equivalence (Contd...)

$$\begin{tikzcd}
\mathcal{X} \arrow[r, "\mathcal{G}"] \arrow[d, bend left, "\mathcal{E}"] & \mathcal{Y} \arrow[d, bend left, "\mathcal{E}"] \\
X \arrow[r, "G"] \arrow[u, bend left, "\mathcal{R}"] & Y \arrow[u, bend left, "\mathcal{R}"]
\end{tikzcd}$$

$$\mathcal{G} = \mathcal{R} \circ G \circ \mathcal{E}$$

- Following Bartolucci et al, Neurips 2023
- Aliasing error: $\varepsilon(\mathcal{G}, G) = \mathcal{G} - \mathcal{R} \circ G \circ \mathcal{E}$
- Representation Equivalent Neural Operator alias ReNO:

$$\varepsilon(\mathcal{G}, G) \equiv 0.$$

- Concept is instantiated Layerwise: $\mathcal{G} = \mathcal{G}_L \circ \cdots \mathcal{G}_\ell \cdots \mathcal{G}_1$:

$$\mathcal{G}_\ell - \mathcal{R} \circ G_\ell \circ \mathcal{E}$$

A ReNO on different Grids

$$
\begin{tikzcd}
\mathcal{X} \arrow[r, "\mathcal{G}"] \arrow[d, bend left, "\mathcal{E}"] & \mathcal{Y} \arrow[d, bend left, "\mathcal{E}"] \\
X \arrow[r, "G"] \arrow[u, bend left, "\mathcal{R}"] & Y \arrow[u, bend left, "\mathcal{R}"]
\end{tikzcd}
\qquad
\begin{tikzcd}
\mathcal{X} \arrow[r, "\mathcal{G}"] \arrow[d, bend left, "\mathcal{E}'"] & \mathcal{Y} \arrow[d, bend left, "\mathcal{E}'"] \\
X' \arrow[r, "G'"] \arrow[u, bend left, "\mathcal{R}'"] & Y' \arrow[u, bend left, "\mathcal{R}'"]
\end{tikzcd}
$$

$$\mathcal{G} = \mathcal{R} \circ G \circ \mathcal{E} \qquad G' = \mathcal{E}' \circ \mathcal{G} \circ \mathcal{R}'$$

- A Natural change of representation (Grid) Formula:
- As $\varepsilon(\mathcal{G}, G) \equiv 0 \equiv \varepsilon(\mathcal{G}, G')$.
- Aliasing $\Rightarrow$ Discrepancies between Resolutions !!

$$
\begin{tikzcd}
X \arrow[r, "G"] & Y \arrow[d, bend left, "\mathcal{E} \circ \mathcal{R}'"] \\
X' \arrow[r, "G'"] \arrow[u, bend left, "\mathcal{E}' \circ \mathcal{R}"] & Y'
\end{tikzcd}
$$

$$G' = \mathcal{E}' \circ \mathcal{R} \circ G \circ \mathcal{E} \circ \mathcal{R}'$$

A Concrete Example: 1-D on a Regular Grid

$$
\begin{tikzcd}
\mathcal{X} \arrow[r, "\mathcal{G}"] \arrow[d, bend left, "\mathcal{E}"] & \mathcal{Y} \arrow[d, bend left, "\mathcal{E}"] \\
X \arrow[r, "G"] \arrow[u, bend left, "\mathcal{R}"] & Y \arrow[u, bend left, "\mathcal{R}"]
\end{tikzcd}
$$

$$\mathcal{G} = \mathcal{R} \circ G \circ \mathcal{E}$$

- $\mathcal{X}, \mathcal{Y}$ are Bandlimited Functions: i.e., $\mathrm{supp}\ \hat{u} \subset [-\Omega, \Omega]$
- Encoding is Pointwise evaluation: $\mathcal{E}(u) = \{u(x_j)\}_{j=1}^{n}$
- Reconstruction in terms of sinc basis:

$$\mathcal{R}(v)(x) = \sum_{j=1}^{n} v_j \mathrm{sinc}\,(x - x_j)$$

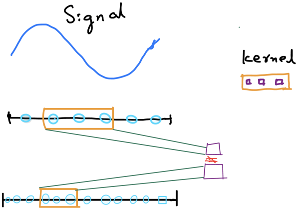
- Nyquist-Shannon $\Rightarrow$ bijection between $\mathcal{X}, X$ on sufficiently dense grid.
- Classical Aliasing Error: $\varepsilon(\mathcal{G}, G) = \mathcal{G} - \mathcal{R} \circ G \circ \mathcal{E}$

CNNs are not ReNOs !

- CNNs rely on Discrete Convolutions with fixed Kernel:

$$K_c[m] = \sum_{i=-s}^{s} k_i c[m-i]$$

- Pointwise evaluations with Sinc basis



- Easy to check that CNNs are Resolution dependent as:

$$\mathcal{G}' \neq \mathcal{E}' \circ \mathcal{R} \circ \mathcal{G} \circ \mathcal{E} \circ \mathcal{R}'$$