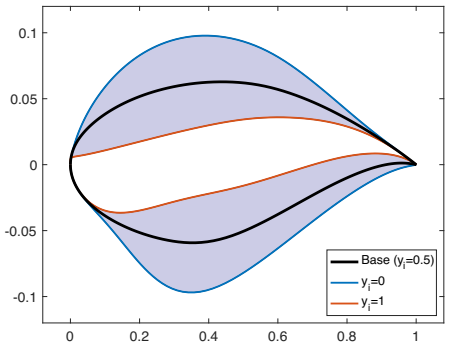
Deep Learning in Scientific Computing 2023: Lecture 8

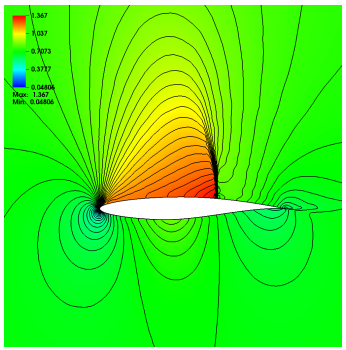
Siddhartha Mishra

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

Last Lecture: Intro to Operator Learning

- X, Y are Banach spaces and $\mu \in \mathrm{Prob}(X)$
- Abstract PDE: $\mathcal{D}_a(u) = f$
- Solution Operator: $\mathcal{G} : X \mapsto Y$ with $\mathcal{G}(a, f) = u$
- Simplified Setting: $\dim\left(\mathrm{Supp}(\mu)\right) = d_y < \infty$
- Corresponds to Parametrized PDEs with finite parameters.
- Find Soln $u(t, x, y)$ or Observable $\mathcal{L}(y)$ for $y \in Y \subset \mathbb{R}^{d_y}$.





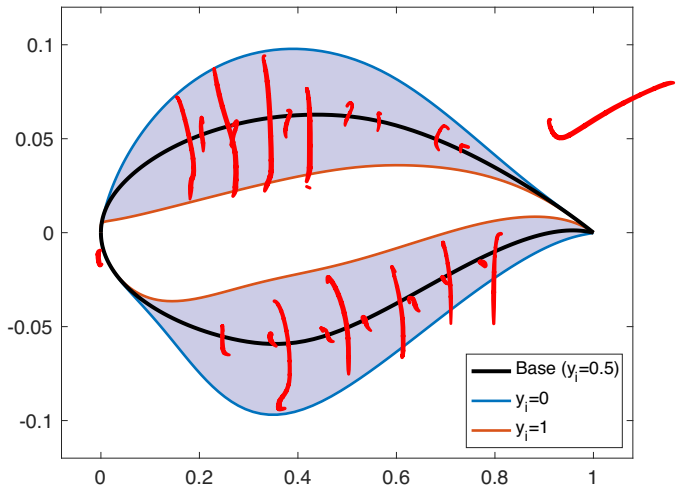
- Approximate Fields or observables with deep neural networks

Application: PDE constrained Optimization

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

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

Splines $\bar{d}$ 20-50



- Change airfoil shape to Minimize Drag for constant Lift.

Airfoil shape optimization

Compressible Euler / Navier-Stokes

$C_D(y, PDE-Sol)$

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

Drag Coeff Lift Structure

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

1000 Design constraint

- 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.

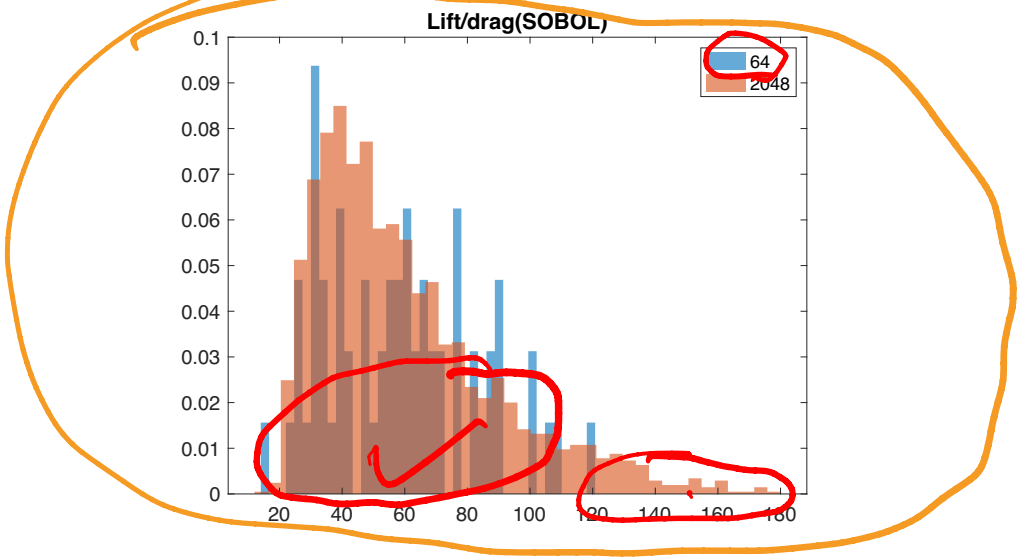
$y_{l+1} = y_l - \eta \nabla_y J(y_l)$, l - iterate

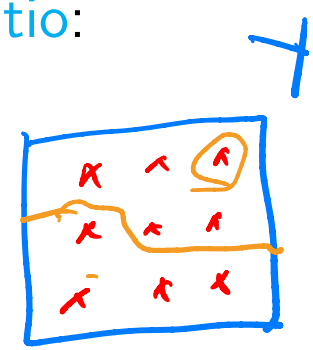
A DNN based surrogate optimization algorithm

- 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 with DNNopt

- Training points may not represent Extrema well.
- Visualized in terms of distribution of Lift to Drag Ratio:



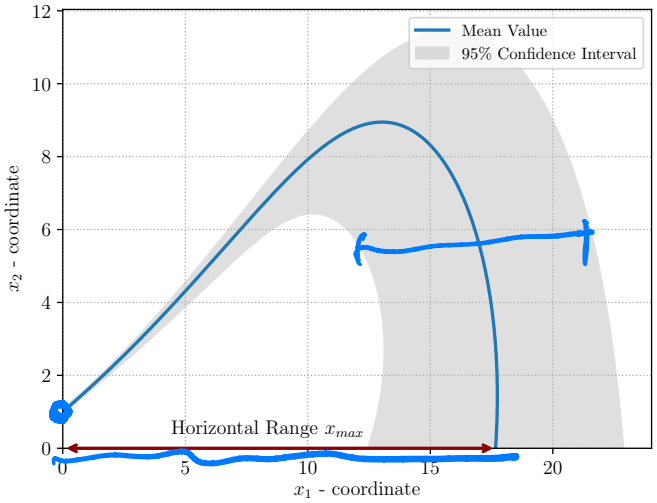


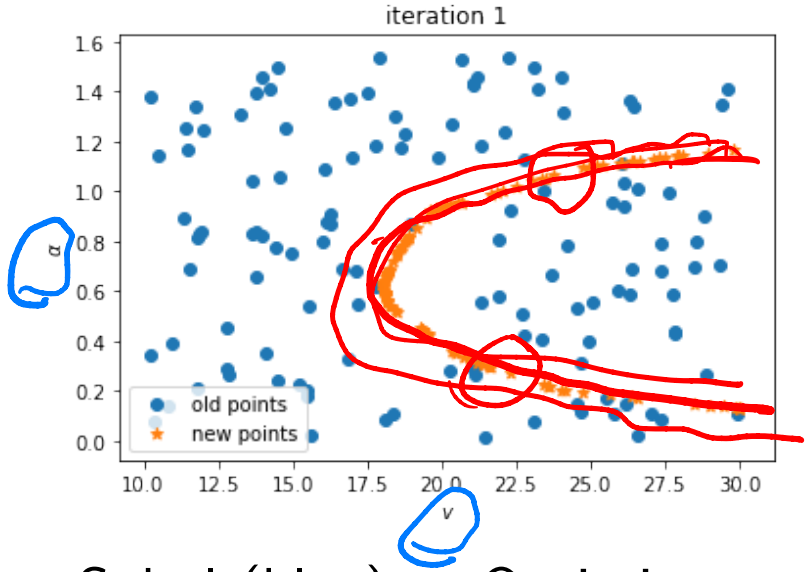
- DNNopt might not reach the desired extrema.
- DNNopt might have high variance.

Projectile Motion

Aim - Choose training pts near the Extrema

- Goal: Optimize horizontal range.





Sobol (blue) vs Optimizers

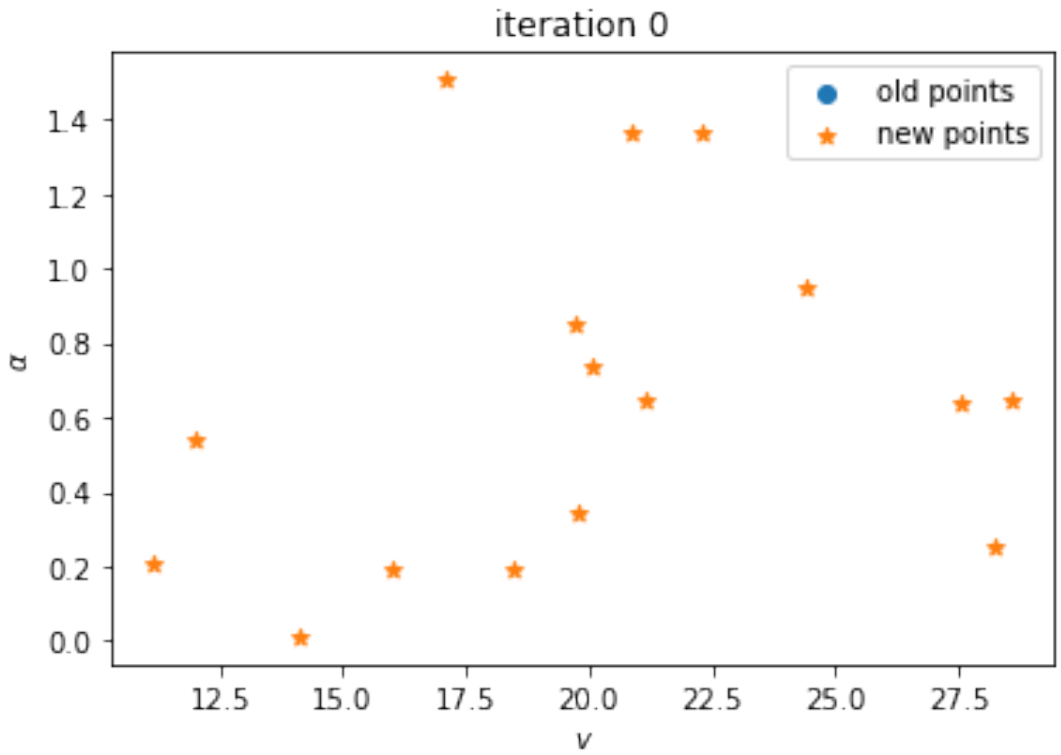
Dynamics :

$$\frac{dx}{dt} = v \qquad x_0 = v\cos\alpha, v\sin\alpha$$

$$\frac{dv}{dt} = Drag(v) - ge_2$$

ISMO: Iteration 0

Iterative Surrogate model optimization



Random ...

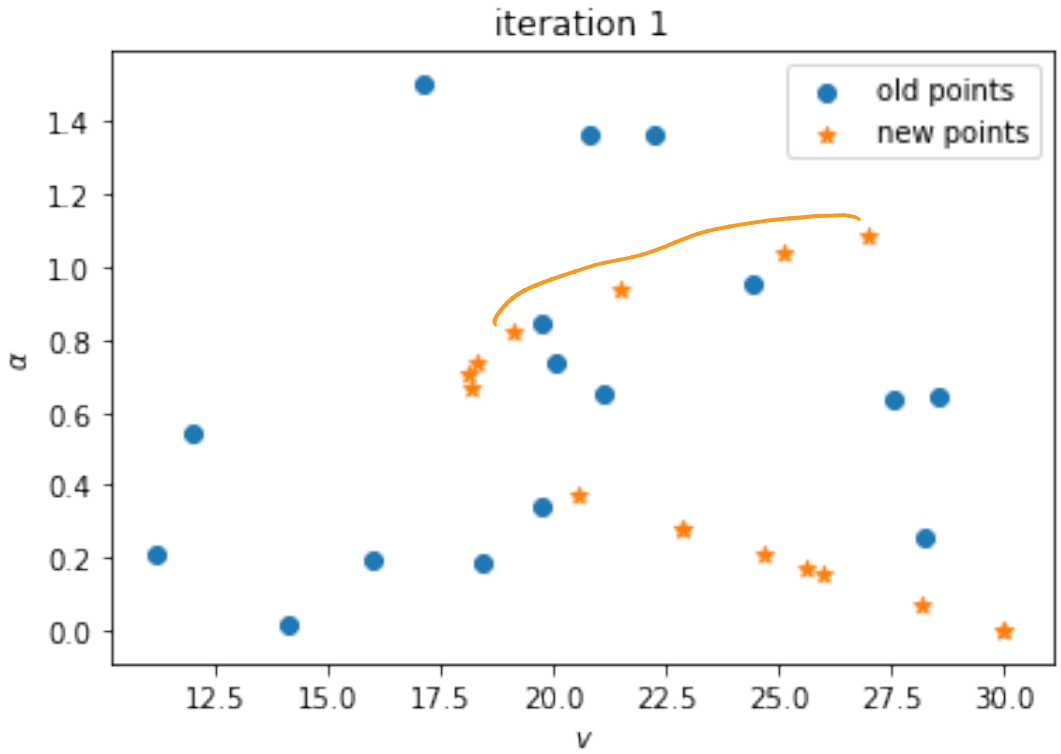
— So

ISMO: Iteration 1

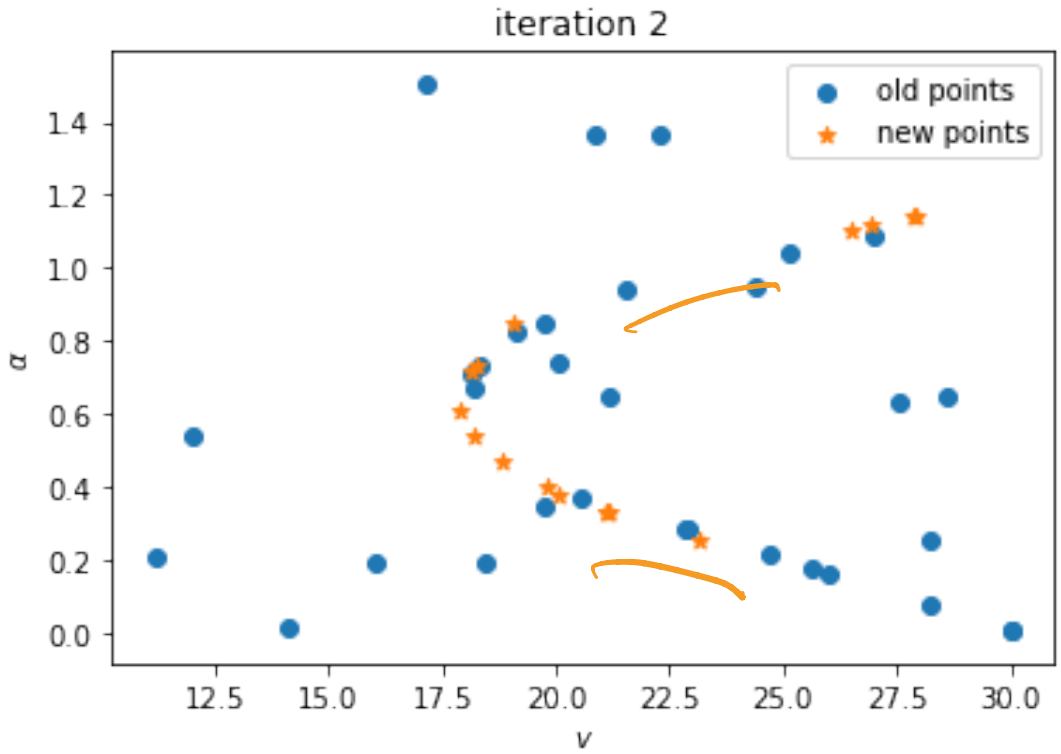
Train NN and use optimizers

Solve optimization problem (Approx)

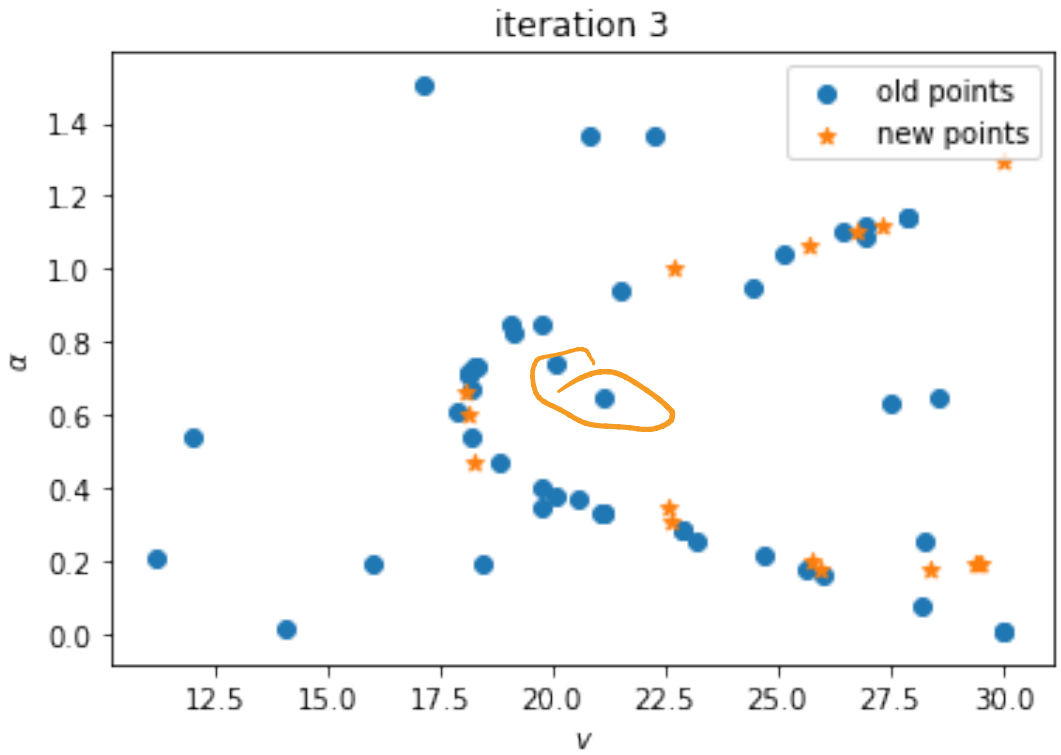
as training pts



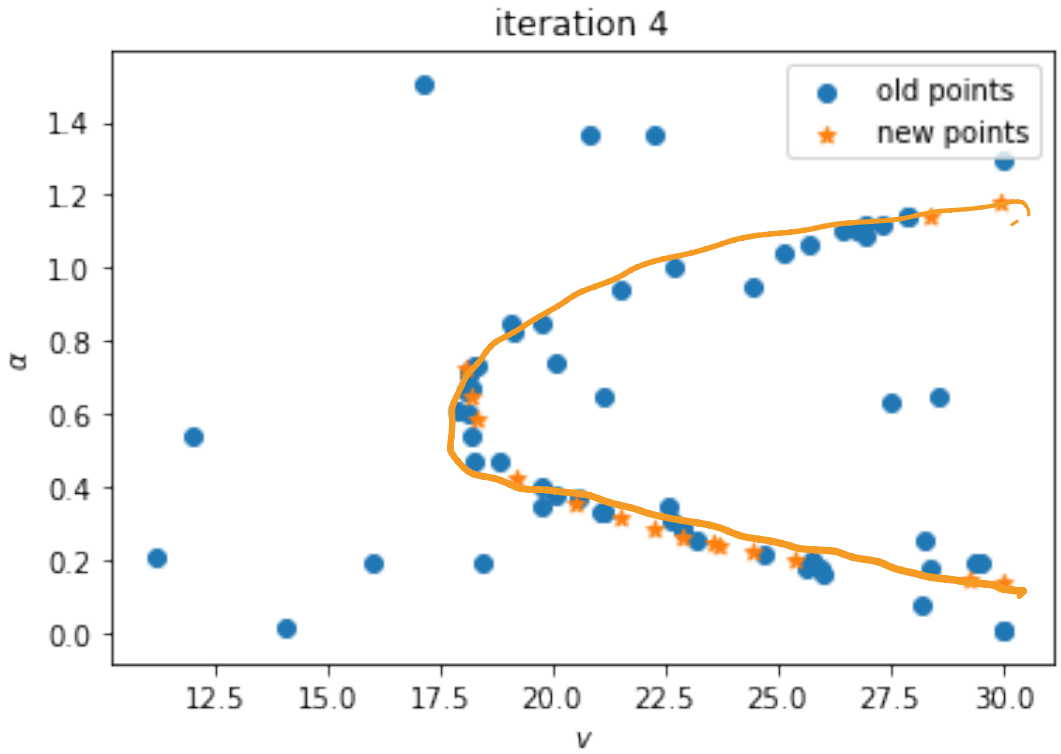
ISMO: Iteration 2



ISMO: Iteration 3



ISMO: Iteration 4



Algorithm of Lye, SM, Ray, Chandrasekhar, 2020

- Iterative Surrogate Model Optimization (ISMO):
- Active learning algorithm for minimizing $J(y)$
  - At Step 0: (Random,Sobol) training set $\mathcal{S}_0 = \{y_i^0\}_{i=1}^N$
  - Train DNN to find $J_0^* \approx J$.
  - For $\{y_j^0\}_{j=1}^{\Delta N}$ as starting value, run standard optimizer to find

$$y_j^1 = \arg\min J_0^*(y)$$

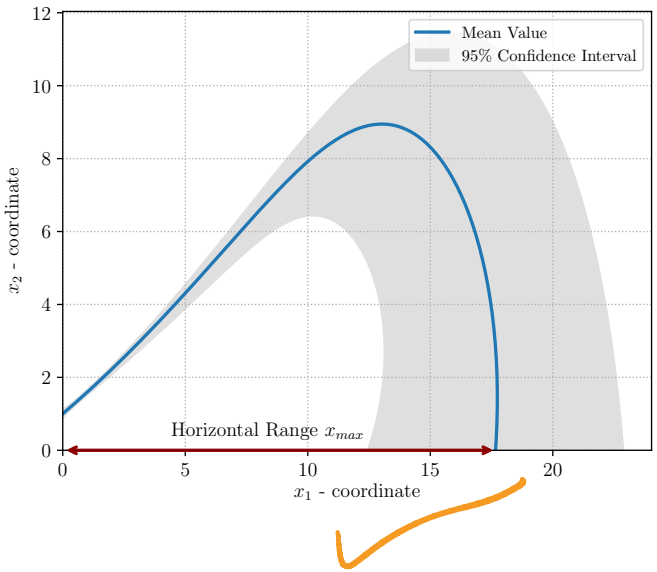
  - Set $\mathcal{S}_1 = \mathcal{S}_0 \cup \{y_j^1\}$ with $j = 1, \ldots, \Delta N$
  - At Step k: Train DNN $J_k^* \approx J$ with training set $\mathcal{S}_{k-1}$.
  - Run standard optimizer to find

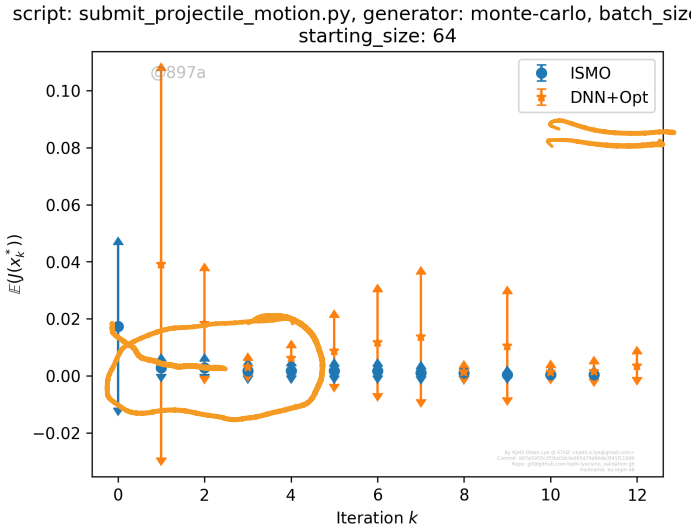
$$y_j^k = \arg\min J_{k-1}^*(y)$$

  - Set $\mathcal{S}_k = \mathcal{S}_{k-1} \cup \{y_j^k\}$ with $j = 1, \ldots, \Delta N$

Projectile Motion

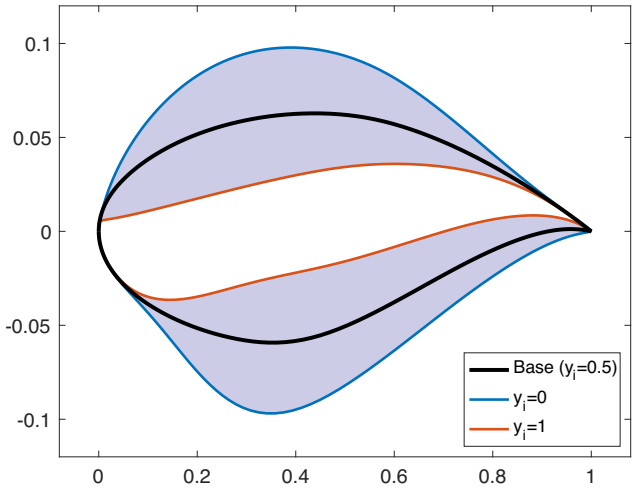
- Goal: Optimize horizontal range.





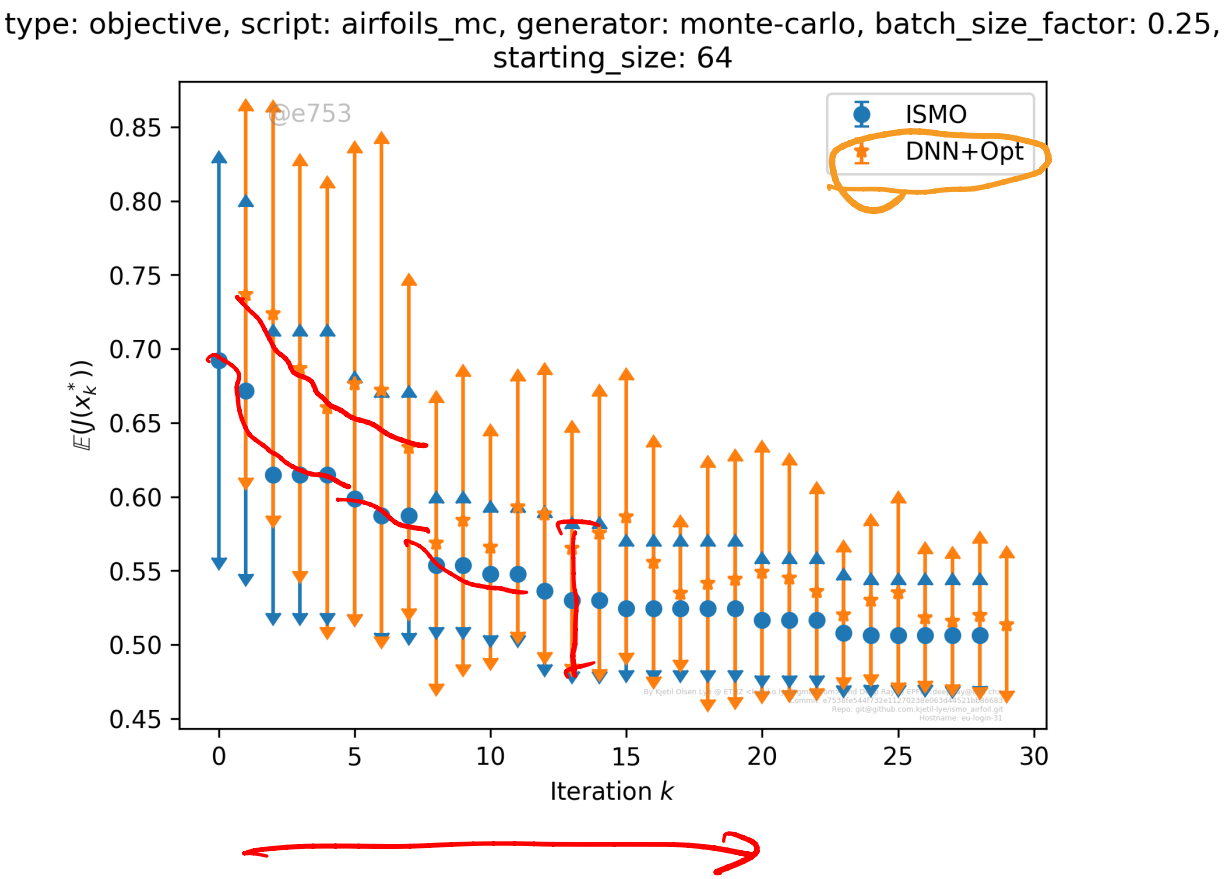
DNNopt vs ISMO

Shape Optimization of airfoils

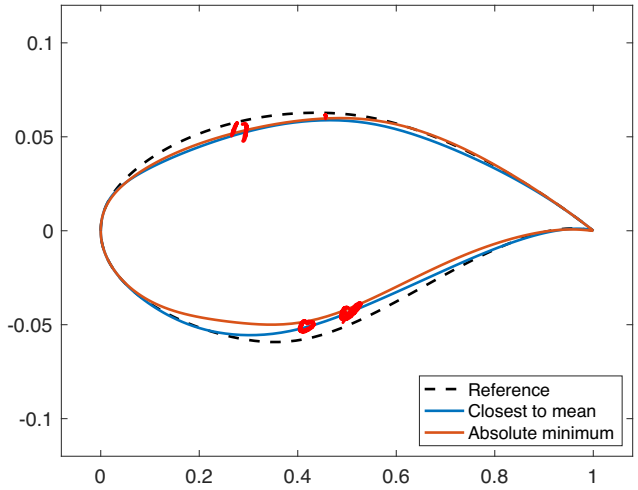


- Shapes are parameterized with Hicks-Henne functions with $\bar{d} = 20$.
- Separate DNNs for Lift and Drag.

Objective function: ISMO vs DNNopt

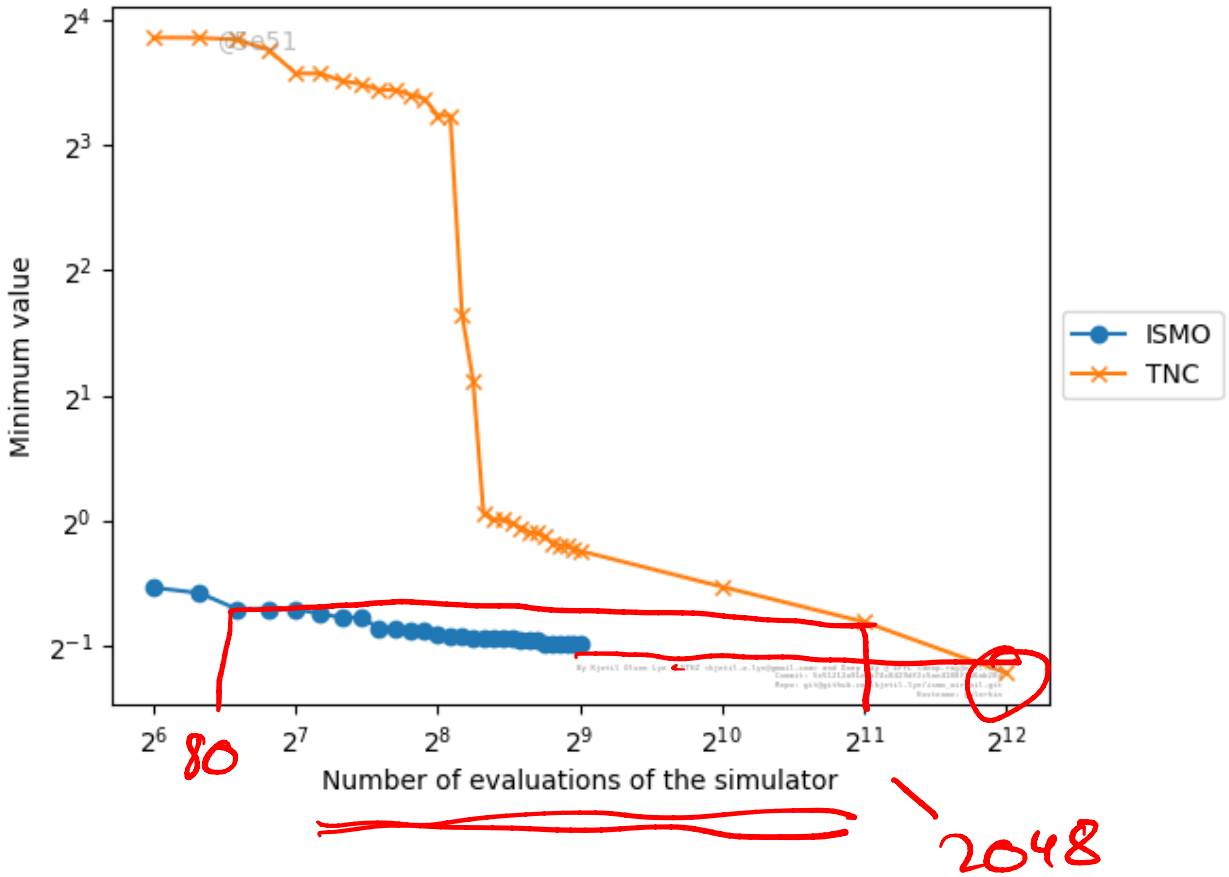


Shape Optimization of airfoils: summary

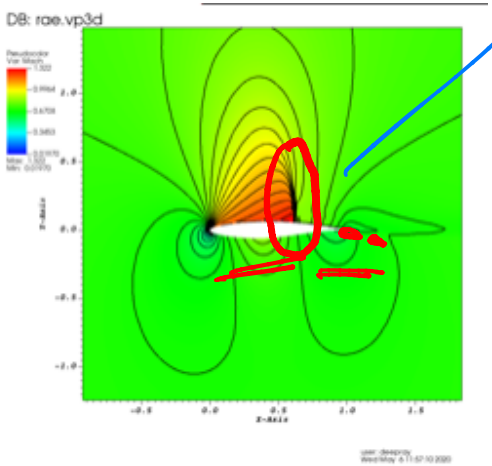


- Reference Drag: 0.0115, Mean optimized Drag: 0.0058
- Reference Lift: 0.876, Mean optimized Lift: 0.887
- Almost 50% Drag reduction on average.

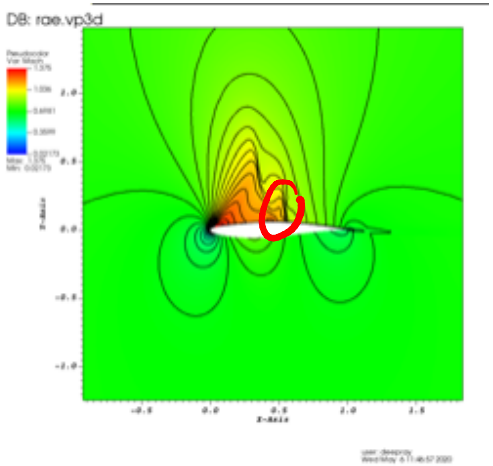
Comparison with Standard Optimizer



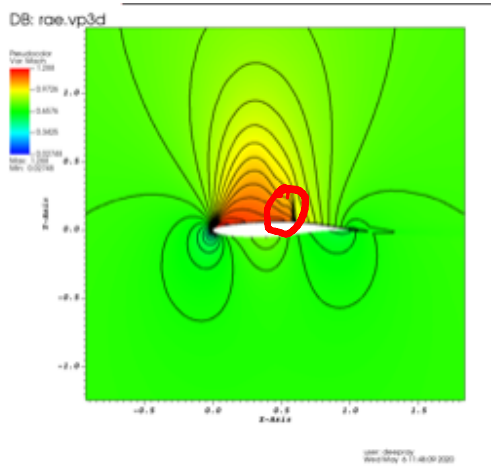
Flow around optimized shapes



Reference



Mean

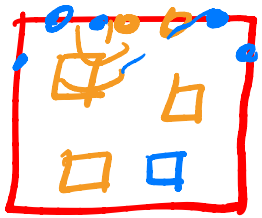


Best

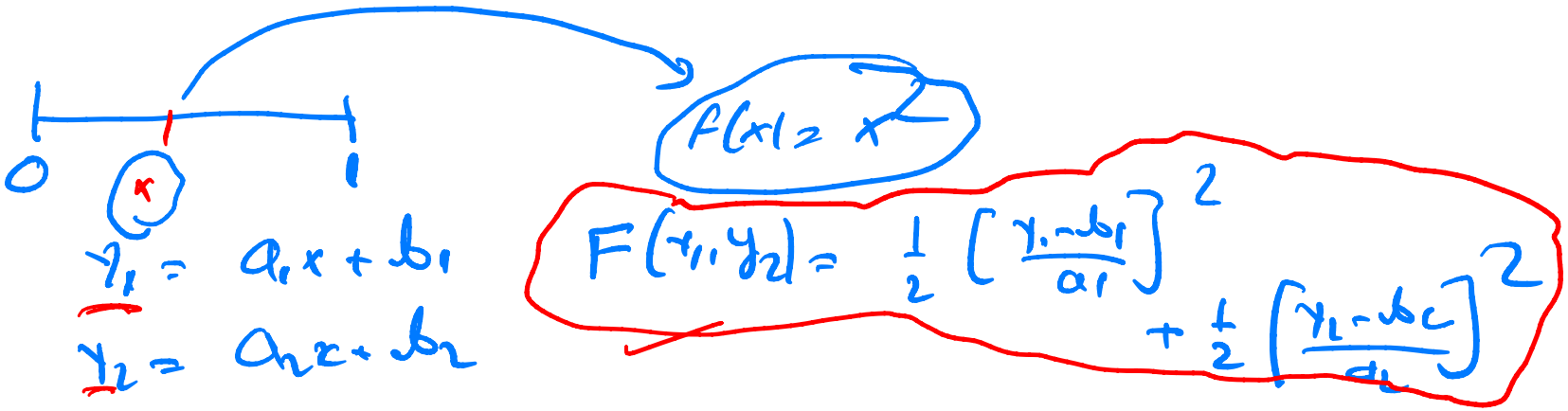
Issues with Parametrization

$G: X \longmapsto Y$ (PDE Soln. Operator) $\dim(\mathrm{Supp}(\mu)) = d_Y$ $c < \infty$

$\mu \in \mathrm{Prob}(X)$

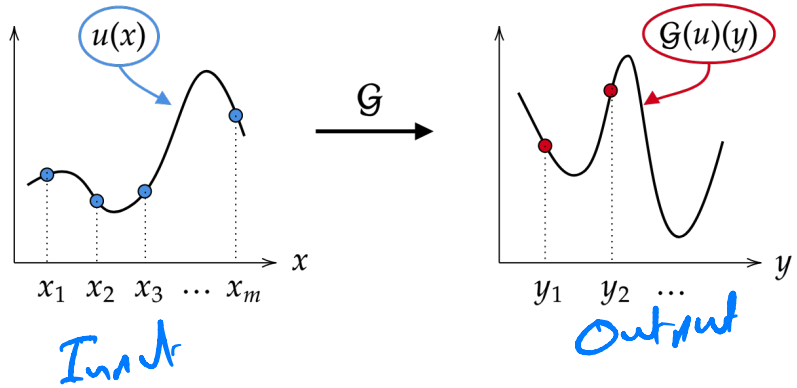


- Difficult to come up with a suitable one. → Expert Domain knowledge
- Not unique.
- Does not Generalize well



Back to Operator Learning

$G: X \longrightarrow Y$ — how to be learned.



- 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

Navier-Stokes:

$u_t + uu_x + vu_y + p_x = \nu(u_{xx} + u_{yy})$

$v_t + uv_x + vv_y + p_y = \nu(v_{xx} + v_{yy})$

$u_x + v_y = 0$

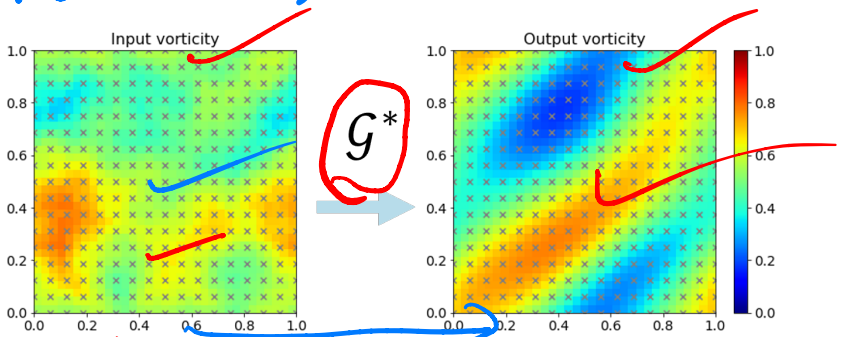
(u, v) → velocity

p → pressure.

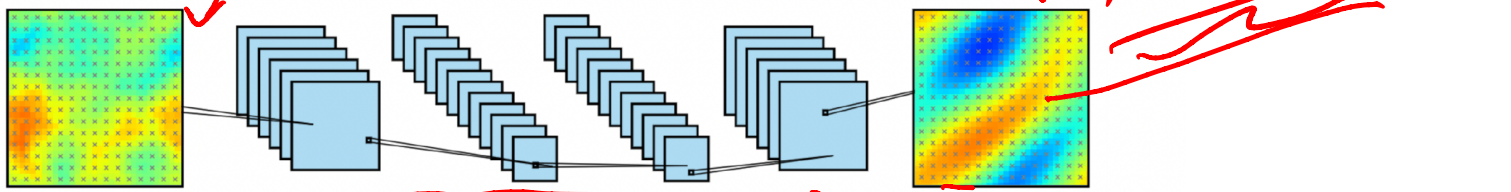
Vorticity $= \omega = \text{Curl(velocity)}$

$\omega = v_x - u_y$

$[0,T]^2$



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



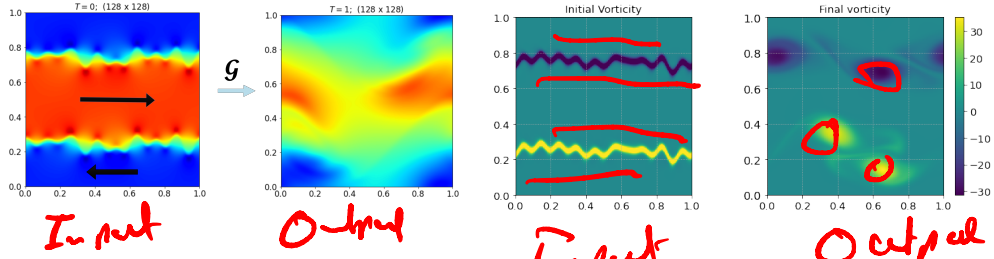
Function — 16x16 Array (vectors)

4096 (vectors)

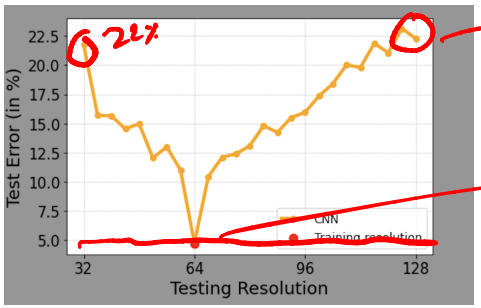
Inconsistent in function space

Does this work ?

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

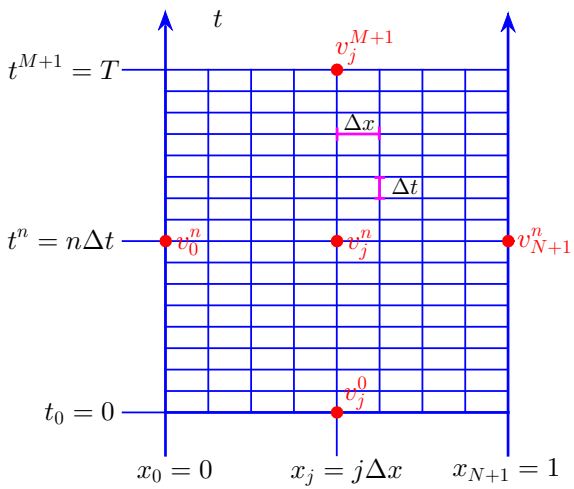


- CNN + Interpolation Results:

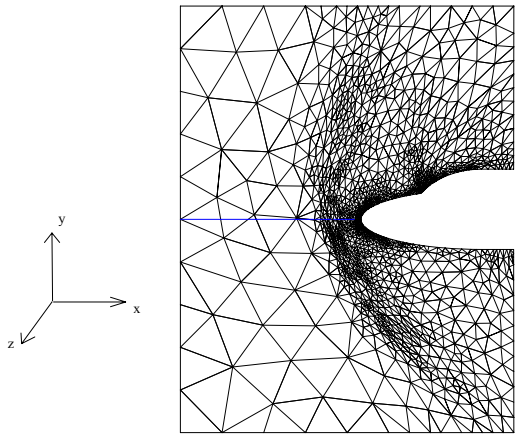


- Consistent with Zhu,Zabaras, 2019.
- Desiderata for Operator Learning:
  - Input + Output are functions.
  - Some notion of Continuous-Discrete Equivalence

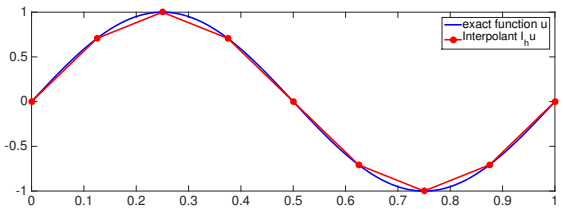
Traditional Numerical Methods



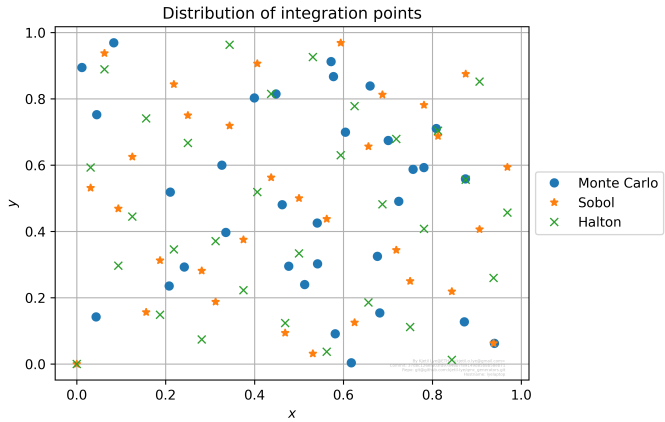
Finite Difference



Finite Volume



Finite Element

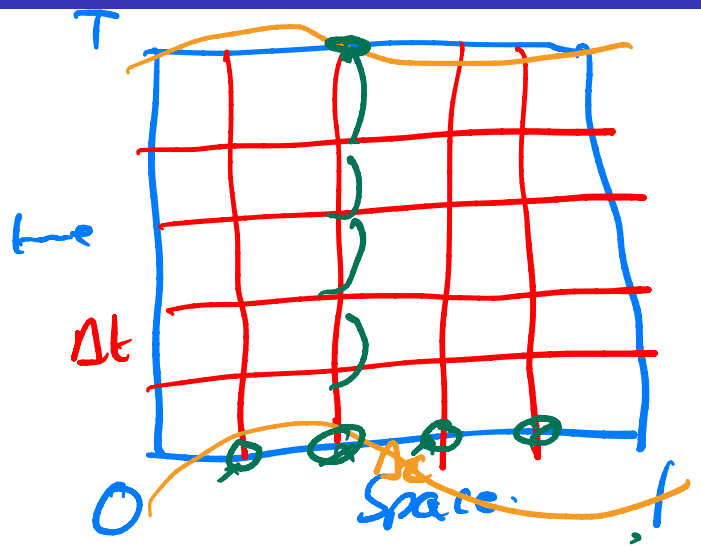


Collocation

Finite Differences for Heat Equation

$$u_t = u_{xx}$$
$$u(0,x) = \bar{u}(x)$$
$$u(0,t) = u(1,t) = 0$$

u_j^n –



Operator

$\bar{u} \mapsto u(\cdot,T)$

Fix $\{\Delta x\}$

$$\frac{u_j^{n+1} - u_j^n}{\Delta t} = \frac{u_{j+1}^n - 2u_j^n + u_{j-1}^n}{\Delta x^2}$$

$$\Rightarrow u_j^{n+1} = \left(1 - \frac{2\Delta t}{\Delta x^2}\right) u_j^n + \frac{\Delta t}{\Delta x^2}\left(u_{j-1}^n + u_{j+1}^n\right)$$

$\bar{u}$ Sampling $\bar{u}_j$ $\longmapsto$ Scheme $\longmapsto$ u_j^N $\longrightarrow$ Interpolation :— $u(\cdot,T)$

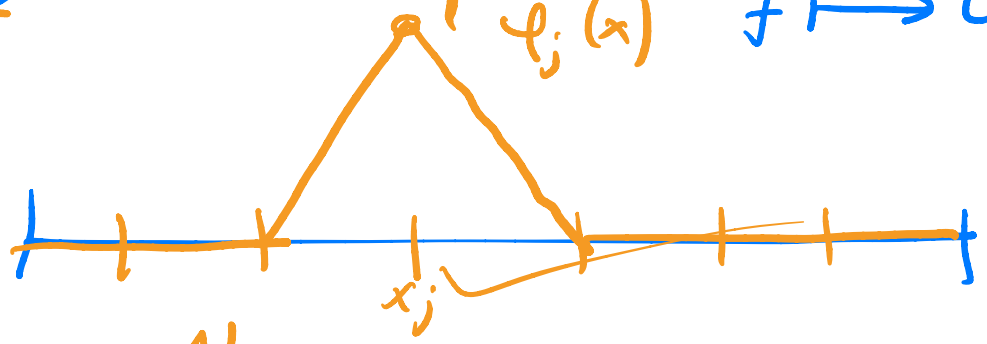
$\bar{u}_j$: vector

u_j^N: vector

Finite Element for Poisson Equation

$-u''(x) = f(x)$ on $x \in (0,1)$ Operator $f \mapsto u$

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



FEM

Choose hat-functions $\{\varphi_j\}_{j=1}^{N}$ — "basis"

Input f $\longmapsto$ $f_j = \int_0^1 f \varphi_j \, dx$ — Load coefficients.

Processing: $f_j \xrightarrow{\text{Physics}} u_j$: $u_j = A^{-1}(f_j)$

$A \to$ stiffness matrix: $A_{ij} = \int_0^1 \langle \nabla \varphi_i, \nabla \varphi_j \rangle \, dx$

Output : $u(x) = \sum_j a_j \varphi_j(x)$

Spectral Method for Poisson Equation

$$-u''(x) = f(x):$$

Basis - Trigonometric function: $\{\sin(k\pi x)\}$

$$f \longmapsto f_k = \int_0^1 f(x) \sin(k\pi x)\, dx$$

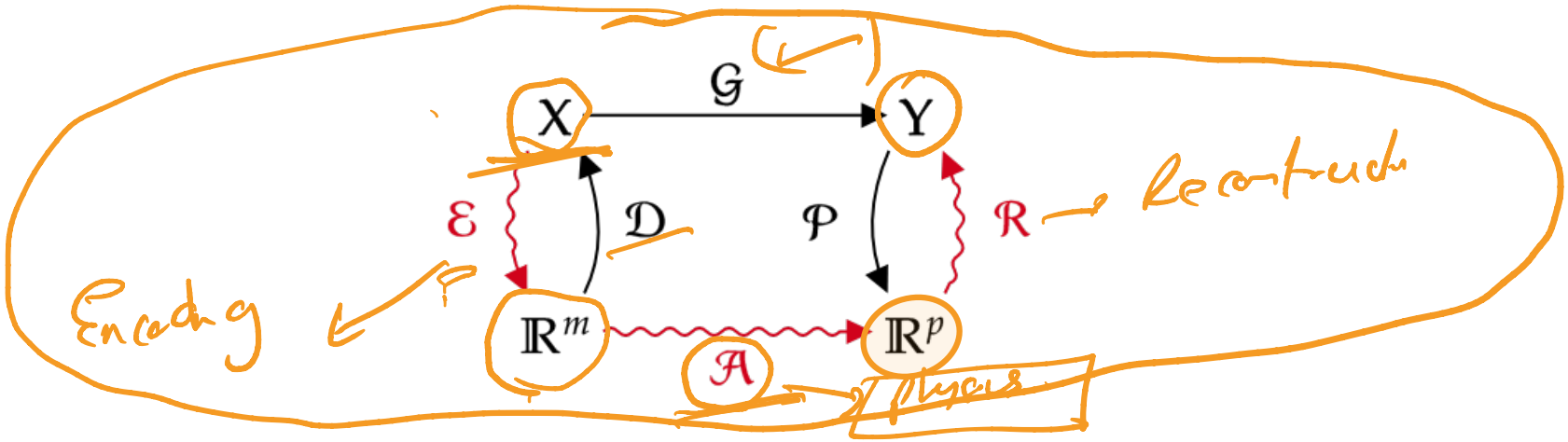
Processing: $u_k = \dfrac{f_k}{\pi^2 k^2}$

Solution: $u(x) = \sum u_k \sin(k\pi x)$

$$f \xrightarrow[\text{Projection}]{} f_j \xrightarrow[\text{Physics}]{\text{Processing}} u_j \xrightarrow{\text{Interpolation}} u$$

Revisiting Numerical Methods

- Can be reinterpreted in the following abstract Paradigm:

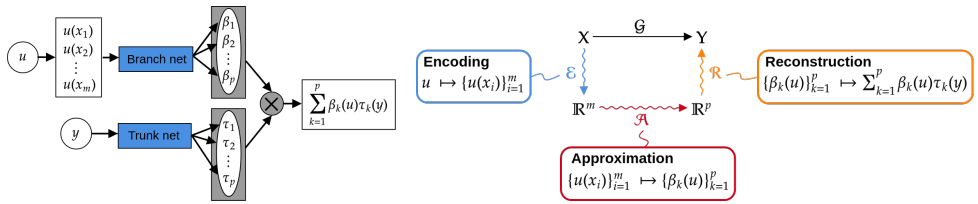


Scheme	Encoder	Approximator	Reconstructor
Finite Difference	Point values	Scheme	Poly. Interpolant
Finite Element	Node Values	Scheme	Galerkin Basis
Finite Volume	Cell Averages	Scheme	Poly. Interpolant
Spectral	Fourier Coeffs.	Scheme	Fourier Interpolant

Operator Networks

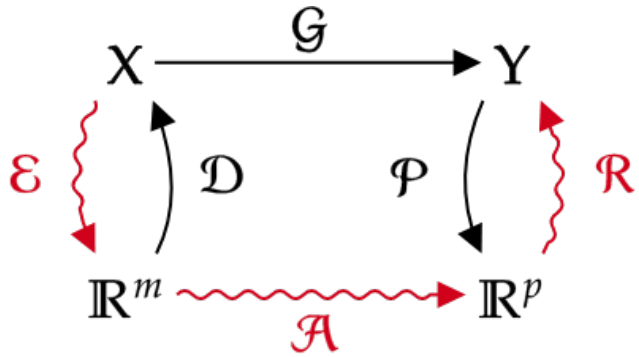
- DeepONets: Chen,Chen 1995, Lu et al, 2020:

$$\mathcal{N}(a)(y) = \tau_0(y) + \sum_{k=1} \beta_k(a)\tau_k(y) \approx \mathcal{G}(a)(y)$$



How to train DeepONets

Other OpLearn Architectures



Architecture	Encoder	Approximator	Reconstructor
DeepOnet	Sensor Evals.	DNNs	DNNs
PCA-Net[1]	Input PCA	DNNs	Output PCA
SNO[2]	Coeffs	DNNs	Chebychev basis

[1] Bhattacharya et al, 2020

[2] Fanaskov and Oseledets, 2022