

Adaptive Sampling Methods for Physics-Informed Neural Networks

Tailored for NGD Optimizers

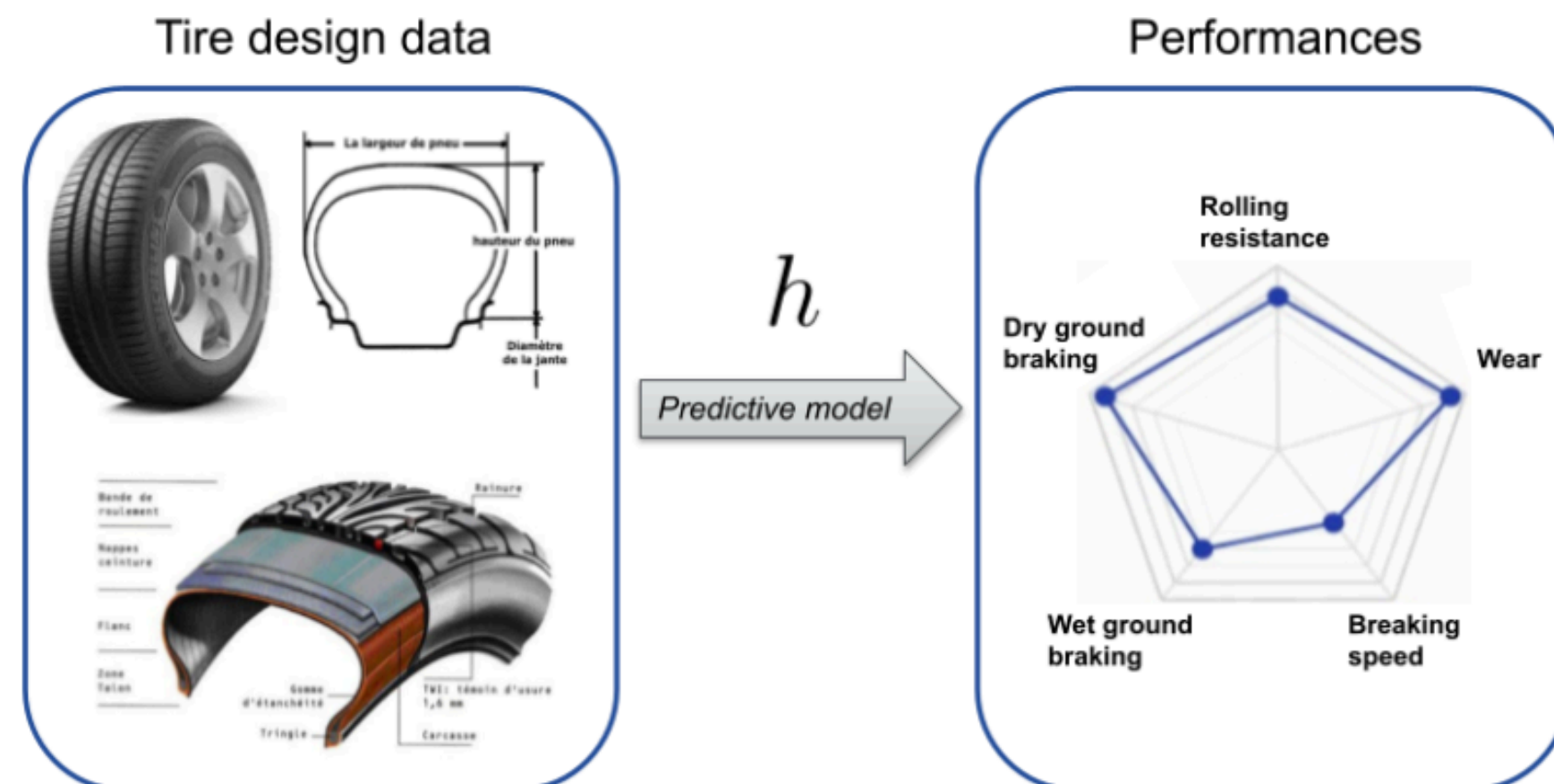
Chourouk El Hassanieh

*Postdoctoral Researcher at Centre Borelli, ENS Paris-Saclay
within the Industrial Data Analytics and Machine Learning IDAML partnership chair*

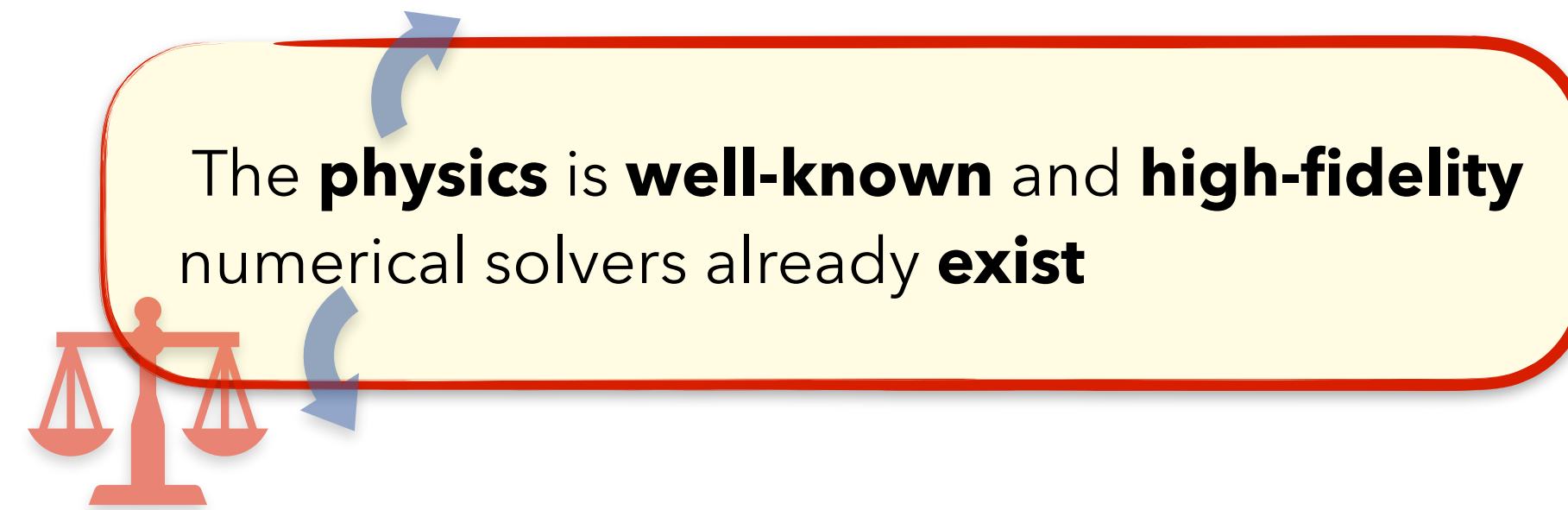
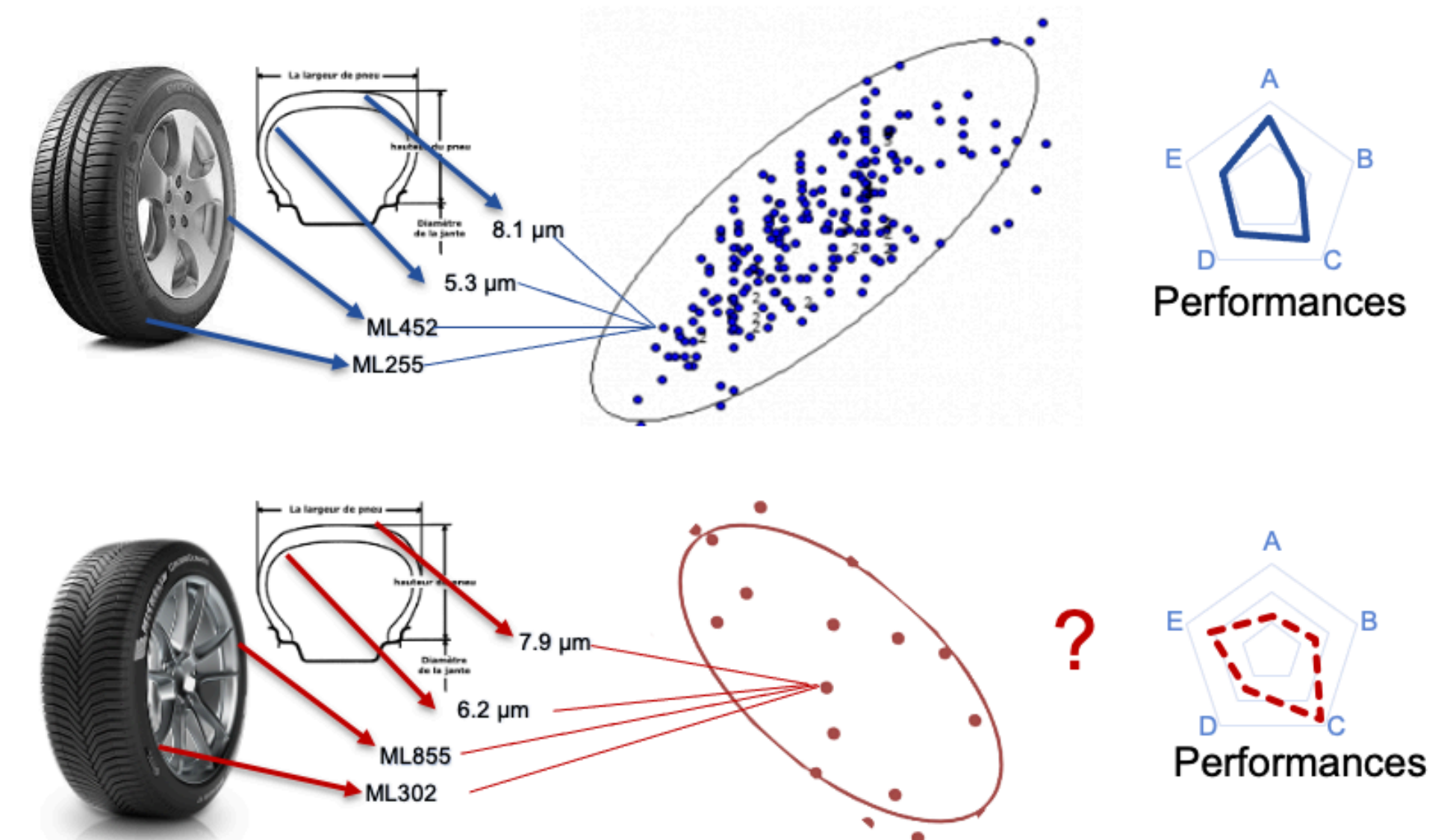
1 Industrial Motivations

Product Design

- ♦ New products are regularly manufactured with a **long** and **costly** process
- ♦ The **data sets** gathered during the development of products are relatively **small**



Credit: Antoine De Mathelin, PhD.

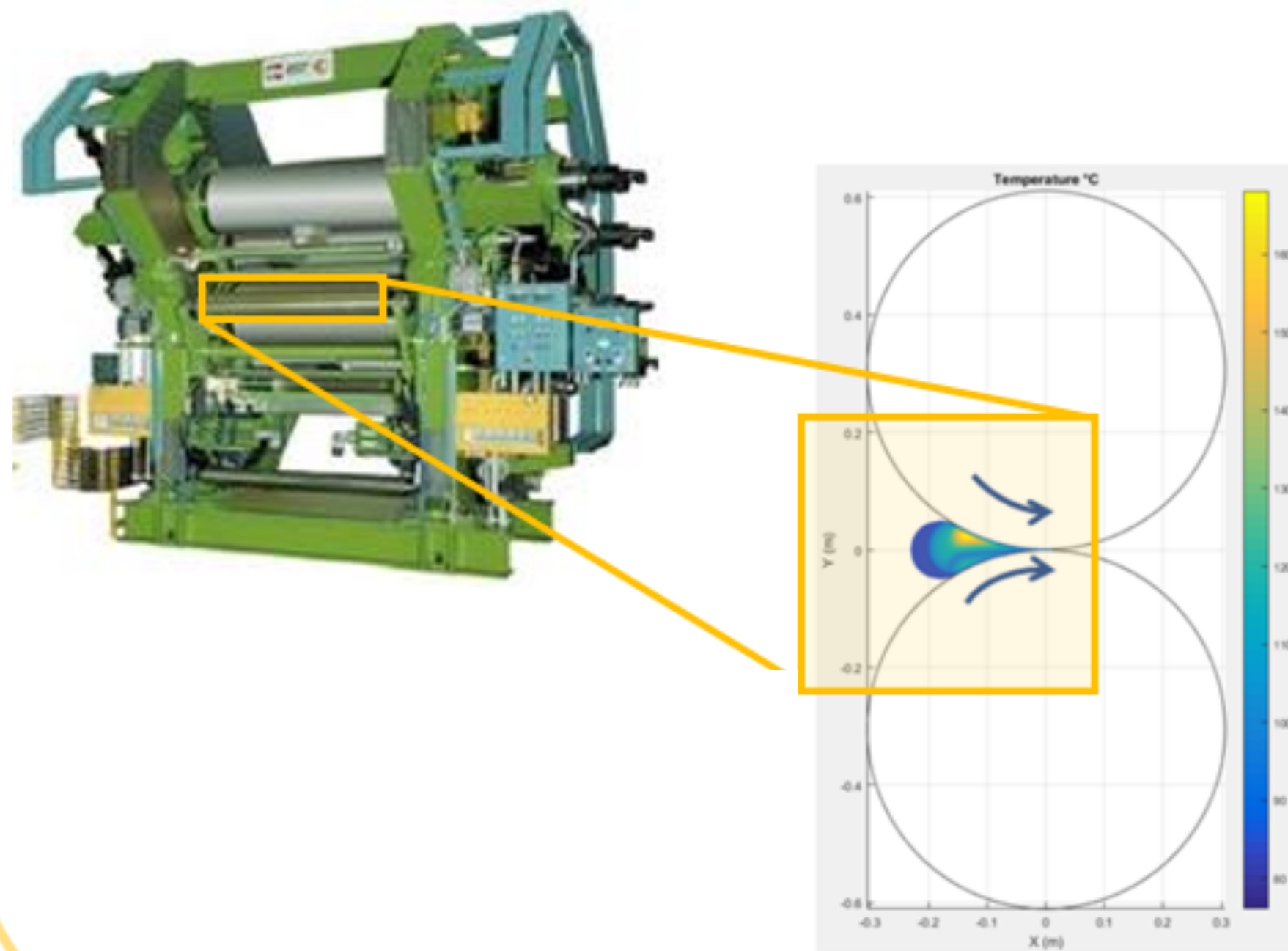


Credit: Antoine De Mathelin, PhD.

1

Industrial Motivations

Production



Credit: Khoa Nguyen, PhD.

◆ The final **quality** of the product strongly depends on raw materials (**composition/proportion**) and fabrication needs to be fully understood for automatic decision making

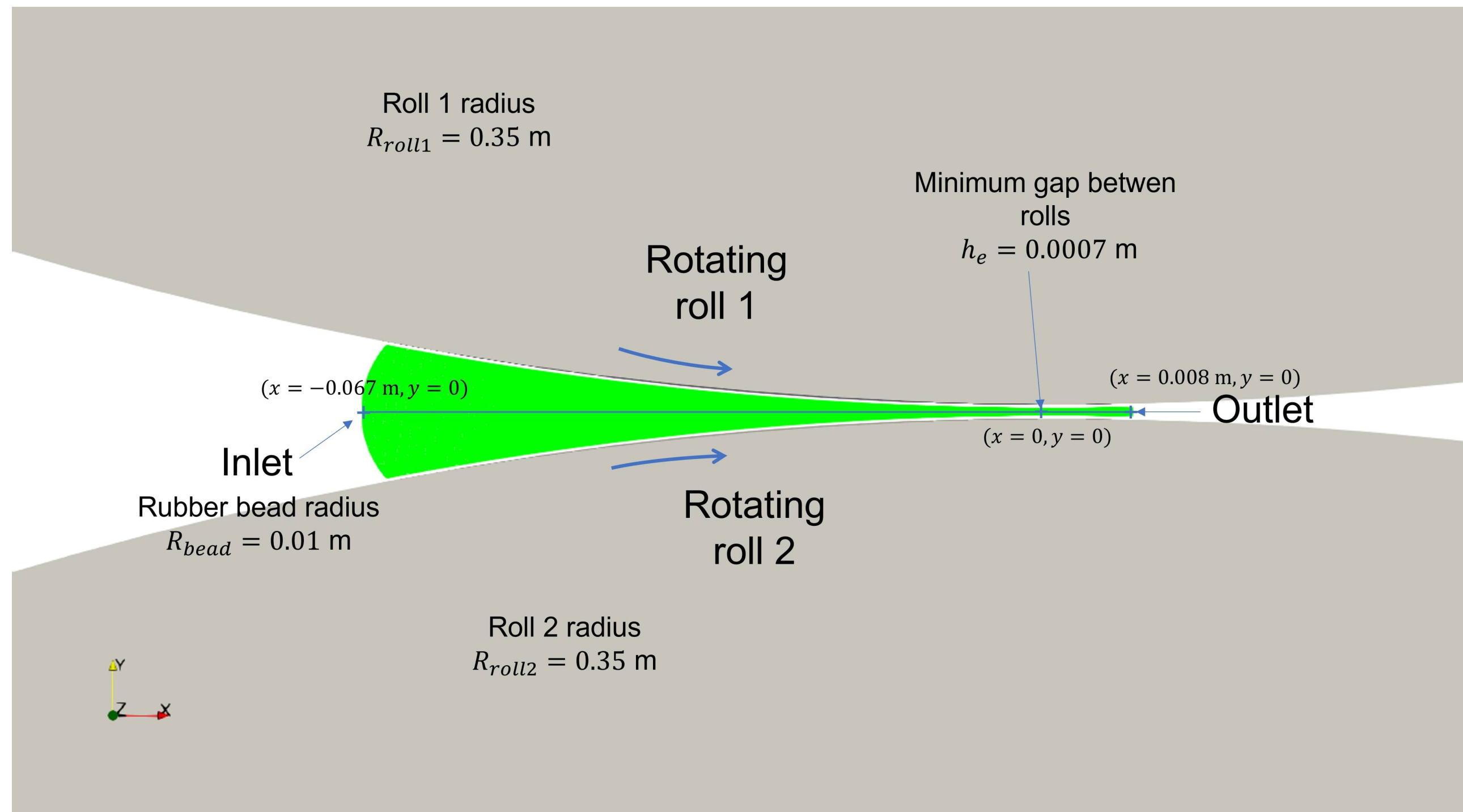
◆ The **data sets** are **small** or **moderate**

Physics + Data

1

Industrial Motivations

Application: The rubber calendering process



Prime interest : monitor the rubber **temperature** inside the **domain**

Create and **assemble** thin layers of **rubber**

The rubber is **smoothed** out through several rotating cylindrical rolls

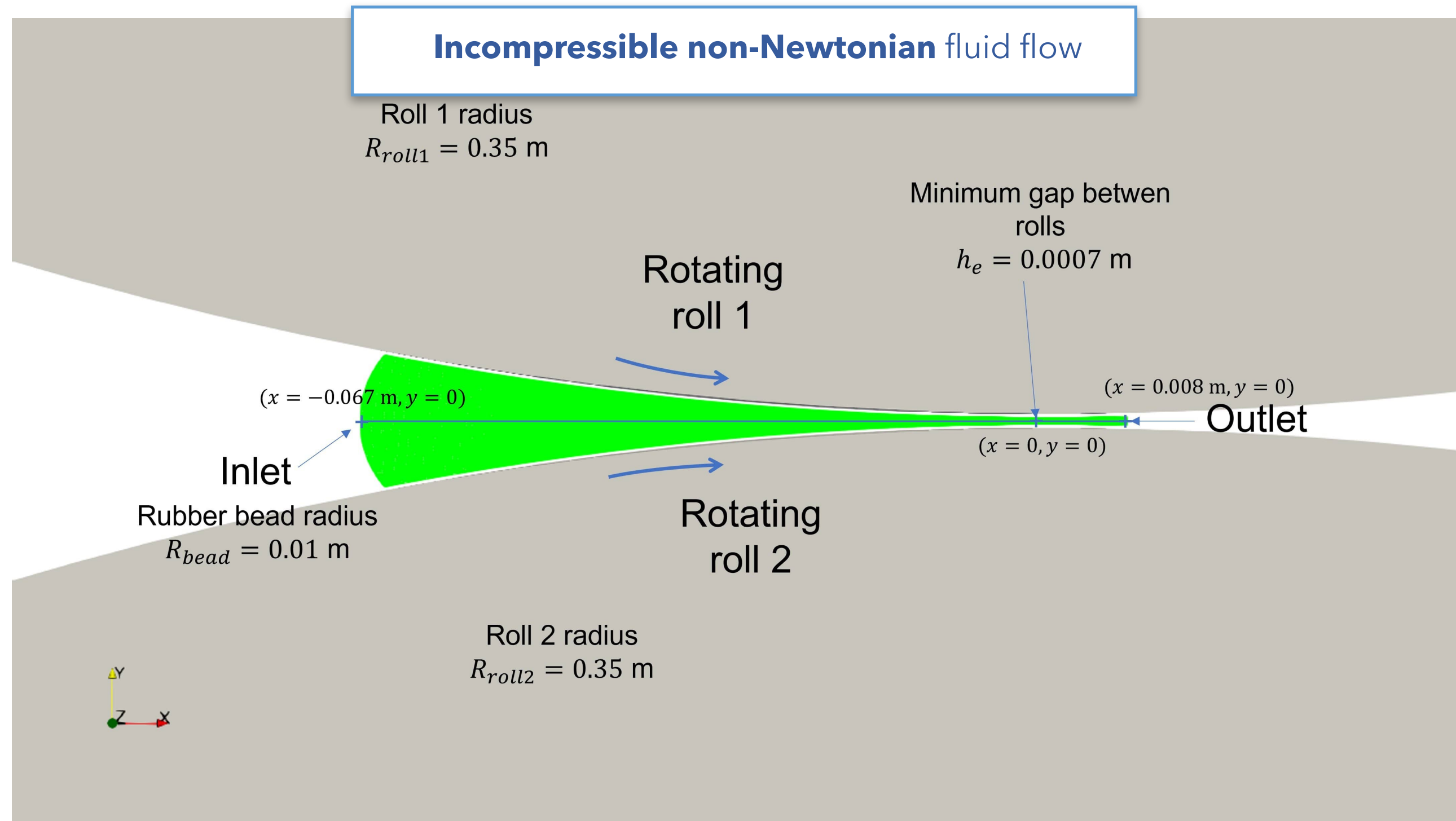
High temperatures could **deteriorate** the final material **quality**



1

Industrial Motivations

Application: The rubber calendering process



Prime interest : monitor the rubber **temperature** inside the **domain**

$$\begin{aligned} \nabla \cdot (2\eta(\mathbf{u}, T)\varepsilon(\mathbf{u})) - \nabla \mathbf{p} &= \vec{0} \rightarrow \\ \nabla \cdot \mathbf{u} &= 0 \rightarrow \\ \mathbf{u} \cdot \nabla T &= \frac{1}{Pe} \nabla^2 T + \frac{Br}{Pe} \eta(\mathbf{u}, T) |\gamma(\mathbf{u})|^2 \end{aligned}$$

$$\lambda = \left(\frac{1}{Pe}, \frac{Br}{Pe} \right)$$

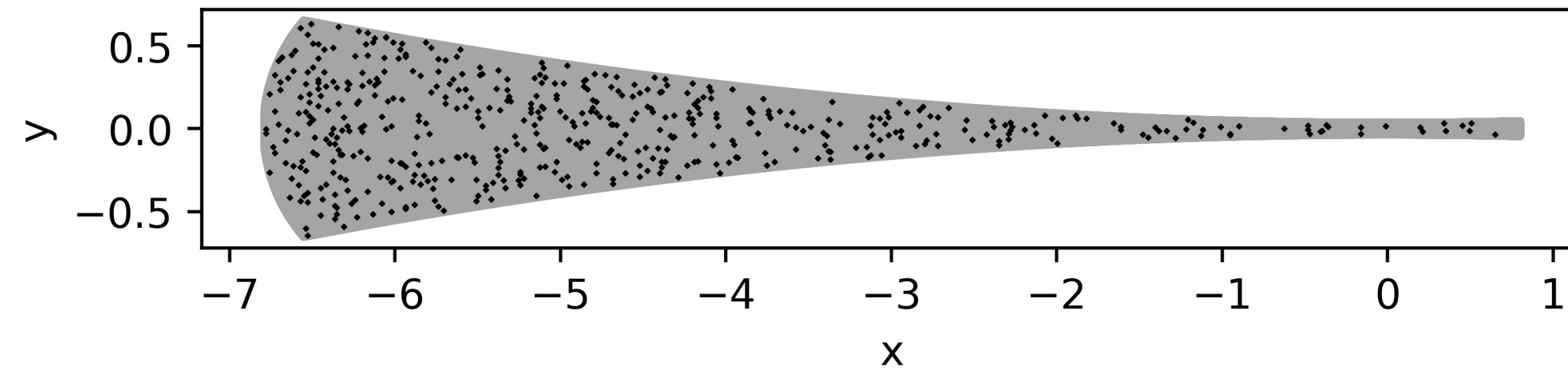
$$\begin{aligned} \eta(\mathbf{u}, T) &= |\gamma(\mathbf{u})|^{n-1} \exp \left(\frac{E_\alpha}{R} \left(\frac{1}{(T_2 - T_1)T + T_1} - \frac{1}{T_\alpha} \right) \right) \\ Pe &= \frac{\rho C_p u_0 L_0}{\lambda} \\ Br &= \frac{Ku_0^2}{\lambda(T_2 - T_1)} \end{aligned}$$



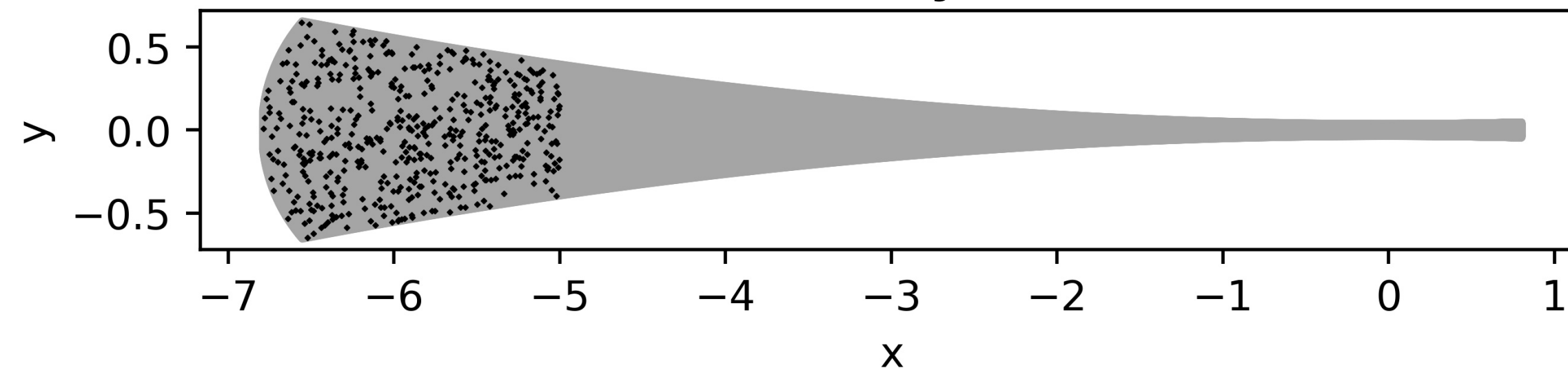
1 Industrial Motivations

Application: The rubber calendering process

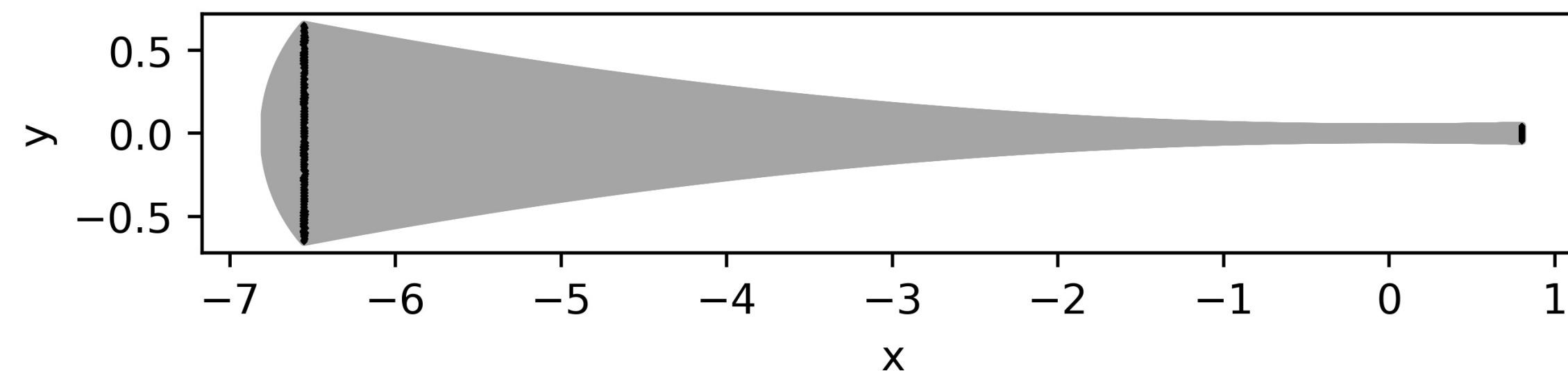
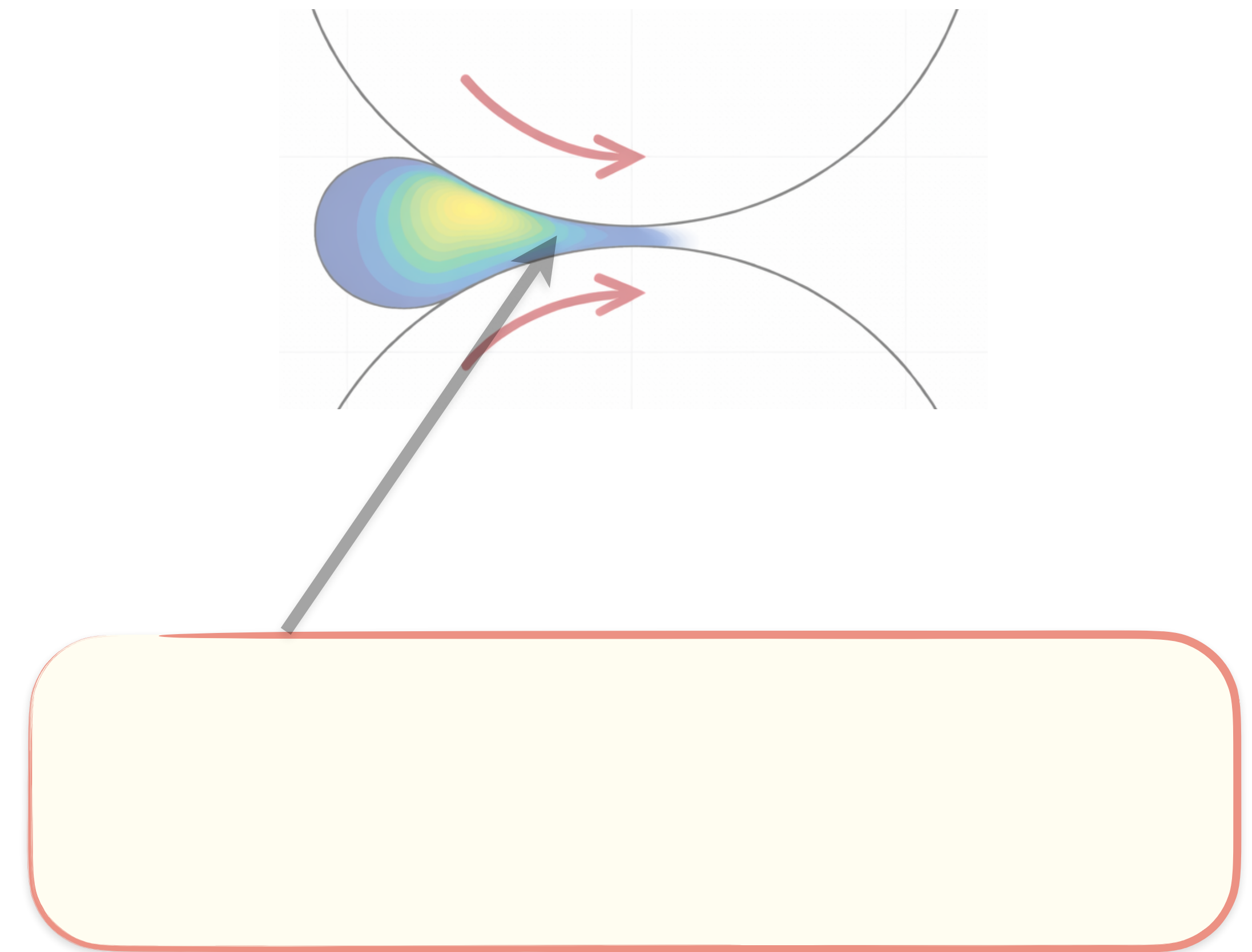
1. Measurements throughout the domain by **FEM solver**



2. Measurements on the **entry** of the domain

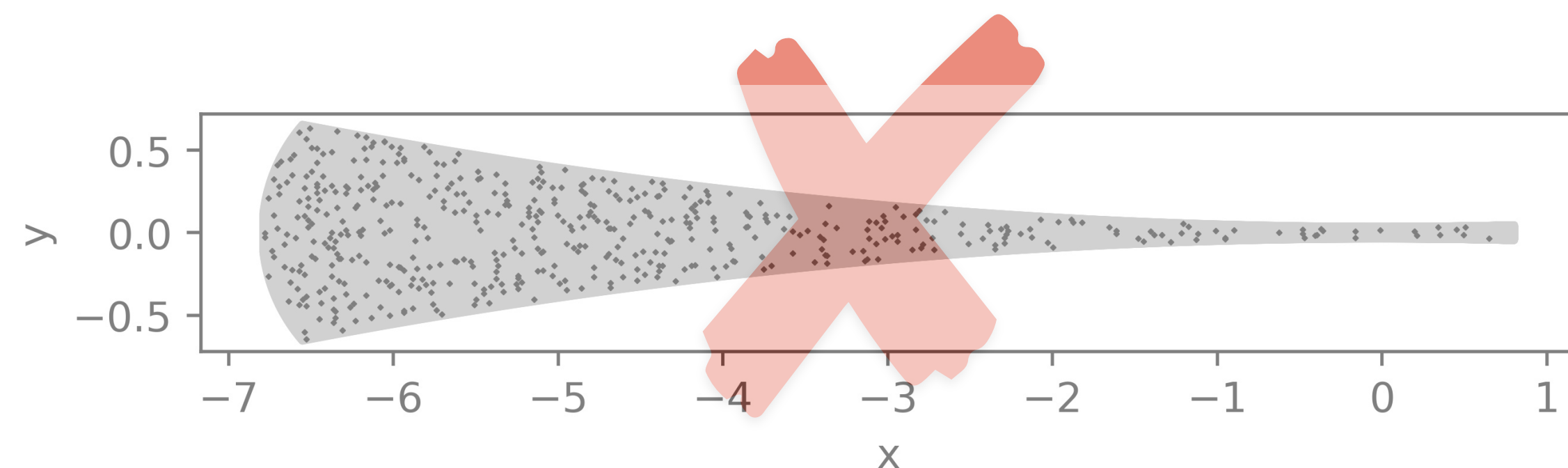


3. Measurements on **both** domain bounds

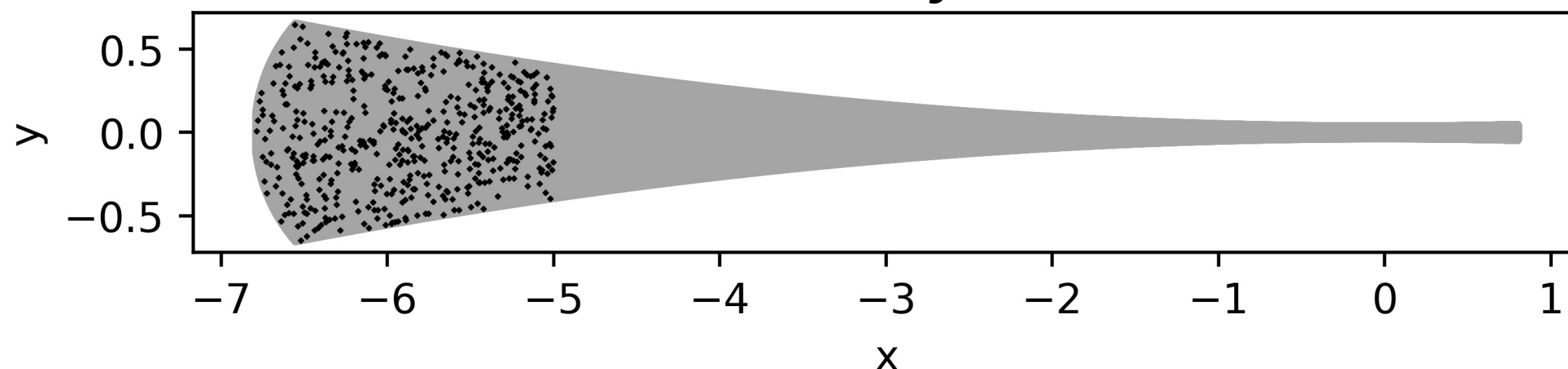


1 Industrial Motivations

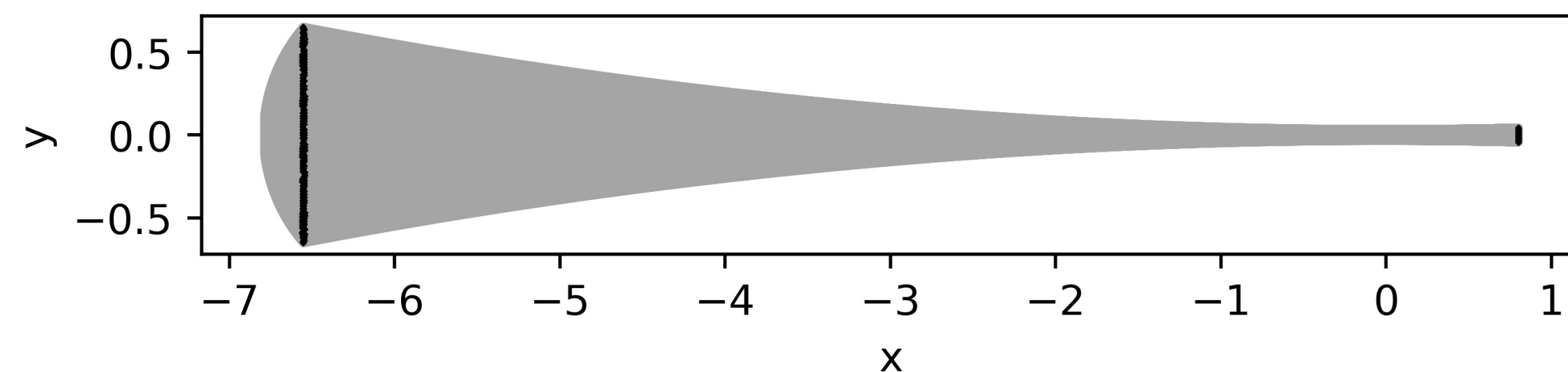
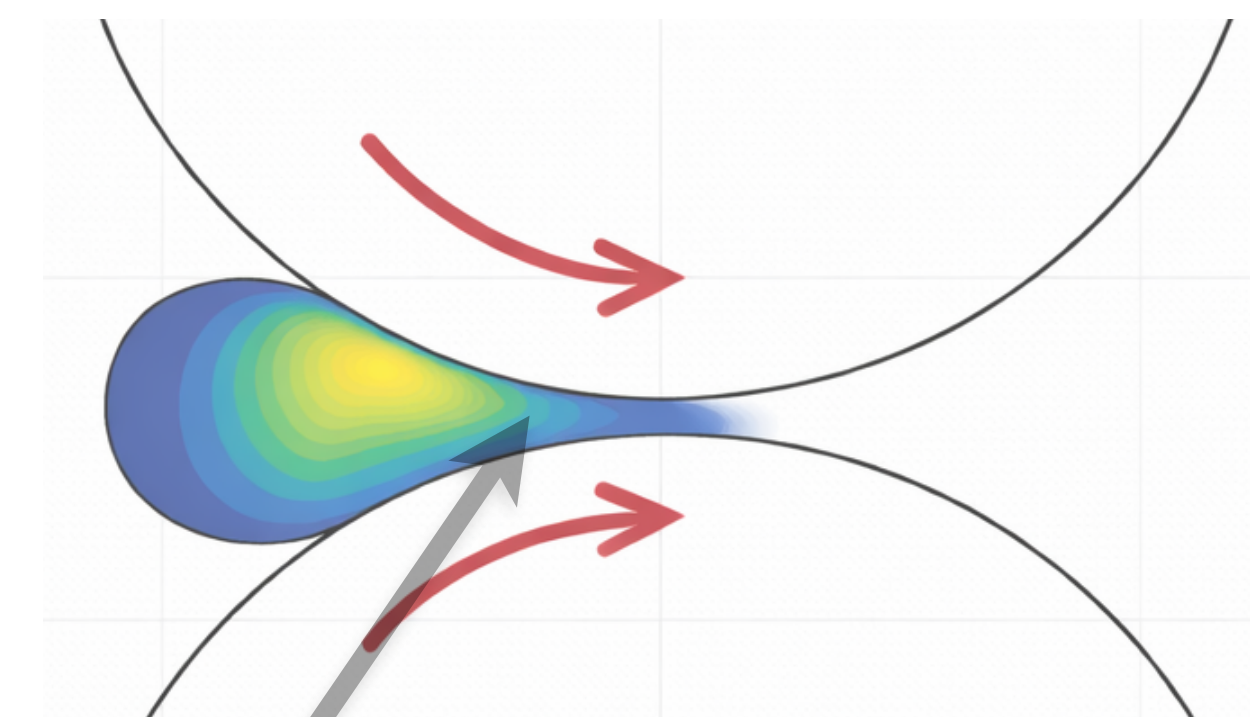
Application: The rubber calendering process



2. Measurements on the **entry** of the domain



3. Measurements on **both** domain bounds



Measuring the temperature on the interior of the rubber is **not physically possible** (we cannot place sensors inside the rubber).



1 Industrial Motivations

Summary

Product design

- ◆ New products are regularly manufactured with a **long** and **costly** process
- ◆ The **data sets** gathered during the development of products is relatively **small**

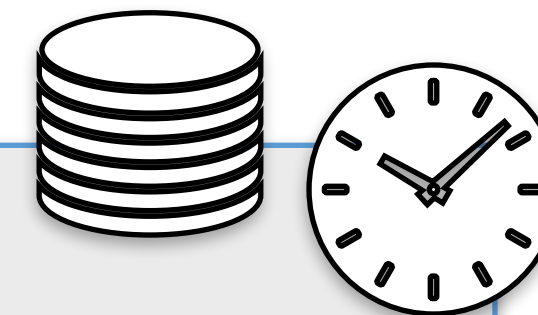
Production

- ◆ The final **quality** of the product strongly depends on raw materials (**composition**/proportion) and **fabrication** needs to be fully understood for automatic **decision** making
- ◆ The **data sets** are **small** or **moderate**

What are we looking for in ML ?



Less costly methods for product testing, conception, decision making, production, etc.



Ex.

Finding a **better initialization** for the in-house numerical solver

Faster results with **reasonable accuracy**

Hybrid modeling : a mix between **data** and **physics**

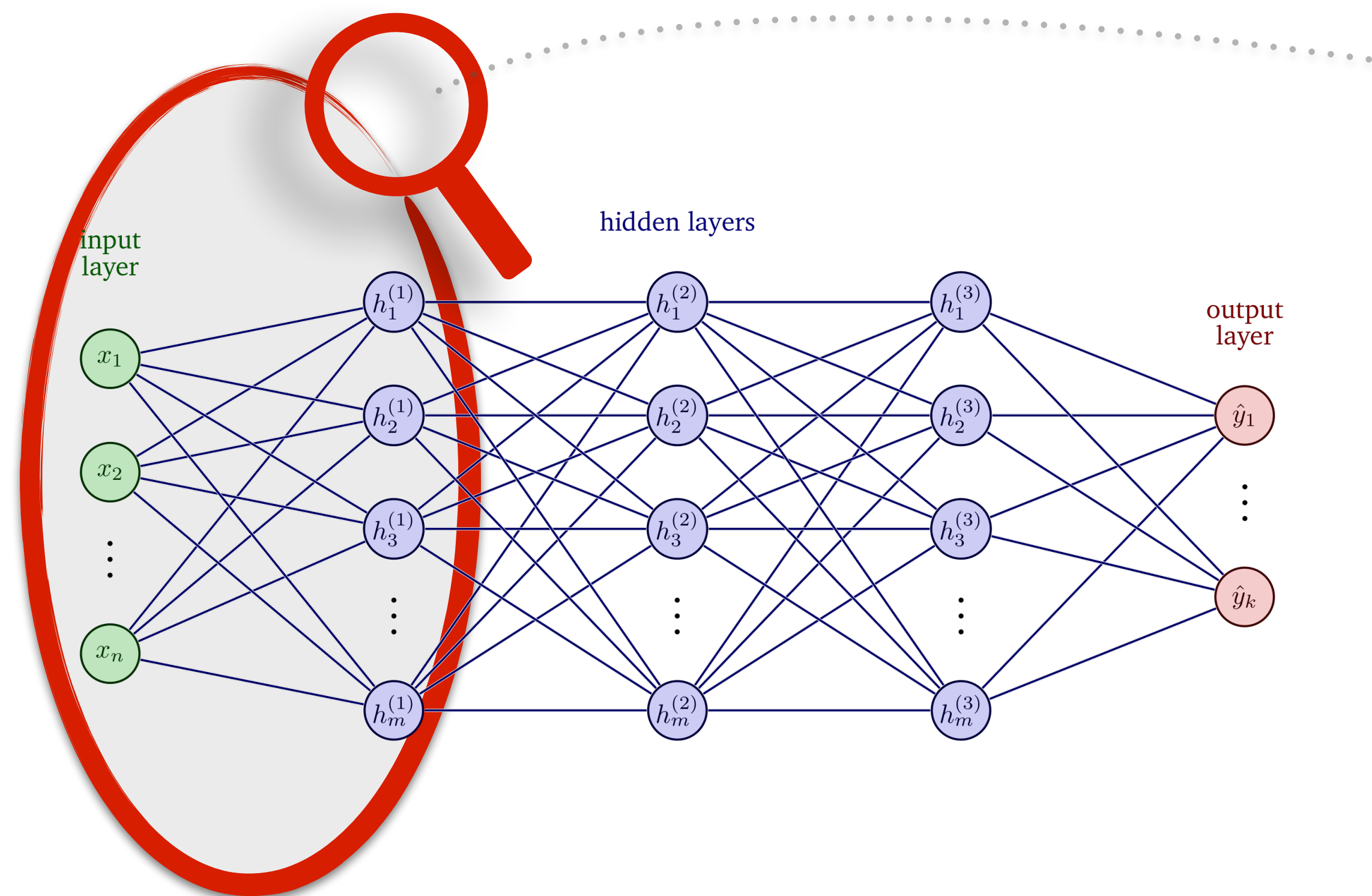
2

What is a PINN ? (Architecture)

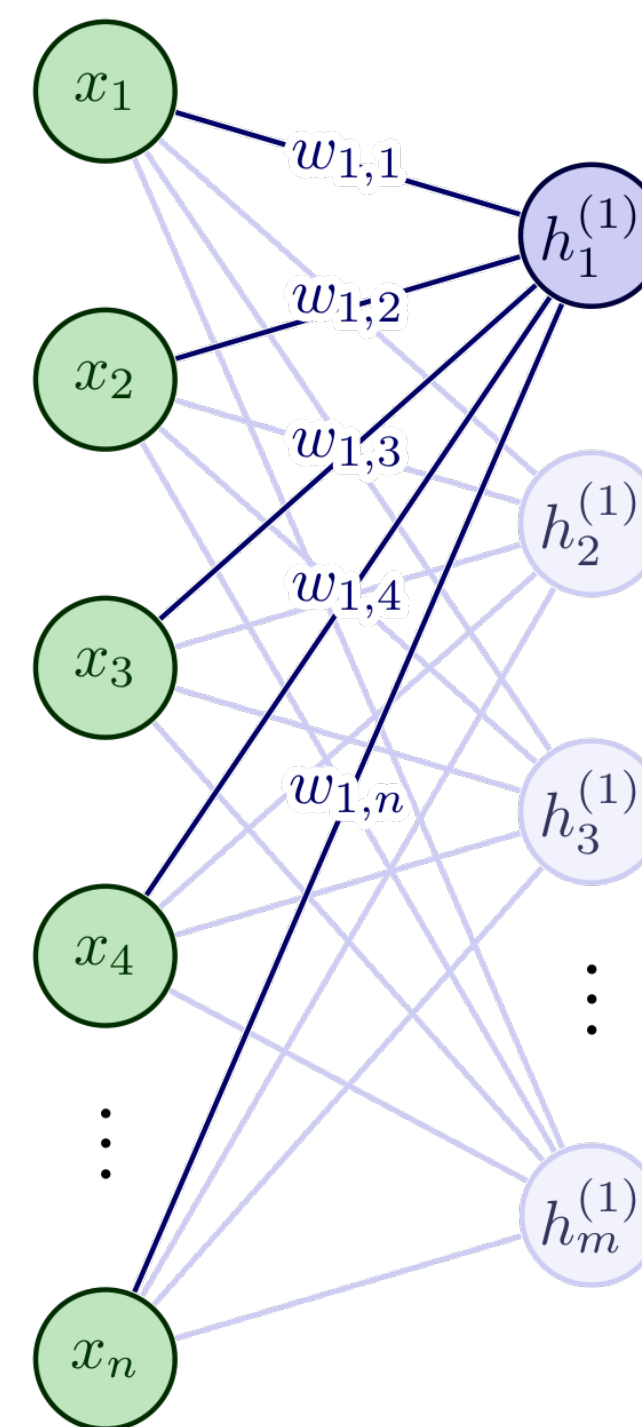
Model: Multi-Layer Perceptron MLP

Output at layer $h^{(l)}$ is given by

$$h^{(l)} = \sigma(W^{(l)}h^{(l-1)} + b^{(l)})$$



Trainable parameters $\theta = \{W^{(l)}, b^{(l)}\}_{l=1}^m$



$$\begin{pmatrix} h_1^{(1)} \\ h_2^{(1)} \\ \vdots \\ h_m^{(1)} \end{pmatrix} = \sigma \left[\begin{pmatrix} w_{1,1} & w_{1,2} & \dots & w_{1,n} \\ w_{2,1} & w_{2,2} & \dots & w_{2,n} \\ \vdots & \vdots & \ddots & \vdots \\ w_{m,1} & w_{m,2} & \dots & w_{m,n} \end{pmatrix} \begin{pmatrix} x_1 \\ x_2 \\ \vdots \\ x_n \end{pmatrix} + \begin{pmatrix} b_1^{(0)} \\ b_2^{(0)} \\ \vdots \\ b_m^{(0)} \end{pmatrix} \right]$$

$$\mathbf{h}^{(1)} = \sigma(\mathbf{W}^{(0)}\mathbf{x} + \mathbf{b}^{(0)})$$

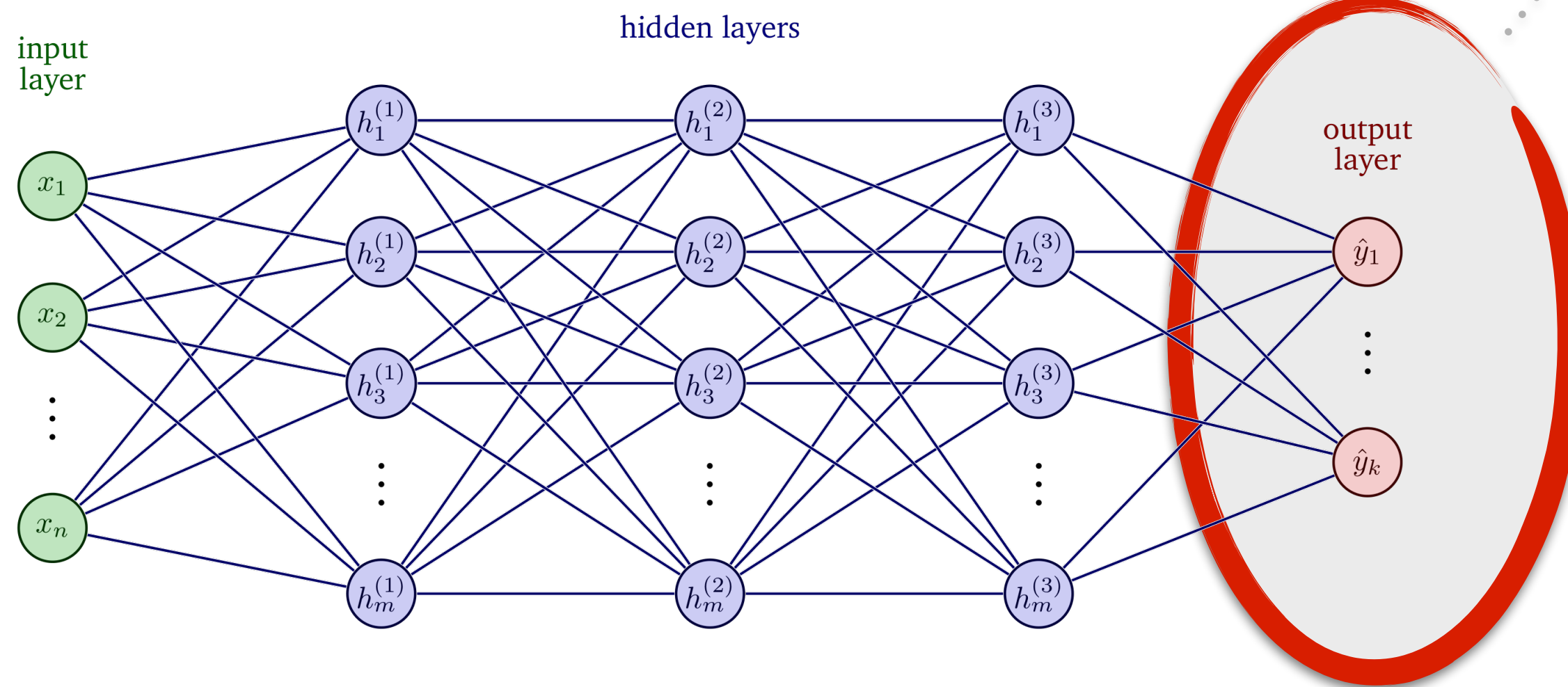
2 What is a PINN ? (Architecture)

Model: PINN = MLP + Physics (PDE)

Output at layer $h^{(l)}$ is given by

$$h^{(l)} = \sigma(W^{(l)}h^{(l-1)} + b^{(l)})$$

Output $\hat{u} = NN(x; \theta)$ is an **approximation** of the **physical solution** u of a partial differential equation

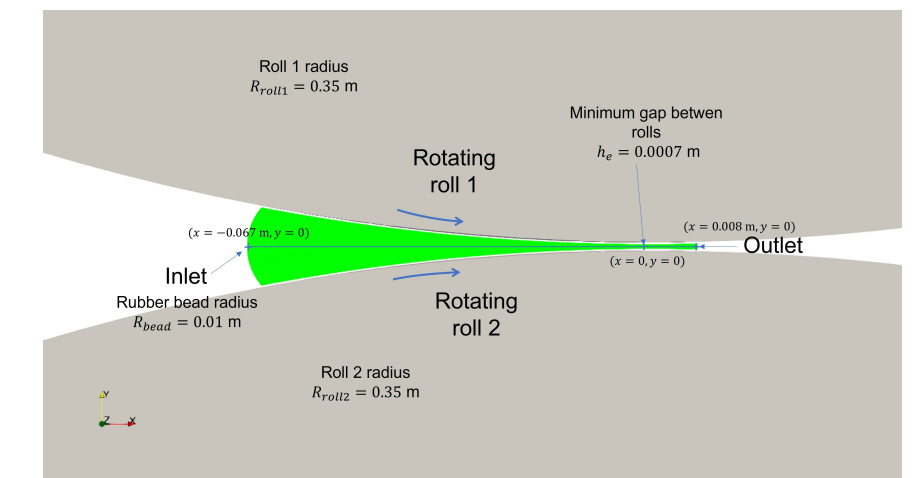


$$\hat{\mathbf{u}}, \hat{\mathbf{p}}, \hat{T} = NN(x; \theta)$$

$$\nabla \cdot (2\eta(\mathbf{u}, T)\varepsilon(\mathbf{u})) - \nabla \mathbf{p} = \vec{0}$$

$$\nabla \cdot \mathbf{u} = 0$$

$$\mathbf{u} \cdot \nabla T = \frac{1}{Pe} \nabla^2 T + \frac{Br}{Pe} \eta(\mathbf{u}, T) |\gamma(\mathbf{u})|^2$$



2 What is a PINN ?

Starting from NN Classical Loss Function

A **classical NN** is trained to minimize a **loss** function typically

$$L(\theta) = \frac{1}{N} \sum_{i=1}^N (NN(x_i, \theta) - y_i)^2$$

In other words,

$$\theta^* = \arg \min_{\theta} L(\theta)$$

Using **gradient descent**

$$\theta \leftarrow \theta - \lambda \nabla_{\theta} L(\theta)$$

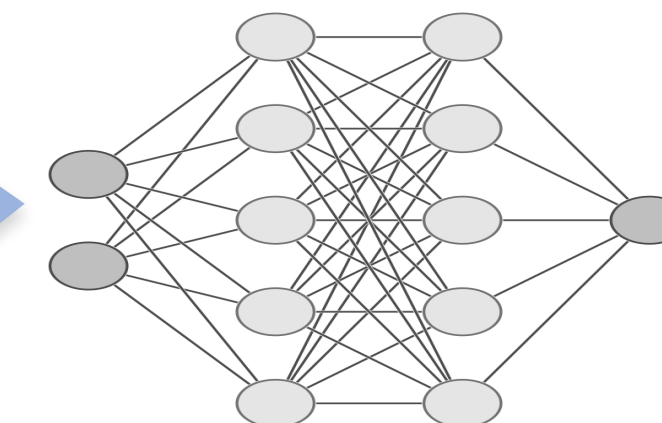

◆ Data Set. $S = \{(x_i, y_i)\}_{i=1}^{n,k}$

2

What is a PINN ?

To a dedicated PINN Loss Function

using a **NN** typically **MLP**



A **PINN** is trained to minimize a **loss** function typically

What are **Collocation** points ?



More details
later ...

Physics

$$L_{data}(\theta) = \frac{1}{N_{data}} \sum_{i=1}^{N_{data}} (NN(x_i, \theta) - y_i)^2$$

+

Collocation points

$$L_{pde}(\theta) = \frac{1}{N_{pde}} \sum_{i=1}^{N_{pde}} (\text{pde residual}(x_i, \theta))^2$$

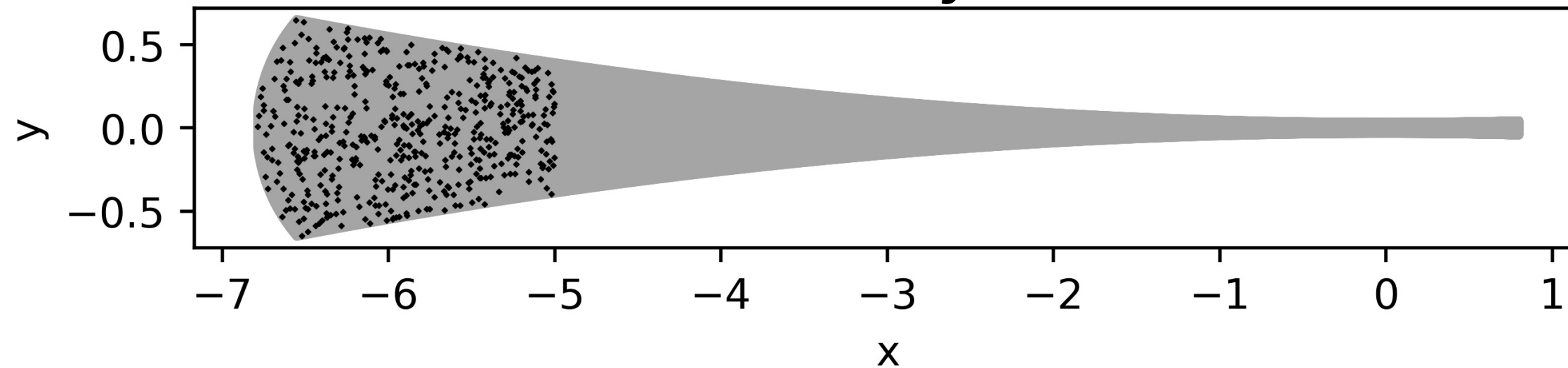
$$L_{bc}(\theta) = \frac{1}{N_{bc}} \sum_{i=1}^{N_{bc}} (NN(x_i, \theta) - BC_i)^2$$

2

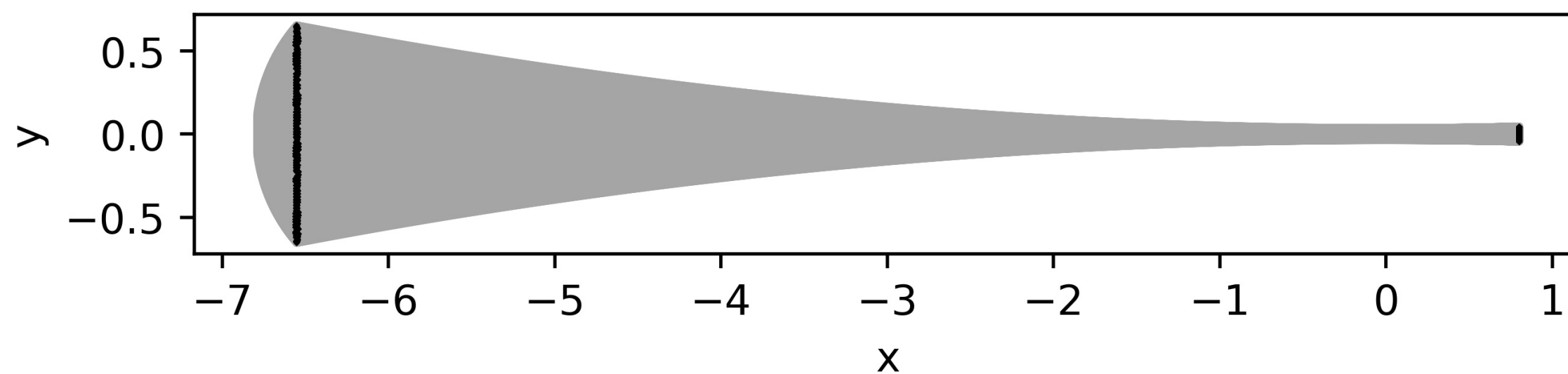
What is a PINN ?

To a dedicated PINN Loss Function

2. Measurements on the **entry** of the domain



3. Measurements on **both** domain bounds



A **PINN** is trained to minimize a **loss** function typically

$$L_{data}(\theta) = \frac{1}{N_{data}} \sum_{i=1}^{N_{data}} (NN(x_i, \theta) - y_i)^2$$



2

What is a PINN ?

To a dedicated PINN Loss Function

What is a **PDE Residual** ?

$$\nabla \cdot (2\eta(\mathbf{u}, T)\varepsilon(\mathbf{u})) - \nabla \mathbf{p} = \vec{0}$$

$$\nabla \cdot \mathbf{u} = 0$$

$$\mathbf{u} \cdot \nabla T = \frac{1}{Pe} \nabla^2 T + \frac{Br}{Pe} \eta(\mathbf{u}, T) |\gamma(\mathbf{u})|^2$$



$$\hat{\mathbf{u}}, \hat{\mathbf{p}}, \hat{T} = NN(x; \theta)$$

$$\mathbf{r}_1(x; \theta) = \nabla \cdot (2\eta(\hat{\mathbf{u}}, \hat{T})\varepsilon(\hat{\mathbf{u}})) - \nabla \hat{\mathbf{p}}$$

$$\mathbf{r}_2(x; \theta) = \nabla \cdot \hat{\mathbf{u}}$$

$$\mathbf{r}_3(x; \theta) = \hat{\mathbf{u}} \cdot \nabla \hat{T} - \frac{1}{Pe} \nabla^2 \hat{T} - \frac{Br}{Pe} \hat{\eta}(\hat{\mathbf{u}}, \hat{T}) |\gamma(\hat{\mathbf{u}})|^2$$

Physics

A **PINN** is trained to minimize a **loss** function typically



$$L_{pde}(\theta) = \frac{1}{N_{pde}} \sum_{i=1}^{N_{pde}} (\text{pde residual}(x_i, \theta))^2$$

$$L_{bc}(\theta) = \frac{1}{N_{bc}} \sum_{i=1}^{N_{bc}} (NN(x_i, \theta) - \text{BC}_i)^2$$



2

What is a PINN ?

To a dedicated **PINN Loss Function**



More details
later ...

Forward Problem

Physics

A **PINN** is trained to minimize a **loss** function typically

$$L_{data}(\theta) = \frac{1}{N_{data}} \sum_{i=1}^{N_{data}} (NN(x_i, \theta) - y_i)^2$$

Inverse Problem

+

$$L_{pde}(\theta) = \frac{1}{N_{pde}} \sum_{i=1}^{N_{pde}} (\text{pde residual}(x_i, \theta))^2$$

$$L_{bc}(\theta) = \frac{1}{N_{bc}} \sum_{i=1}^{N_{bc}} (NN(x_i, \theta) - \text{BC}_i)^2$$

2

What is a PINN ?

Optimization

A **classical NN** is trained to minimize a **loss** function typically

$$L(\theta) = \frac{1}{N} \sum_{i=1}^N (NN(x_i, \theta) - y_i)^2$$

A **PINN** is trained to minimize a **loss** function typically

$$L_{pinn}(\theta) = w_{data}L_{data}(\theta) + w_{pde}L_{pde}(\theta) + w_{bc}L_{bc}(\theta)$$

Solve $\theta^* = \arg \min_{\theta} L(\theta)$

Using **gradient descent** $\theta \leftarrow \theta - \lambda \nabla_{\theta} L(\theta)$
 Step-size

Loss weights



Optimizers used for PINNs

Adam LBFGS **ADAM + LBFGS** **NatGrad** NTK NNTK ...

Most common

Our Choice

 AnaGRAM, ENGD, ...

3

How to determine the physical loss L_{pinn} ?

Forward Problem

Complete / Partial physical knowledge
(Partial Differential Equation)



Small or moderate data set $S = \{(x_i, y_i)\}_{i=1}^{n,k}$
No available **data / measurements**

$$\begin{cases} \mathcal{F}[u](x) = f(x), & x \in \Omega \\ \mathcal{B}[u](x) = b(x), & x \in \partial\Omega \end{cases}$$

3

How to determine the physical loss L_{pinn} ?

Forward Problem

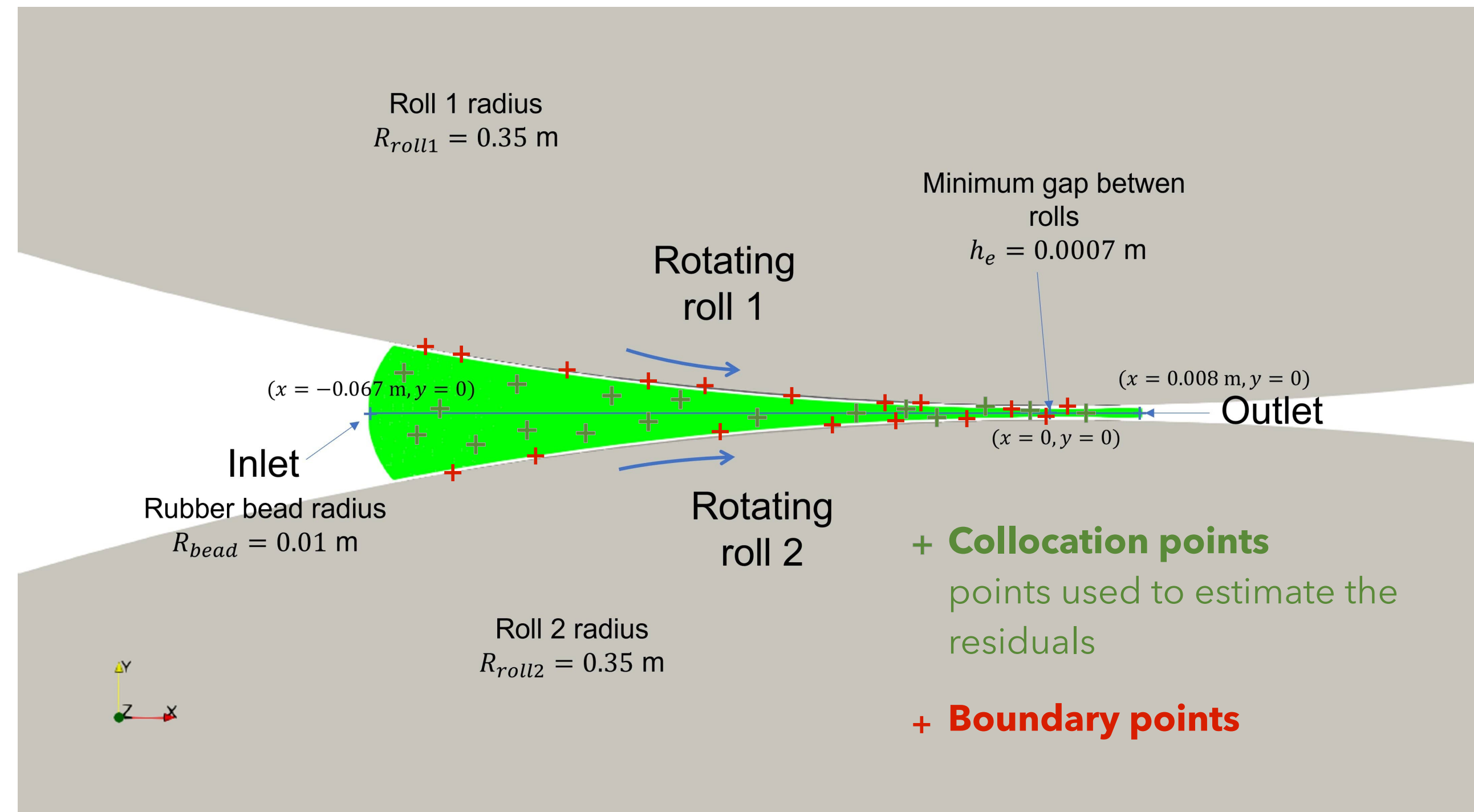
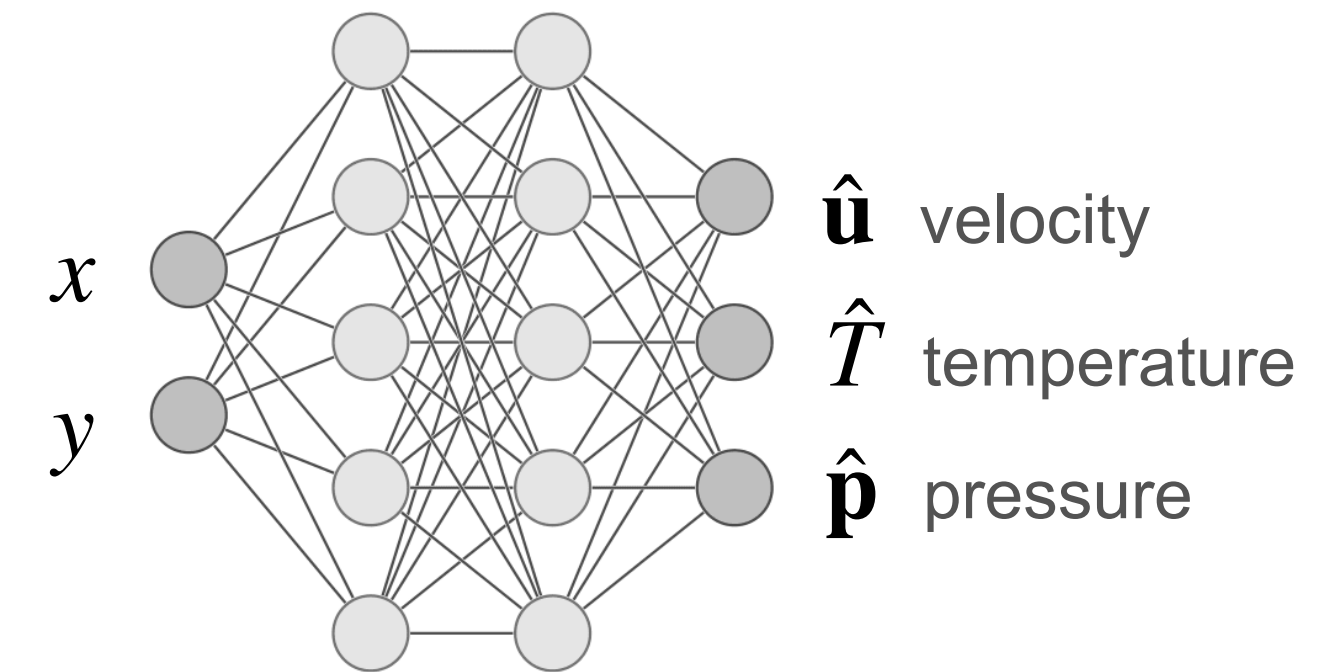
Known

$$\nabla \cdot (2\eta(\mathbf{u}, T)\varepsilon(\mathbf{u})) - \nabla \mathbf{p} = \vec{0}$$

$$\nabla \cdot \mathbf{u} = 0$$

$$\mathbf{u} \cdot \nabla T = \frac{1}{Pe} \nabla^2 T + \frac{Br}{Pe} \eta(\mathbf{u}, T) |\gamma(\mathbf{u})|^2$$

$$\lambda = \left(\frac{1}{Pe}, \frac{Br}{Pe} \right)$$



3

How to determine the physical loss L_{pinn} ?

Forward Problem

$$\nabla \cdot (2\eta(\mathbf{u}, T)\varepsilon(\mathbf{u})) - \nabla \mathbf{p} = \vec{0}$$

$$\nabla \cdot \mathbf{u} = 0$$

$$\mathbf{u} \cdot \nabla T = \frac{1}{Pe} \nabla^2 T + \frac{Br}{Pe} \eta(\mathbf{u}, T) |\gamma(\mathbf{u})|^2$$

Easy to compute

Automatic differentiation

PDE Residuals

$$r_1^1 = \frac{\partial}{\partial x} \left(2\eta(\hat{u}, \hat{T}) \frac{\partial \hat{u}_x}{\partial x} \right) + \frac{\partial}{\partial y} \left(\eta(\hat{u}, \hat{T}) \left(\frac{\partial \hat{u}_x}{\partial y} + \frac{\partial \hat{u}_y}{\partial x} \right) \right) - \frac{\partial \hat{p}}{\partial x}$$

$$r_1^2 = \frac{\partial}{\partial x} \left(\eta(\hat{u}, \hat{T}) \left(\frac{\partial \hat{u}_x}{\partial y} + \frac{\partial \hat{u}_y}{\partial x} \right) \right) + \frac{\partial}{\partial y} \left(2\eta(\hat{u}, \hat{T}) \frac{\partial \hat{u}_y}{\partial y} \right) - \frac{\partial \hat{p}}{\partial y}$$

$$r_2 = \frac{\partial \hat{u}_x}{\partial x} + \frac{\partial \hat{u}_y}{\partial y}$$

$$r_3 = \hat{u}_x \frac{\partial \hat{T}}{\partial x} + \hat{u}_y \frac{\partial \hat{T}}{\partial y} - \frac{1}{Pe} \left(\frac{\partial^2 \hat{T}}{\partial x^2} + \frac{\partial^2 \hat{T}}{\partial y^2} \right) - \frac{Br}{Pe} \eta(\hat{u}, \hat{T}) |\gamma(\hat{u})|^2$$

$$L_{pde}(\theta) = \frac{1}{N} \sum_{i=1}^N \left[r_1^1(x_i, y_i; \theta) + r_1^2(x_i, y_i; \theta) + r_2(x_i, y_i; \theta) + r_3(x_i, y_i; \theta) \right]^2$$

3

How to determine the physical loss L_{pinn} ?

Forward Problem

$$\nabla \cdot (2\eta(\mathbf{u}, T)\varepsilon(\mathbf{u})) - \nabla \mathbf{p} = \vec{0}$$

$$\nabla \cdot \mathbf{u} = 0$$

$$\mathbf{u} \cdot \nabla T = \frac{1}{Pe} \nabla^2 T + \frac{Br}{Pe} \eta(\mathbf{u}, T) |\gamma(\mathbf{u})|^2$$

Total PINN Loss

$$L_{pinn}(\theta) = w_{pde} L_{pde}(\theta) + w_{bc} L_{bc}(\theta)$$

$$+ w_{data} L_{data}(\theta)$$



PDE Residuals

$$r_1^1 = \frac{\partial}{\partial x} \left(2\eta(\hat{u}, \hat{T}) \frac{\partial \hat{u}_x}{\partial x} \right) + \frac{\partial}{\partial y} \left(\eta(\hat{u}, \hat{T}) \left(\frac{\partial \hat{u}_x}{\partial y} + \frac{\partial \hat{u}_y}{\partial x} \right) \right) - \frac{\partial \hat{p}}{\partial x}$$

$$r_1^2 = \frac{\partial}{\partial x} \left(\eta(\hat{u}, \hat{T}) \left(\frac{\partial \hat{u}_x}{\partial y} + \frac{\partial \hat{u}_y}{\partial x} \right) \right) + \frac{\partial}{\partial y} \left(2\eta(\hat{u}, \hat{T}) \frac{\partial \hat{u}_y}{\partial y} \right) - \frac{\partial \hat{p}}{\partial y}$$

$$r_2 = \frac{\partial \hat{u}_x}{\partial x} + \frac{\partial \hat{u}_y}{\partial y}$$

$$r_3 = \hat{u}_x \frac{\partial \hat{T}}{\partial x} + \hat{u}_y \frac{\partial \hat{T}}{\partial y} - \frac{1}{Pe} \left(\frac{\partial^2 \hat{T}}{\partial x^2} + \frac{\partial^2 \hat{T}}{\partial y^2} \right) - \frac{Br}{Pe} \eta(\hat{u}, \hat{T}) |\gamma(\hat{u})|^2$$

$$L_{pde}(\theta) = \frac{1}{N} \sum_{i=1}^N [r_1^1(x_i, y_i; \theta) + r_1^2(x_i, y_i; \theta) + r_2(x_i, y_i; \theta) + r_3(x_i, y_i; \theta)]$$



3

How to determine the physical loss L_{pinn} ?

Inverse Problem

Partial physical knowledge
(Partial Differential Equation)



Small or **moderate** data set $S = \{(x_i, y_i)\}_{i=1}^{n,k}$
Available **measurements**

Unknown parameter

$$\begin{cases} \mathcal{F}[u](x; \underline{\lambda}) = f(x), & x \in \Omega \\ \mathcal{B}[u](x) = b(x), & x \in \partial\Omega \end{cases}$$

3

How to determine the physical loss L_{pinn} ?

Inverse Problem

$$\nabla \cdot (2\eta(\mathbf{u}, T)\varepsilon(\mathbf{u})) - \nabla \mathbf{p} = \vec{0}$$

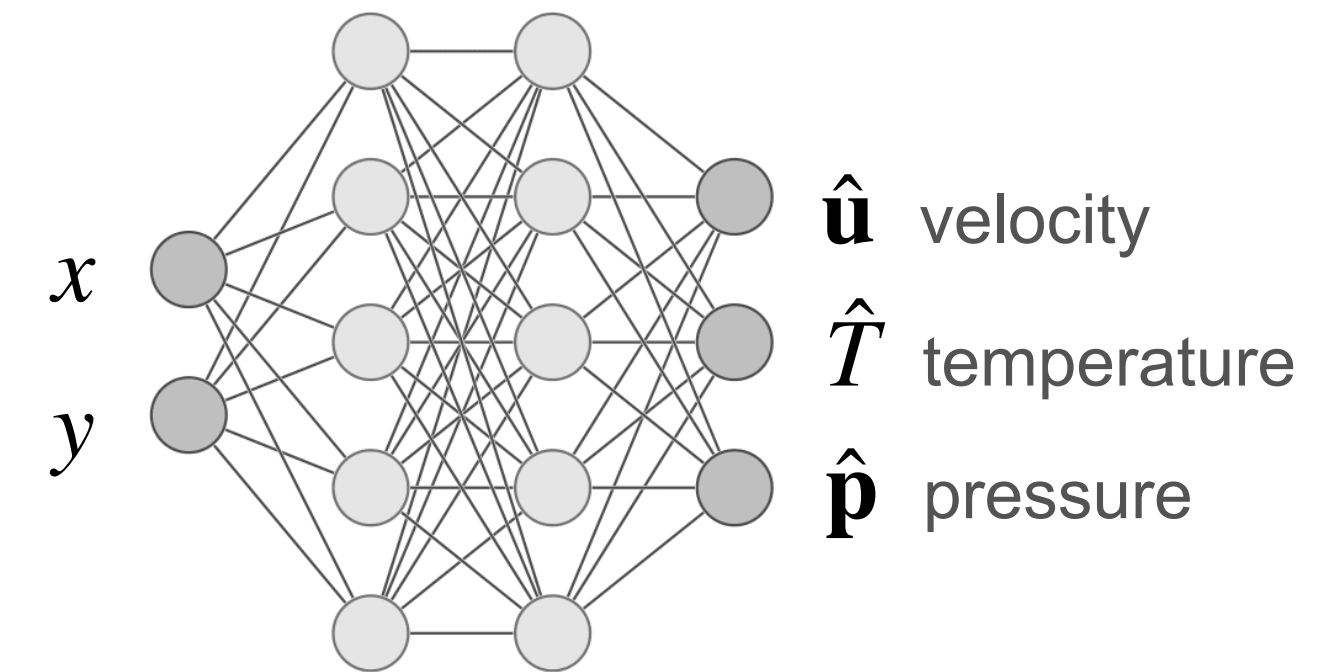
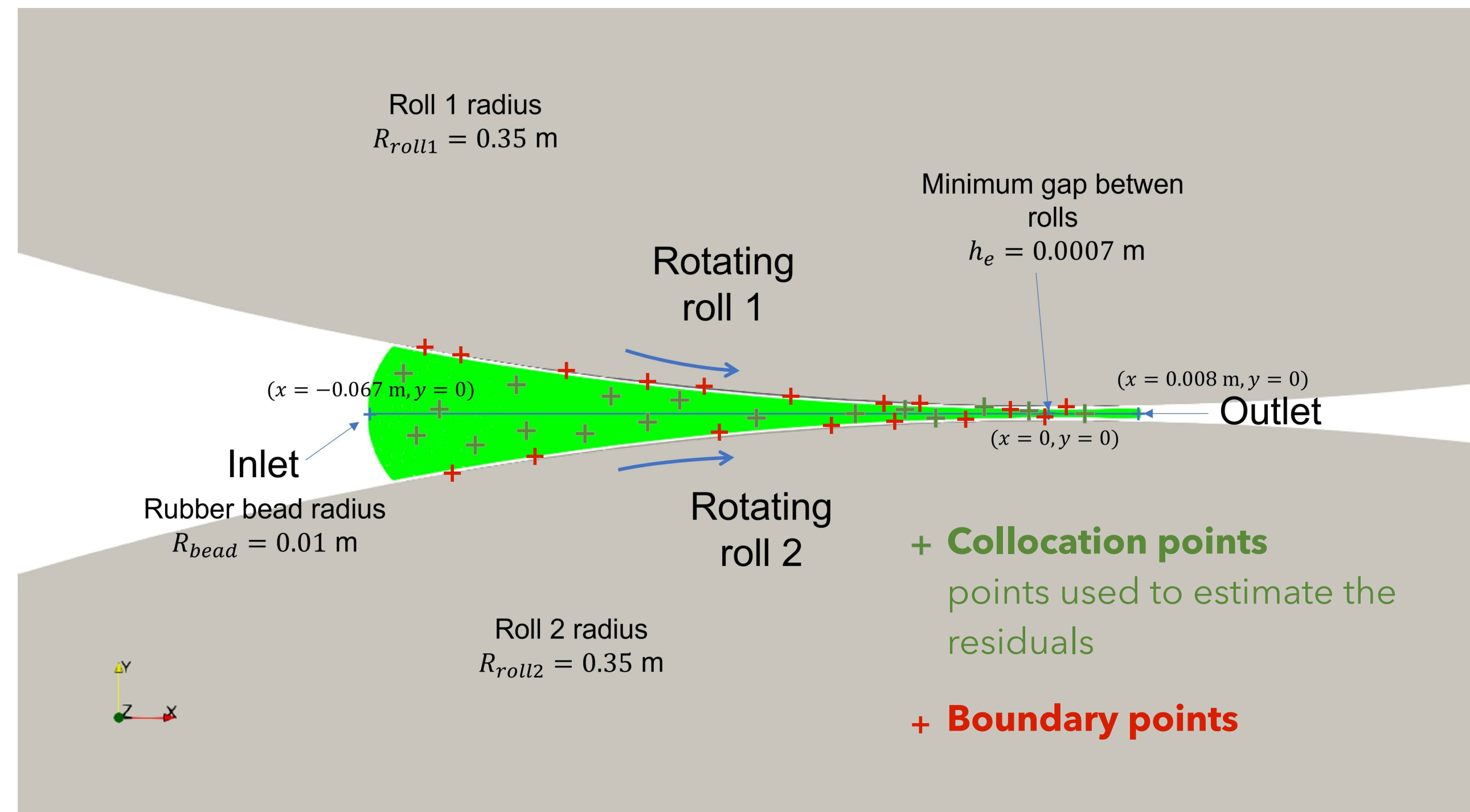
$$\nabla \cdot \mathbf{u} = 0$$

$$\mathbf{u} \cdot \nabla T = \frac{1}{Pe} \nabla^2 T + \frac{Br}{Pe} \eta(\mathbf{u}, T) |\gamma(\mathbf{u})|^2$$

Unknown

$$\lambda = \left(\frac{1}{Pe}, \frac{Br}{Pe} \right)$$

Trainable parameters $\theta = \{W^{(l)}, b^{(l)}, \lambda\}_{l=1}^m$



3

How to determine the physical loss L_{pinn} ?

Inverse Problem

$$\nabla \cdot (2\eta(\mathbf{u}, T)\varepsilon(\mathbf{u})) - \nabla \mathbf{p} = \vec{0}$$

$$\nabla \cdot \mathbf{u} = 0$$

$$\mathbf{u} \cdot \nabla T = \frac{1}{Pe} \nabla^2 T + \frac{Br}{Pe} \eta(\mathbf{u}, T) |\gamma(\mathbf{u})|^2$$

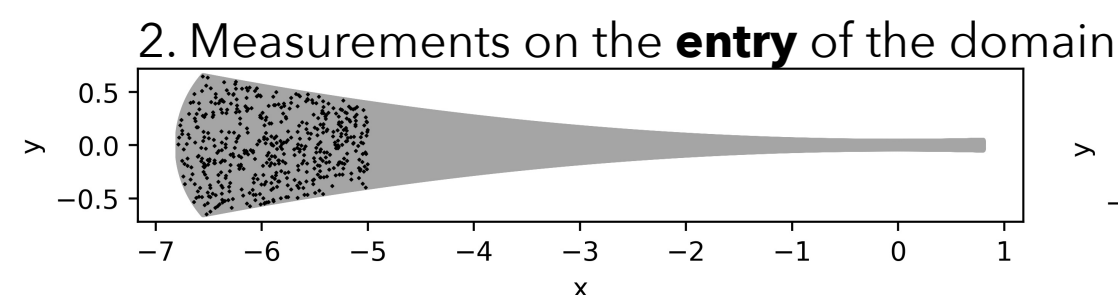
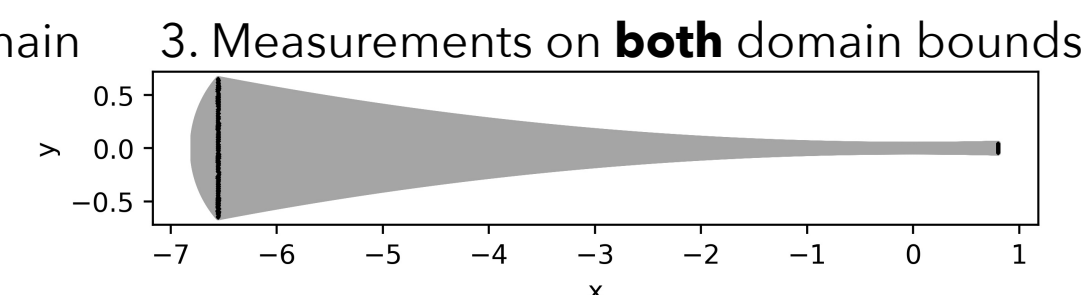
Unknown

$$\lambda = \left(\frac{1}{Pe}, \frac{Br}{Pe} \right)$$

Total PINN Loss

$$L_{pinn}(\theta) = w_{pde} L_{pde}(\theta) + w_{bc} L_{bc}(\theta)$$

$$+ w_{data} L_{data}(\theta)$$



PDE Residuals

$$r_1^1 = \frac{\partial}{\partial x} \left(2\eta(\hat{u}, \hat{T}) \frac{\partial \hat{u}_x}{\partial x} \right) + \frac{\partial}{\partial y} \left(\eta(\hat{u}, \hat{T}) \left(\frac{\partial \hat{u}_x}{\partial y} + \frac{\partial \hat{u}_y}{\partial x} \right) \right) - \frac{\partial \hat{p}}{\partial x}$$

$$r_1^2 = \frac{\partial}{\partial x} \left(\eta(\hat{u}, \hat{T}) \left(\frac{\partial \hat{u}_x}{\partial y} + \frac{\partial \hat{u}_y}{\partial x} \right) \right) + \frac{\partial}{\partial y} \left(2\eta(\hat{u}, \hat{T}) \frac{\partial \hat{u}_y}{\partial y} \right) - \frac{\partial \hat{p}}{\partial y}$$

$$r_2 = \frac{\partial \hat{u}_x}{\partial x} + \frac{\partial \hat{u}_y}{\partial y}$$

$$r_3 = \hat{u}_x \frac{\partial \hat{T}}{\partial x} + \hat{u}_y \frac{\partial \hat{T}}{\partial y} - \frac{1}{Pe} \left(\frac{\partial^2 \hat{T}}{\partial x^2} + \frac{\partial^2 \hat{T}}{\partial y^2} \right) - \frac{Br}{Pe} \eta(\hat{u}, \hat{T}) |\gamma(\hat{u})|^2$$

$$L_{pde}(\theta) = \frac{1}{N} \sum_{i=1}^N [r_1^1(x_i, y_i; \theta) + r_1^2(x_i, y_i; \theta) + r_2(x_i, y_i; \theta) + r_3(x_i, y_i; \theta)]$$



4

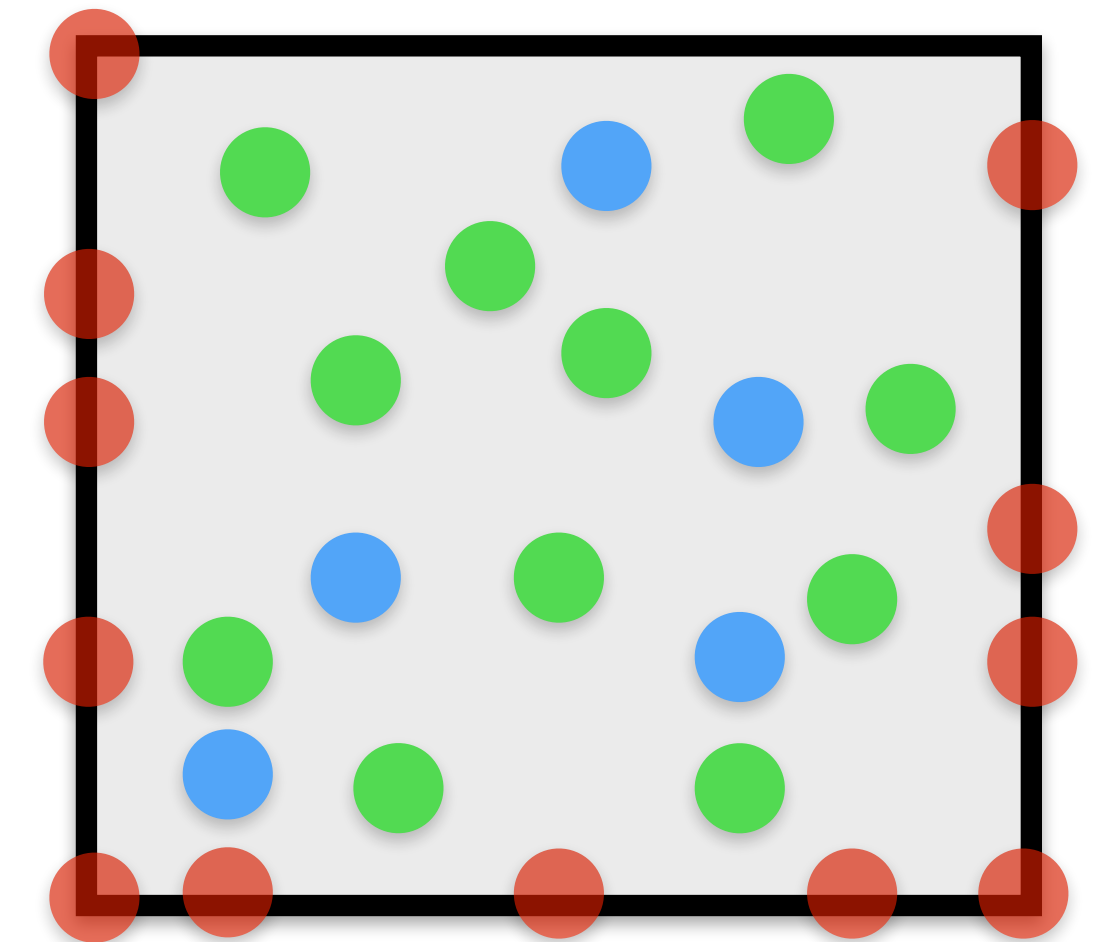
Approximating the PDE residuals

The PINN is trained to **minimize** a **physical loss** function composed of the **PDE residuals** on the domain Ω

$$R(u) = \int_{\Omega} \left(\mathcal{F}[u](x) - f(x) \right)^2 dx$$

and at the boundary $\partial\Omega$

$$R_{bc}(u) = \int_{\partial\Omega} \left(\mathcal{B}[u](x) - b(x) \right)^2 dx$$



● Supervised points $\{x_i\}_{i=1}^{N_{data}}$

● Collocation points $\{x_i\}_{i=1}^N$

● Boundary points $\{x_i\}_{i=1}^{N_{bc}}$

4

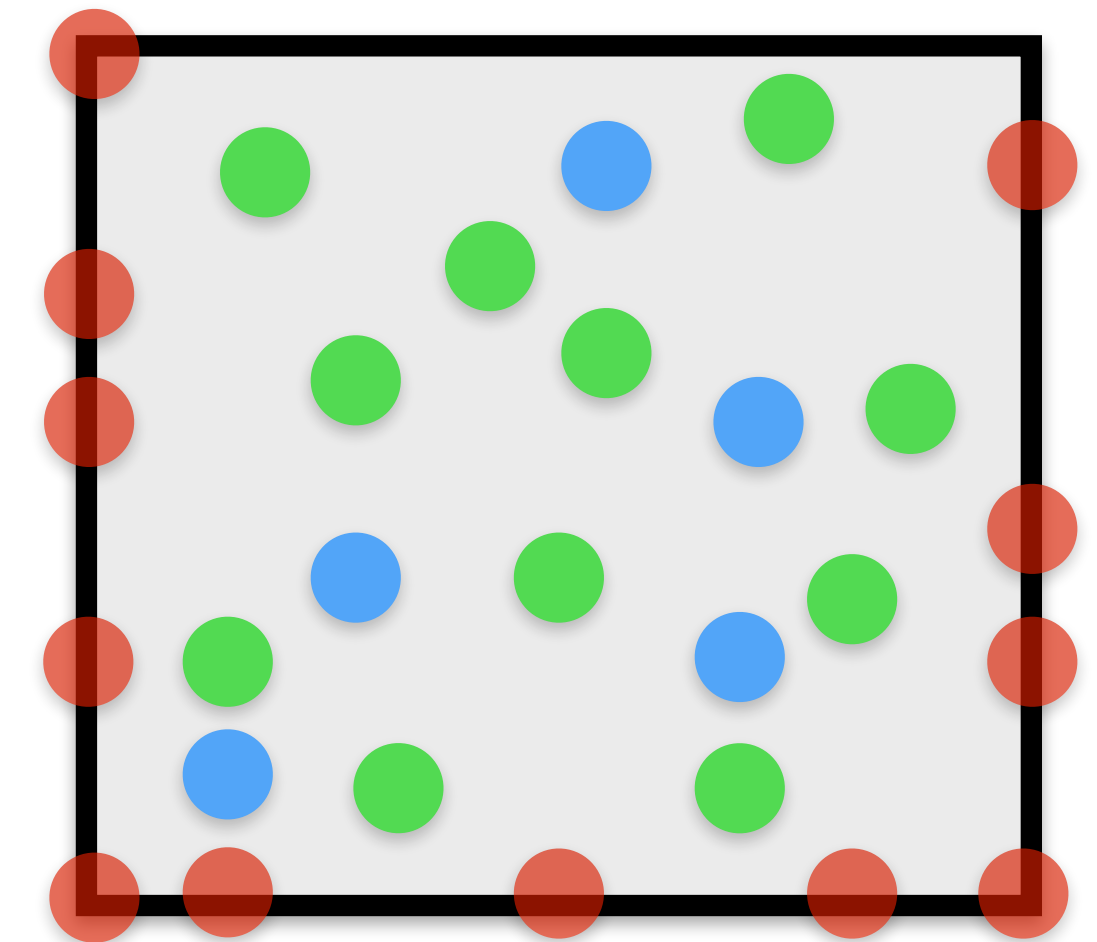
Approximating the PDE residuals

The PINN is trained to **minimize** a **physical loss** function composed of the **PDE residuals** on the domain Ω

$$R(u) = \int_{\Omega} (\mathcal{F}[u](x) - f(x))^2 dx \approx \frac{1}{N} \sum_{i=1}^N (\mathcal{F}[u](x_i) - f(x_i))^2$$

and at the boundary $\partial\Omega$

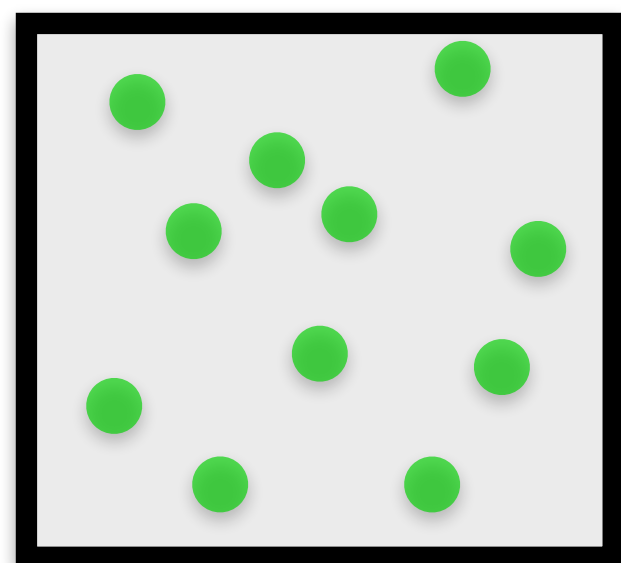
$$R_{bc}(u) = \int_{\partial\Omega} (\mathcal{B}[u](x) - b(x))^2 dx \approx \frac{1}{N_{bc}} \sum_{i=1}^{N_{bc}} (\mathcal{B}[u](x_i) - g(x_i))^2$$



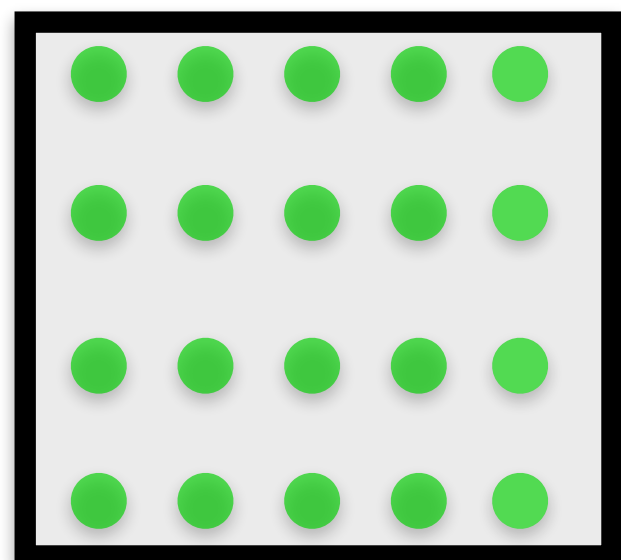
- Supervised points $\{x_i\}_{i=1}^{N_{data}}$
- Collocation points $\{x_i\}_{i=1}^N$
- Boundary points $\{x_i\}_{i=1}^{N_{bc}}$

4 Choosing collocation points

Fixed Methods



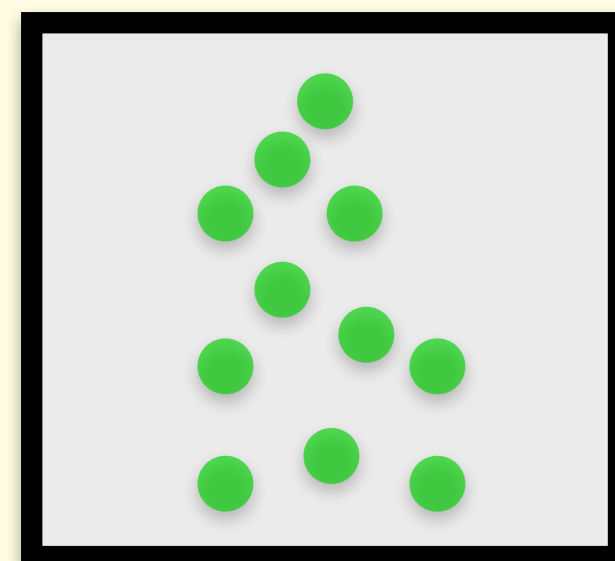
● Random $\{x_i\}_{i=1}^N$



● Uniform $\{x_i\}_{i=1}^N$

Other fixed
sampling methods:
Sobol, LHS, ...

Adaptive Methods

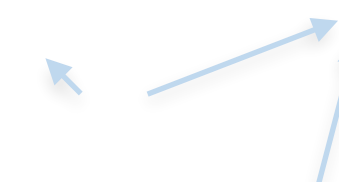


● Adaptive $\{x_i\}_{i=1}^N$

Residual-based

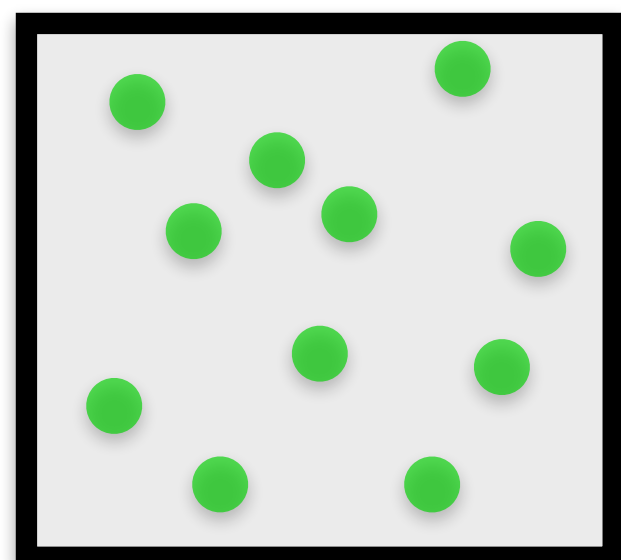
● RAR

Residual-based Adaptive Refinement

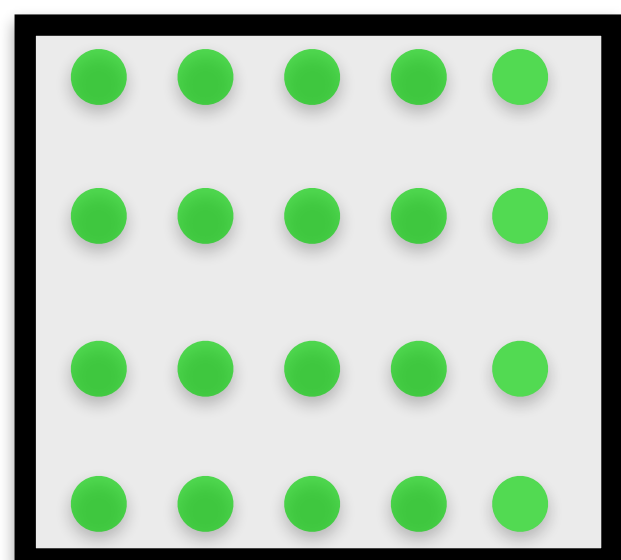


4 Choosing collocation points

Fixed Methods

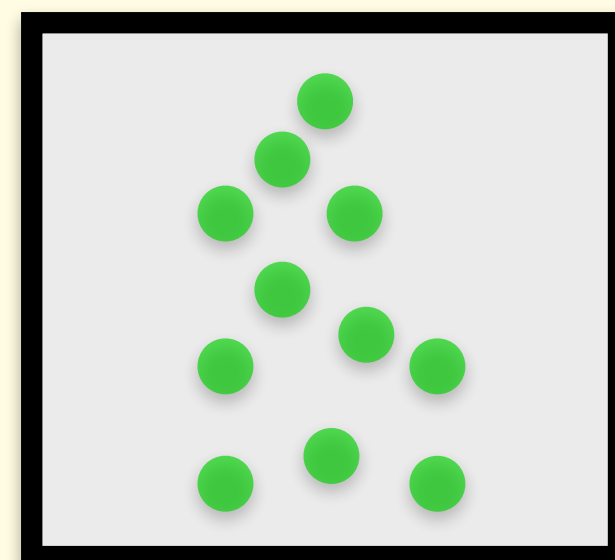


● Random $\{x_i\}_{i=1}^N$



● Uniform $\{x_i\}_{i=1}^N$

Adaptive Methods



● Adaptive $\{x_i\}_{i=1}^N$

Residual-based

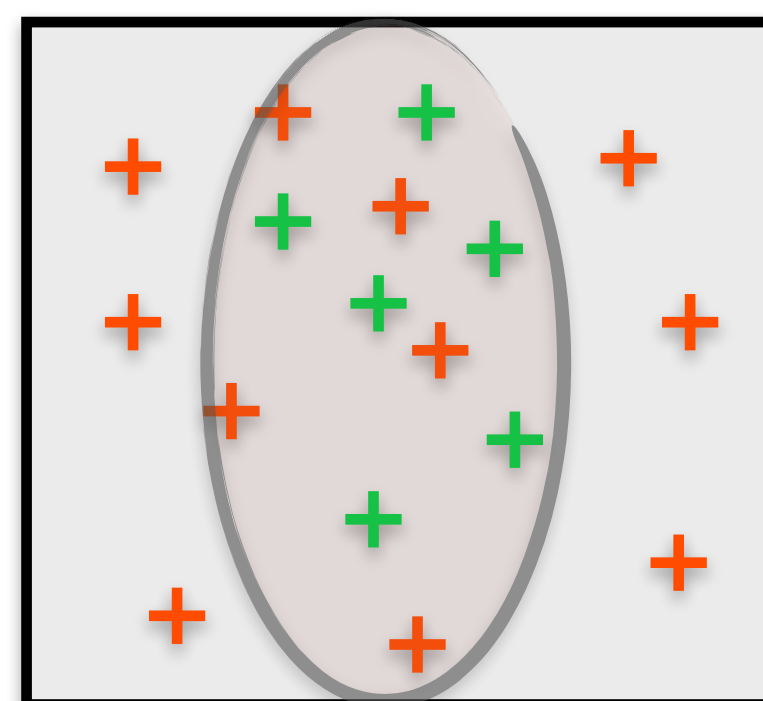
● RAR

Residual-based Adaptive Refinement

Add new points where the PDE residuals are **large**

High computational cost

Uncontrollable number of collocation points



+ **Newly added** points at iteration k

+ Points conserved from iteration $k - 1$

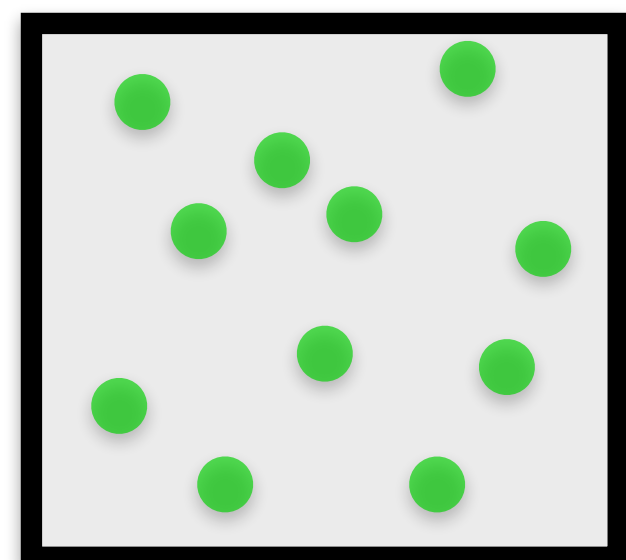
Region with **high PDE residuals**



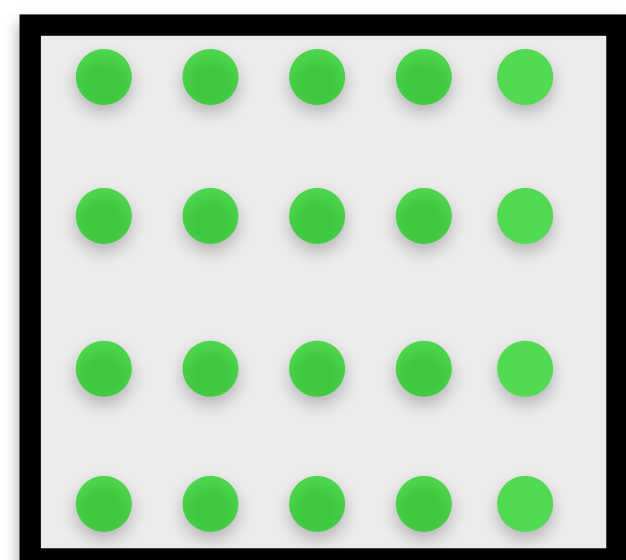
[DeepXDE: A Deep Learning Library for Solving Differential Equations.](#) Lu, Lu and Meng, Xuhui and Mao, Zhiping and Karniadakis, George Em 2021

4 Choosing collocation points

Fixed Methods

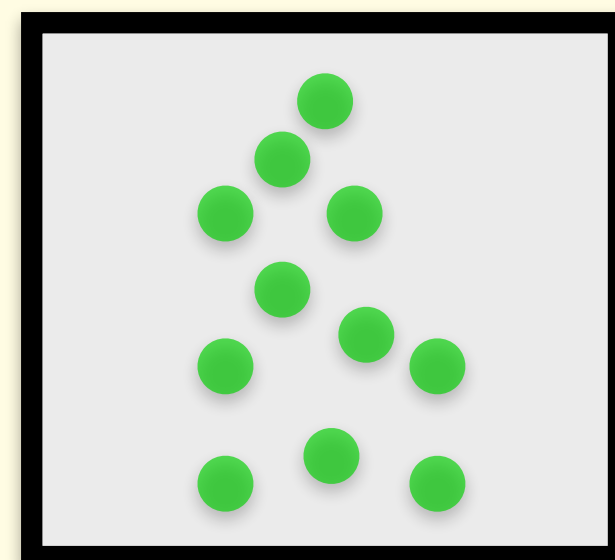


● Random $\{x_i\}_{i=1}^N$



● Uniform $\{x_i\}_{i=1}^N$

Adaptive Methods



● Adaptive $\{x_i\}_{i=1}^N$

Residual-based

● RAD

Residual-based Adaptive Distribution

After k iterations, Resample Randomly all points according to $p(x)$

$$p(x) \propto \frac{\epsilon^k(x)}{\mathbb{E}[\epsilon^k(x)] + c}$$

residual $\epsilon^k(x)$ hyper-parameters

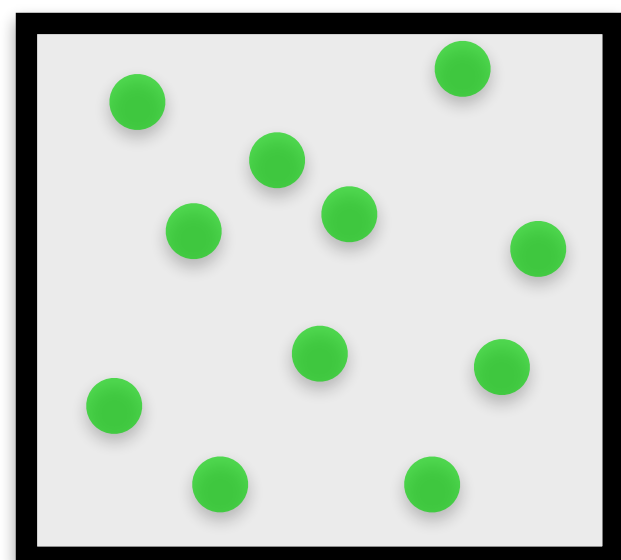
● RARD

Residual-based Adaptive Refinement with Distribution

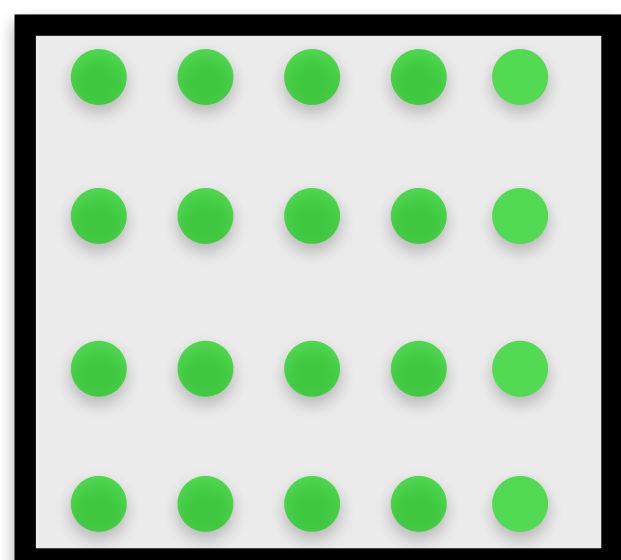
After k iterations, Resample Randomly m more points according to $p(x)$

4 Choosing collocation points

Fixed Methods

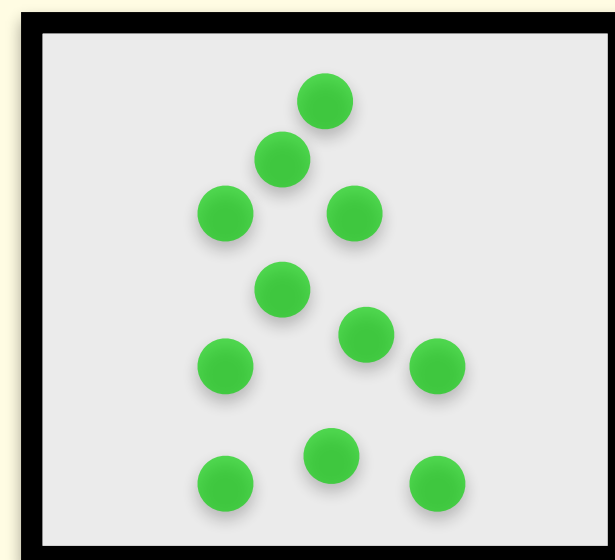


● Random $\{x_i\}_{i=1}^N$



● Uniform $\{x_i\}_{i=1}^N$

Adaptive Methods



● Adaptive $\{x_i\}_{i=1}^N$

Residual-based

● RAD

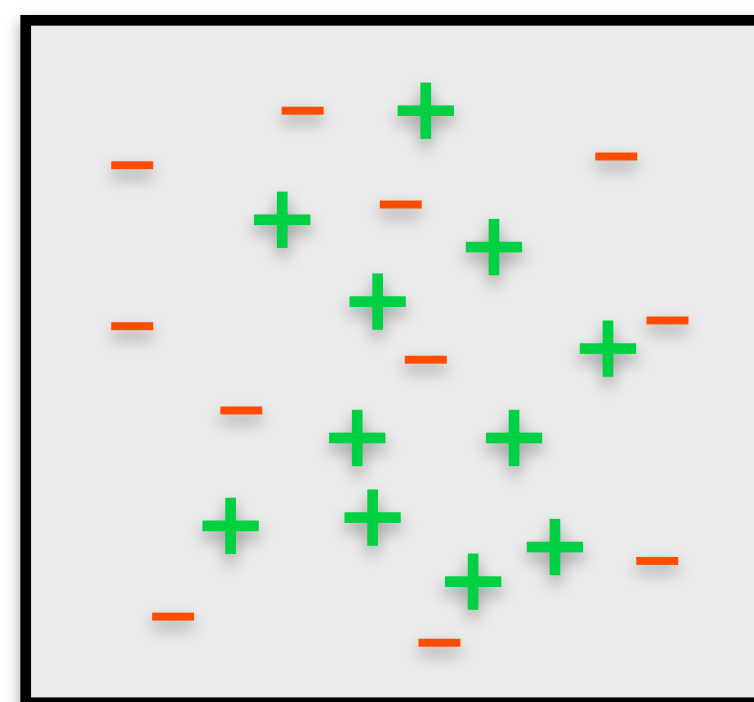
After k iterations, Resample Randomly all points according to $p(x)$

$$p(x) \propto \frac{\epsilon^k(x)}{\mathbb{E}[\epsilon^k(x)] + c}$$

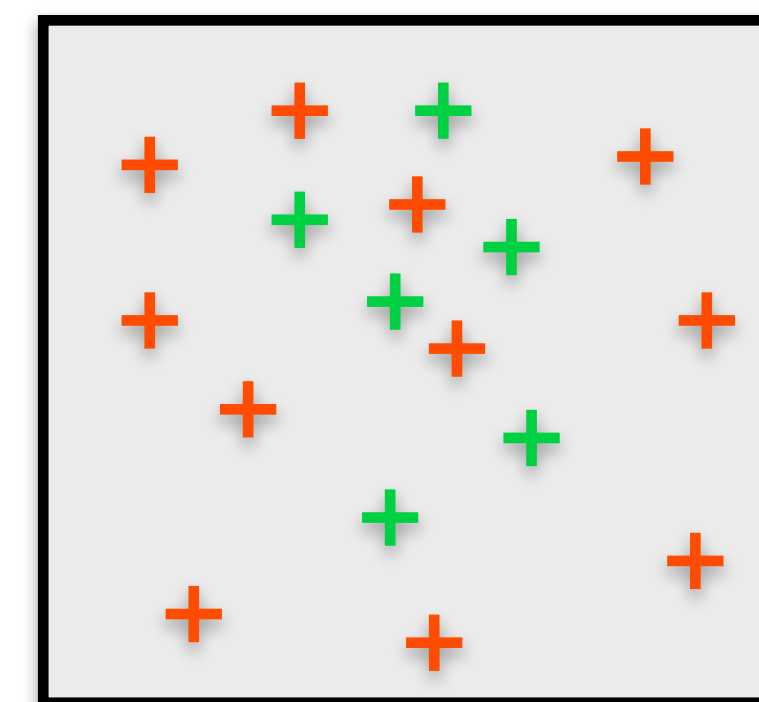
● RARD

After k iterations, Resample Randomly m more points according to $p(x)$

$p(x)$ is **not easy to estimate** correctly in higher dimensions



RAD. Resample all points with $p(x)$



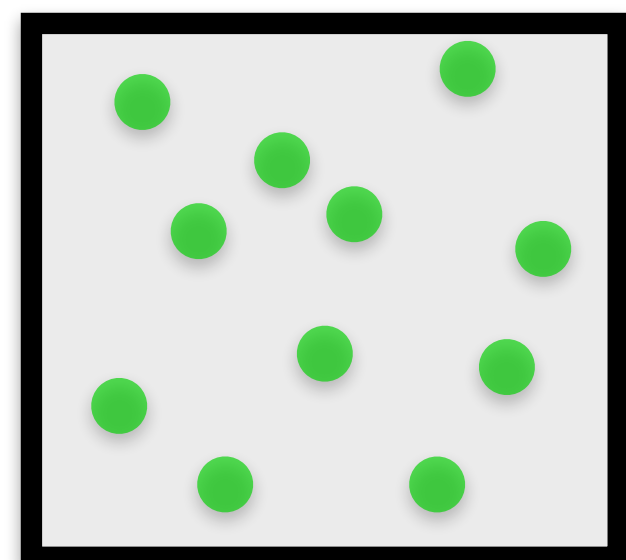
RARD. Adding m points with $p(x)$



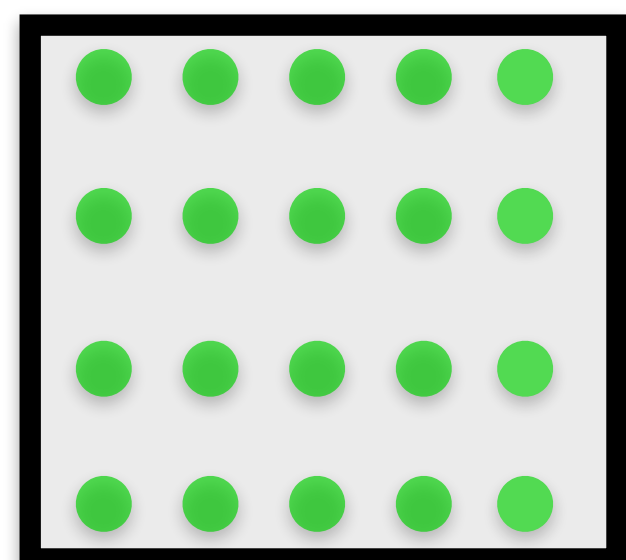
[A comprehensive study of non-adaptive and residual-based adaptive sampling for physics-informed neural networks.](#) **Chenxi Wu, Min Zhu, Qinyang Tan, Yadhu Kartha, Lu Lu 2023**

4 Choosing collocation points

Fixed Methods

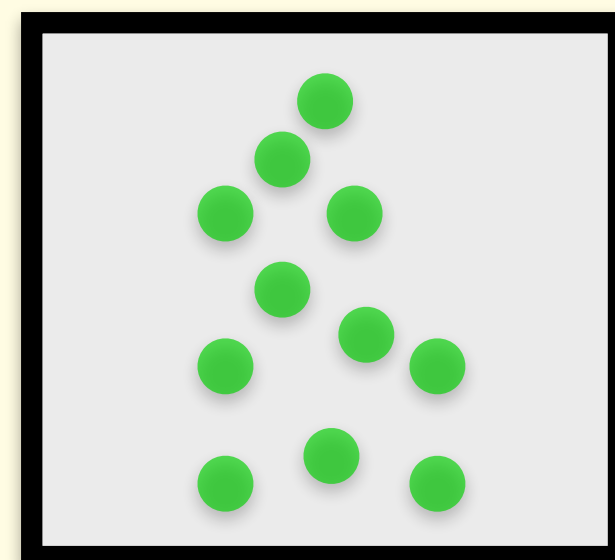


● Random $\{x_i\}_{i=1}^N$



● Uniform $\{x_i\}_{i=1}^N$

Adaptive Methods

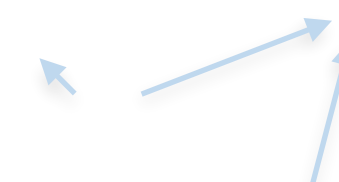


● Adaptive $\{x_i\}_{i=1}^N$

Residual-based

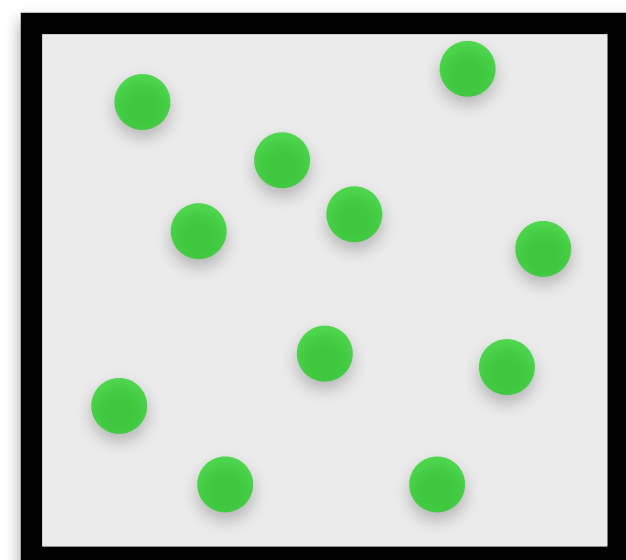
● FBOAL

Fixed-Budget Online Adaptive Learning

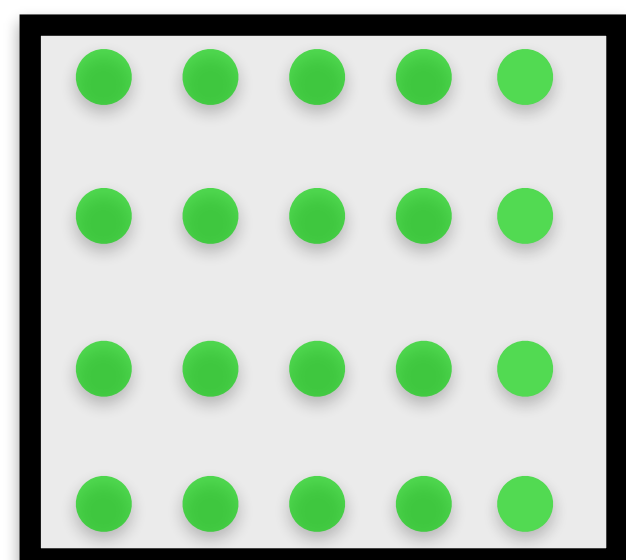


4 Choosing collocation points

Fixed Methods

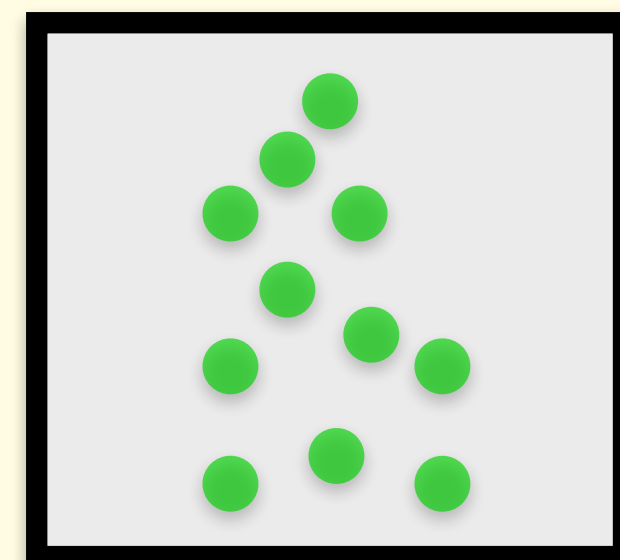


● Random $\{x_i\}_{i=1}^N$



● Uniform $\{x_i\}_{i=1}^N$

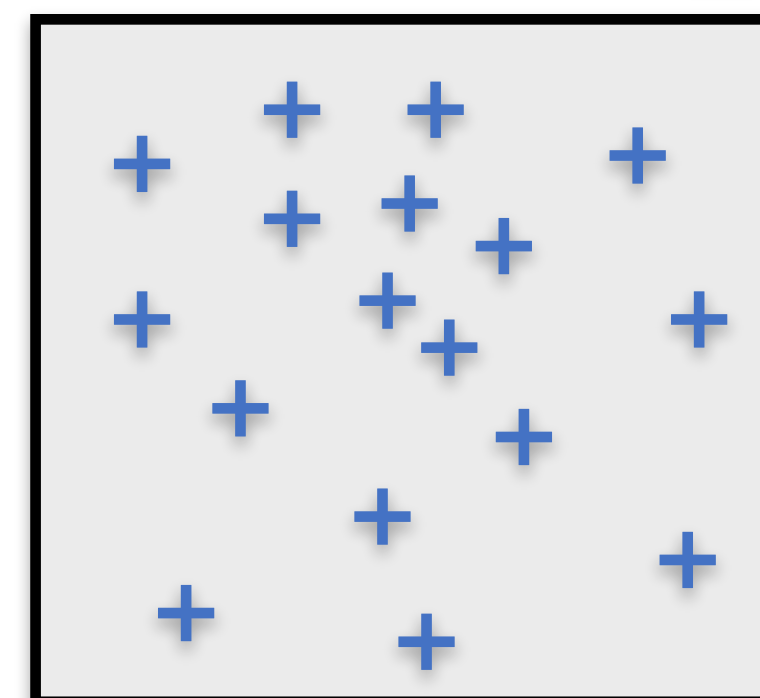
Adaptive Methods



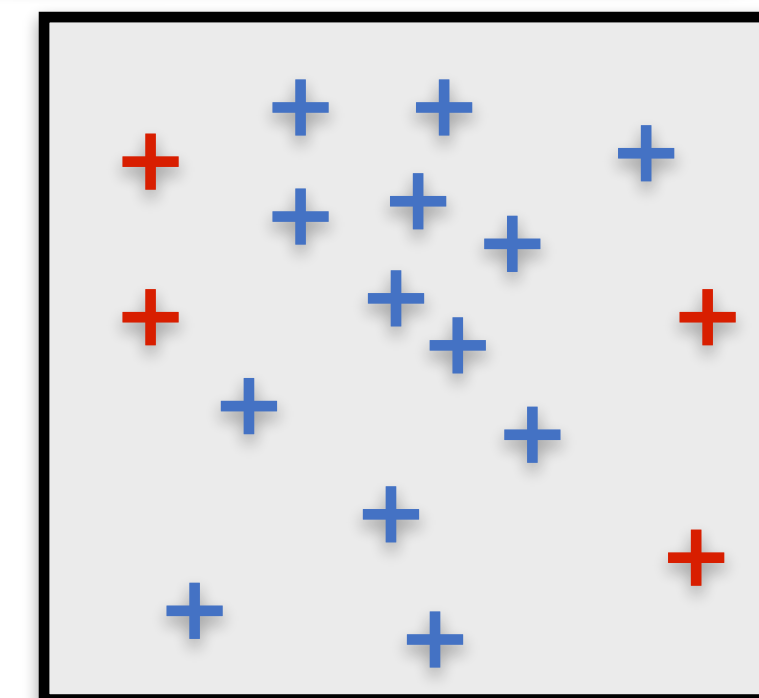
● Adaptive $\{x_i\}_{i=1}^N$

Residual-based

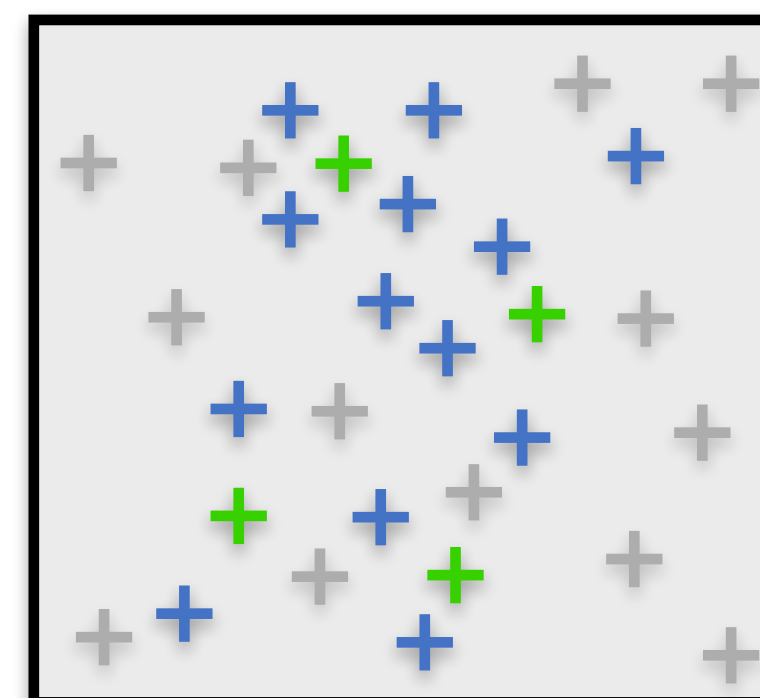
FBOAL



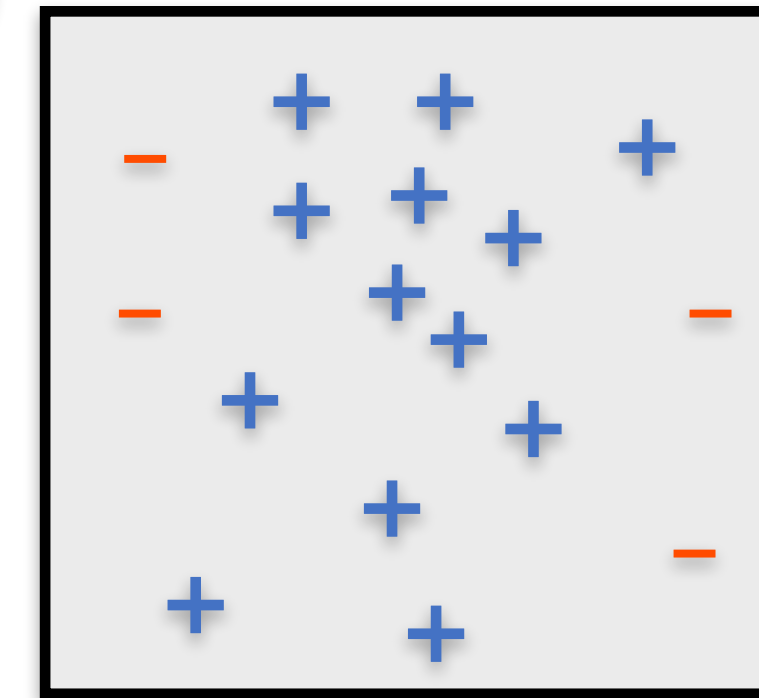
Step 1. Sample randomly in the domain



Step 2. Compute the residuals on the sample
Select m points with the lowest residual values



Step 4. Resample N points in the domain
Add the m points with the highest residual values

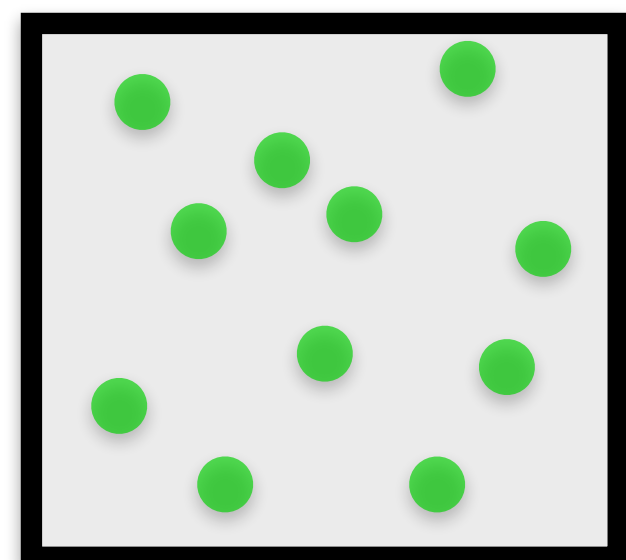


Step 3. Remove the m points with the lowest residual values

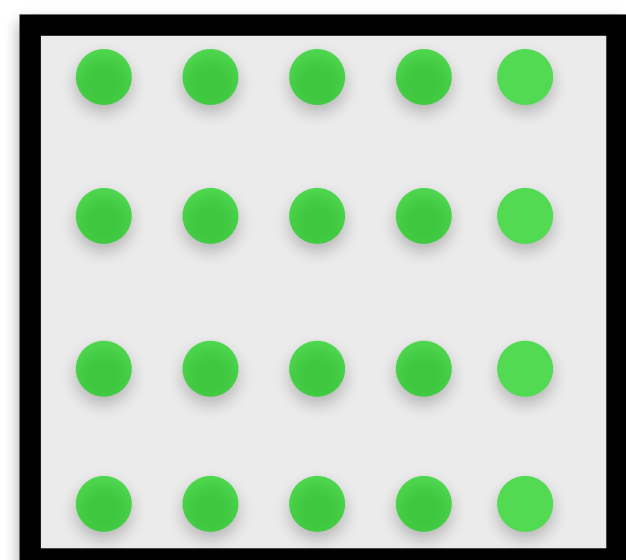


4 Choosing collocation points

Fixed Methods

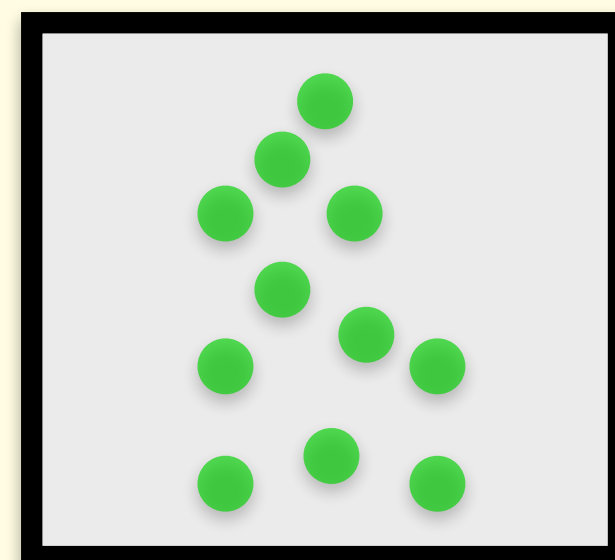


● Random $\{x_i\}_{i=1}^N$



● Uniform $\{x_i\}_{i=1}^N$

Adaptive Methods



● Adaptive $\{x_i\}_{i=1}^N$

Residual-based

● FBOAL

+ *Domain Decomposition*

For now we will consider the Vanilla setting !

Sub-domains = 1



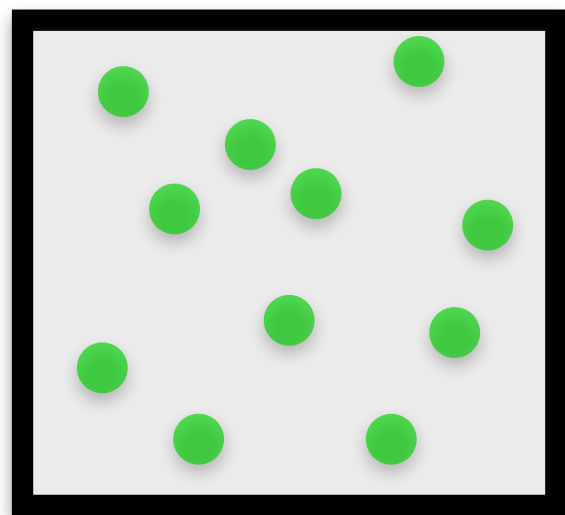
4

Choosing collocation points

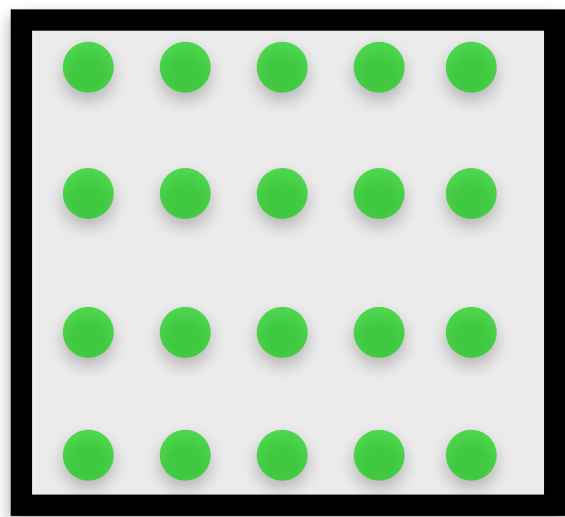
Optimizing the positions of the collocation points during training

Sampling Methods

Fixed Methods

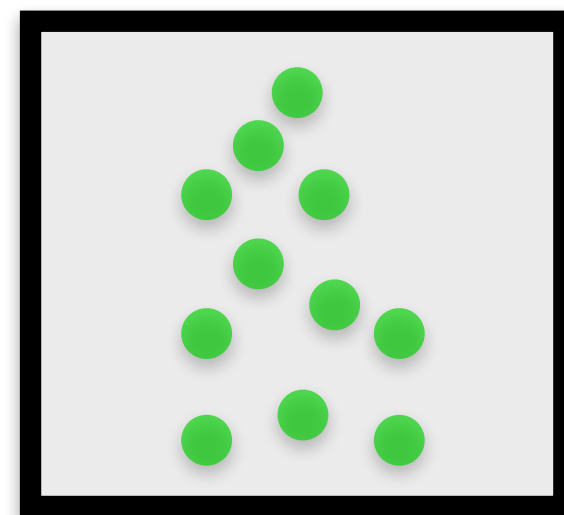


● Random $\{x_i\}_{i=1}^N$



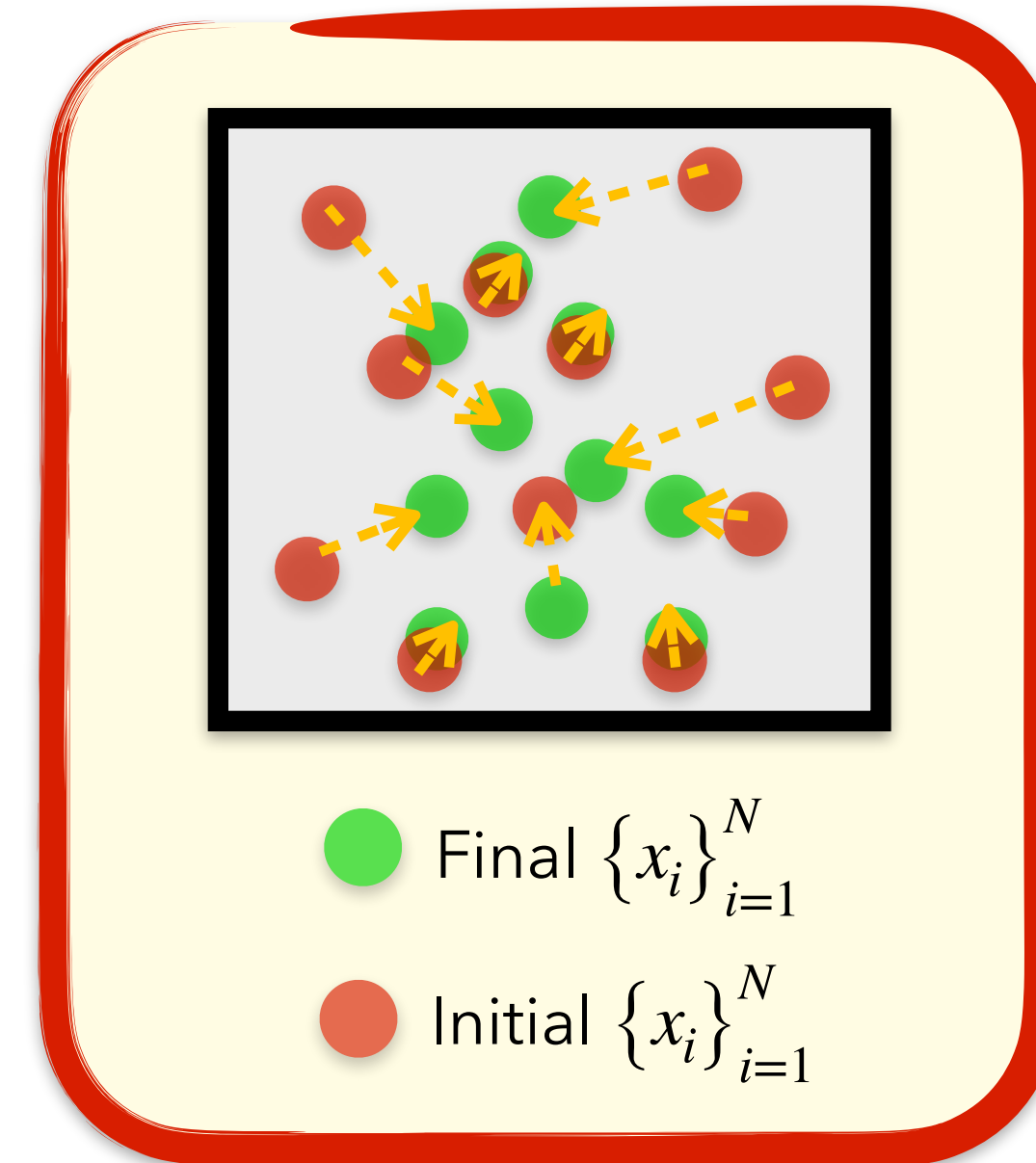
● Uniform $\{x_i\}_{i=1}^N$

Adaptive Methods



● Adaptive $\{x_i\}_{i=1}^N$

Optimization Methods



Instead of **resampling** points after k iterations, we only sample once then **move** the points.

Gradient Ascent

Adam

LBFGS

ADAM + LBFGS

NatGrad



4

Choosing collocation points

Optimizing the positions of the collocation points during training

Algorithm 1 LAS : Langevin dynamics-based Adaptive Sampling

Set network architecture $u : \mathbb{R}^d \rightarrow H$

Initialize neural network parameters θ_0

Initialize collocation set $\mathcal{X}^0$

for $i = 1$ to MaxEpochs **do**

for $k = 1$ to s **do**

 Train model on current collocation set

 Compute gradient $d_i = \nabla_{\mathcal{X}} L_{pde}$

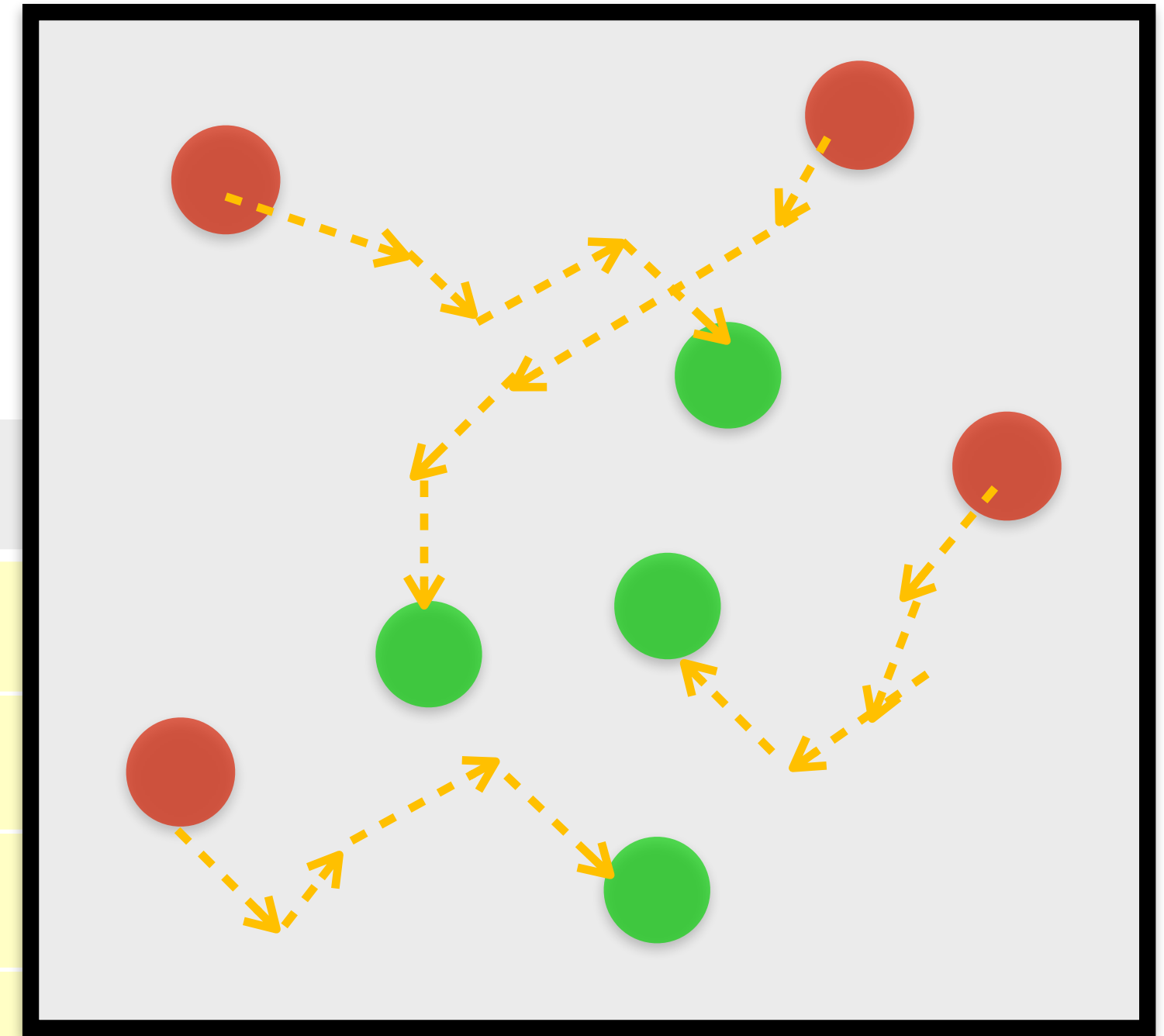
 Update collocation points: $\mathcal{X}^{i+1} = \mathcal{X}^i + \lambda d_i + \text{noise}$

 Clip the updated points to the grid (avoid migrating out of the domain)

End for

End for

Return trained model $\hat{u}_\theta$



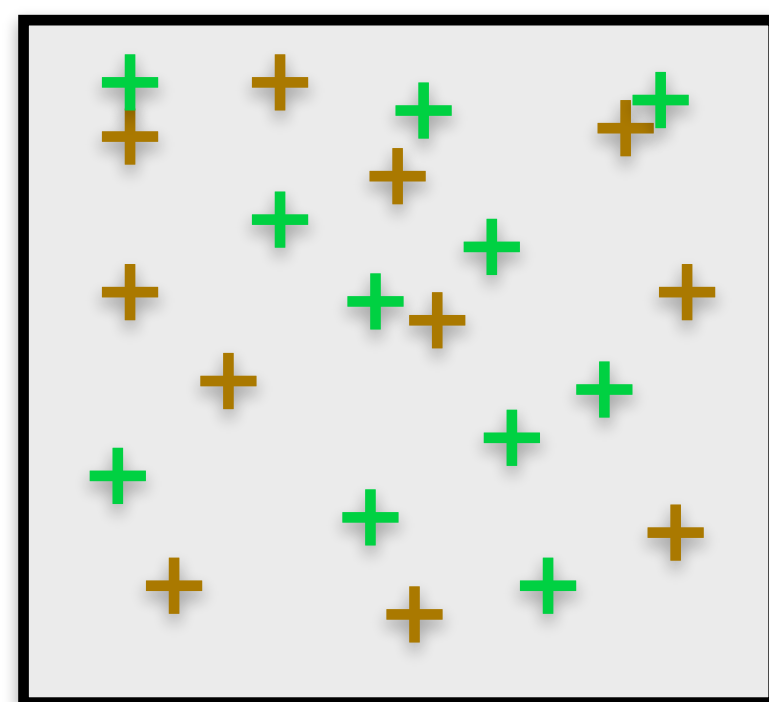
● Final $\{x_i\}_{i=1}^N$
 ● Initial $\{x_i\}_{i=1}^N$



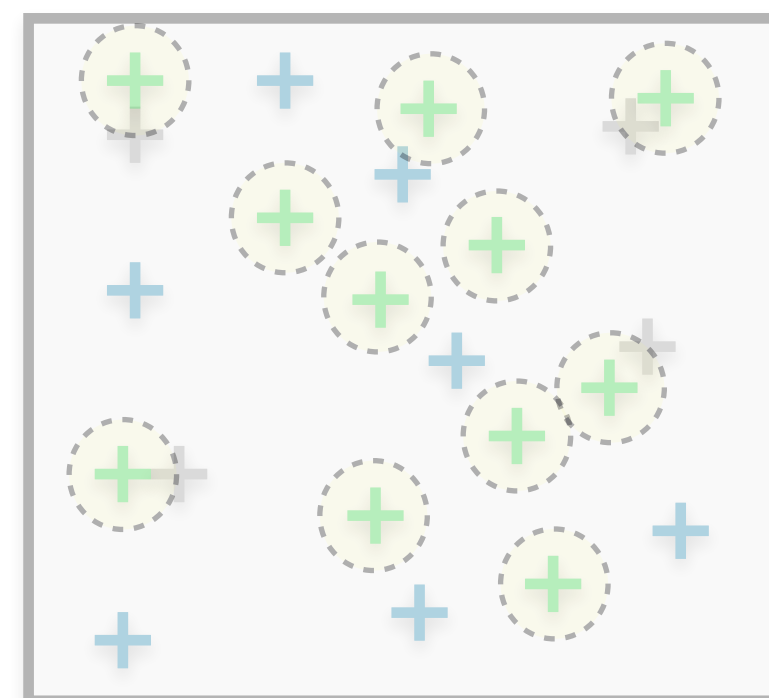
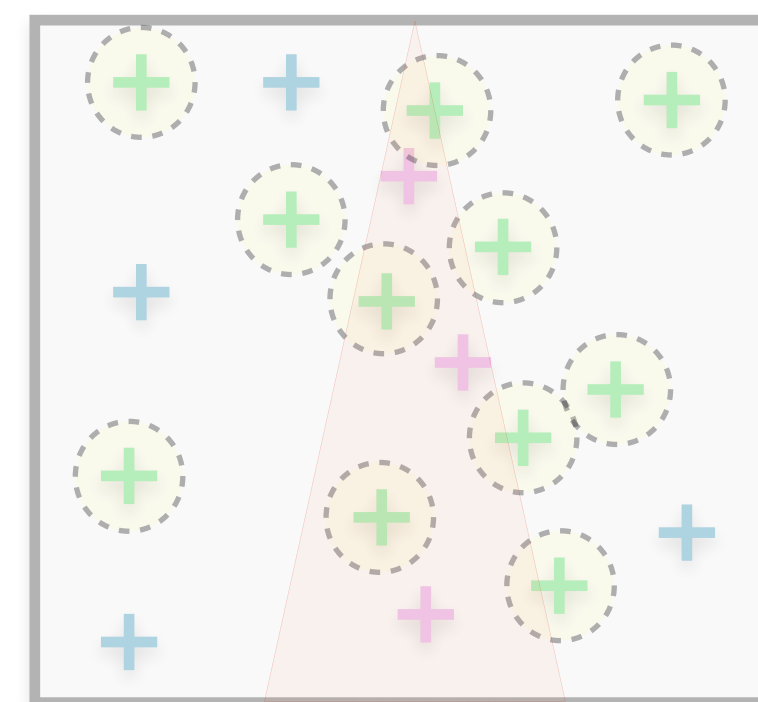
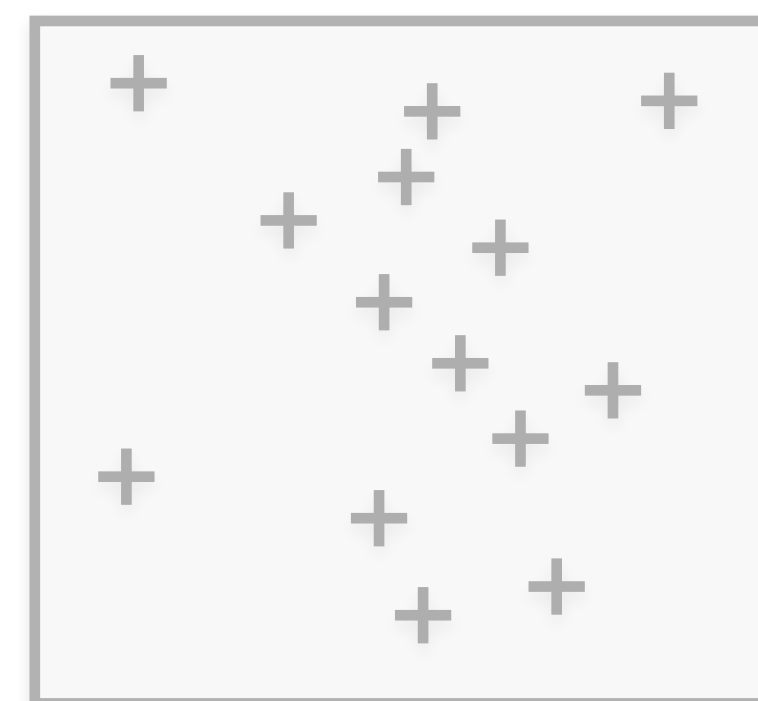
4

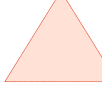
Choosing collocation points

Repulsion Masks - FBOAL +



Sample randomly a second set of points 

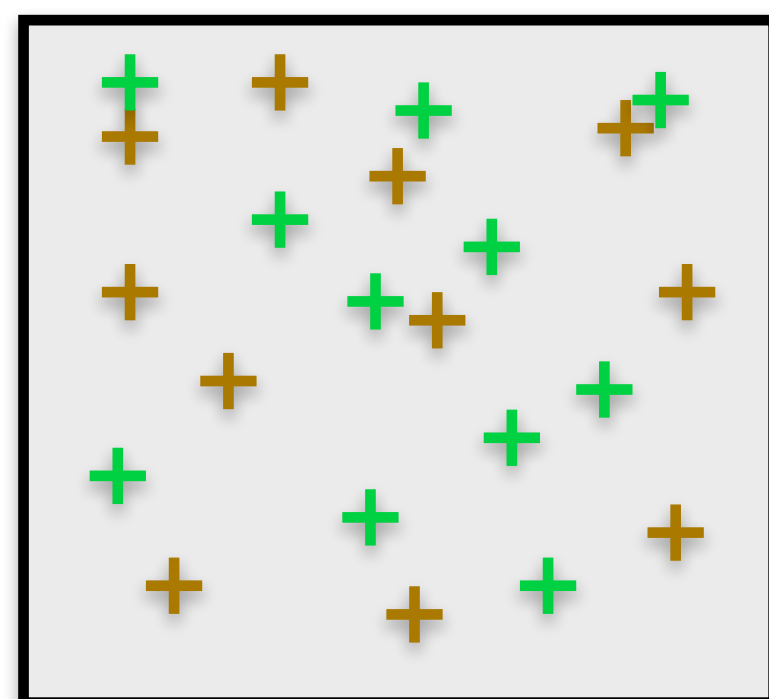


-  **Existing sample** points at iteration k
-  **Random sample** points at iteration $k + 1$
-  **Candidate points**
-  **Rejected points**
-  **Final points**
-  **Repulsion radius**
-  **High residual area**

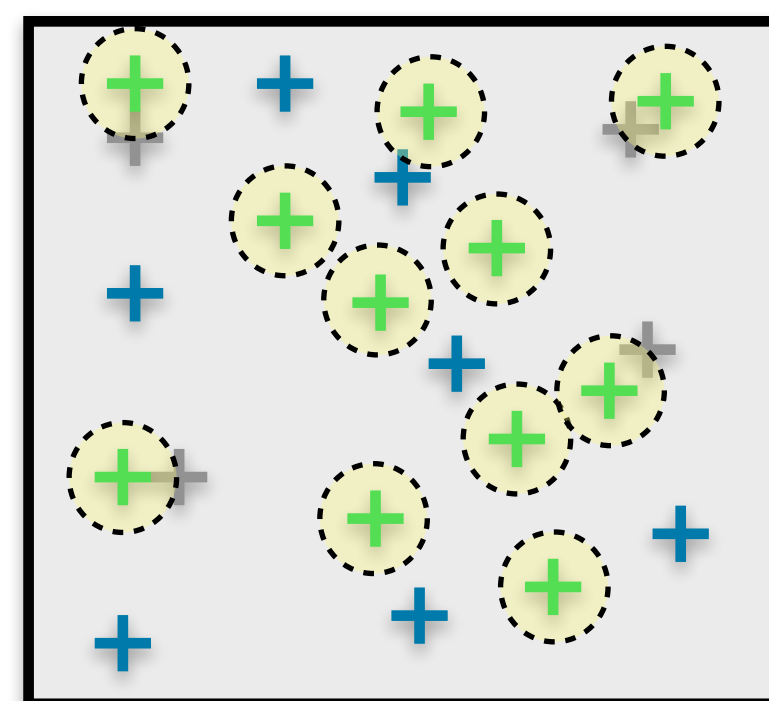
4

Choosing collocation points

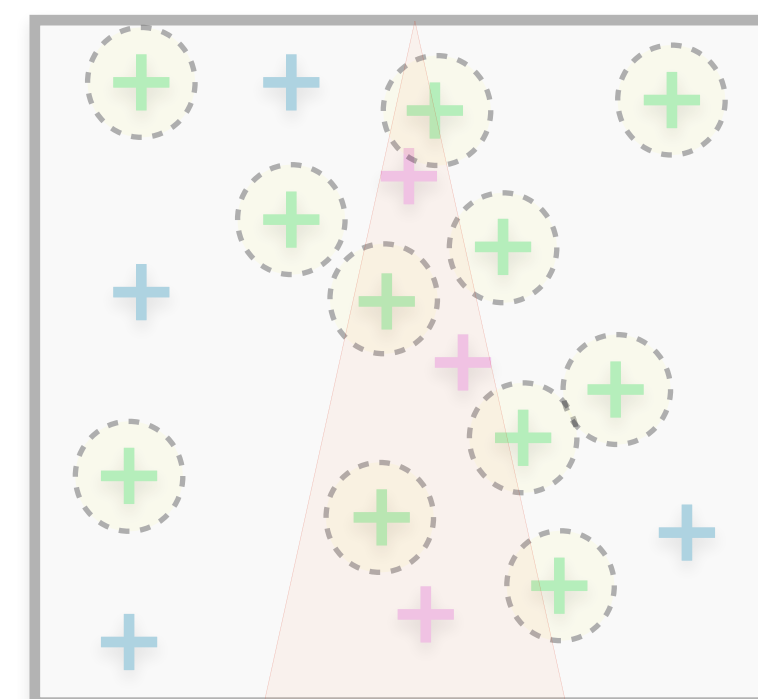
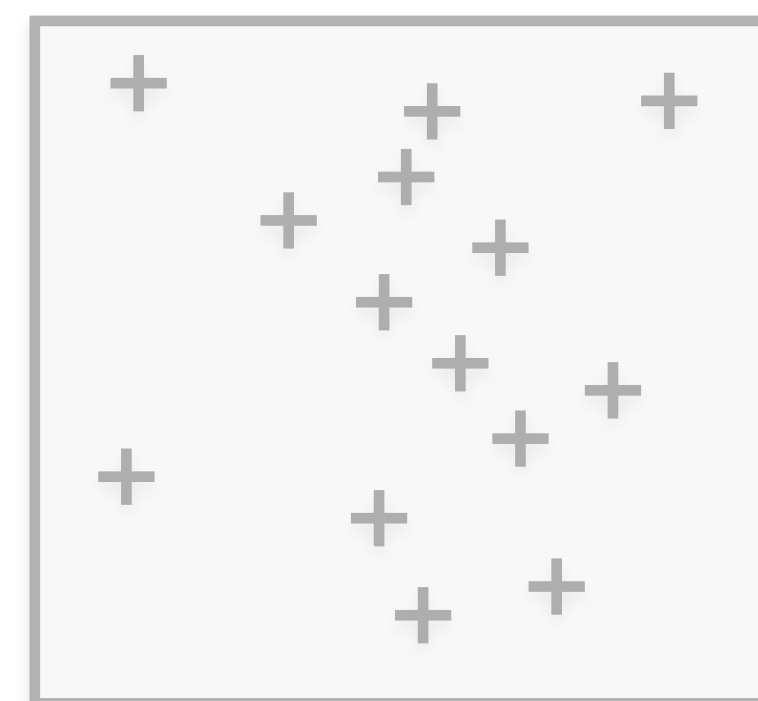
Repulsion Masks - FBOAL +



Sample randomly a second set of points +



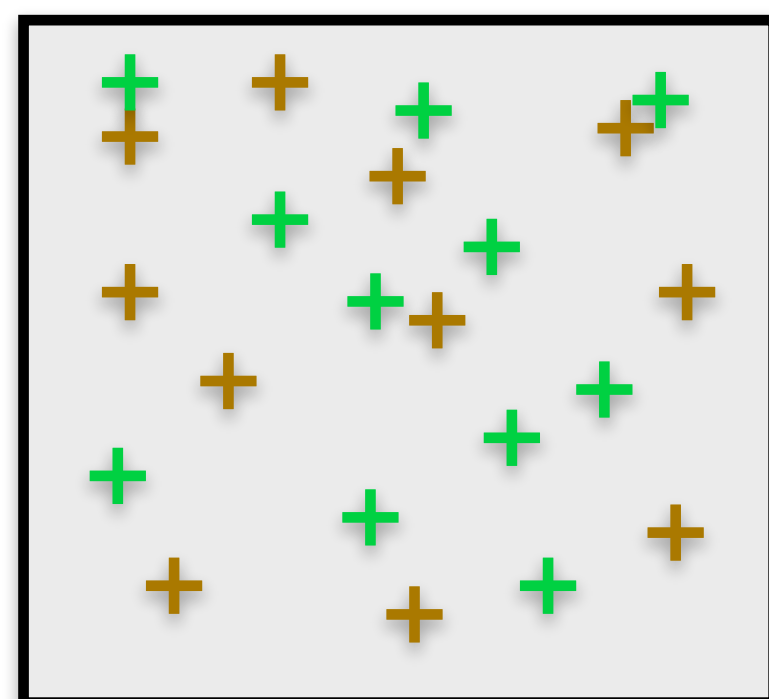
For each collocation point +, reject the points + that fall within the radius



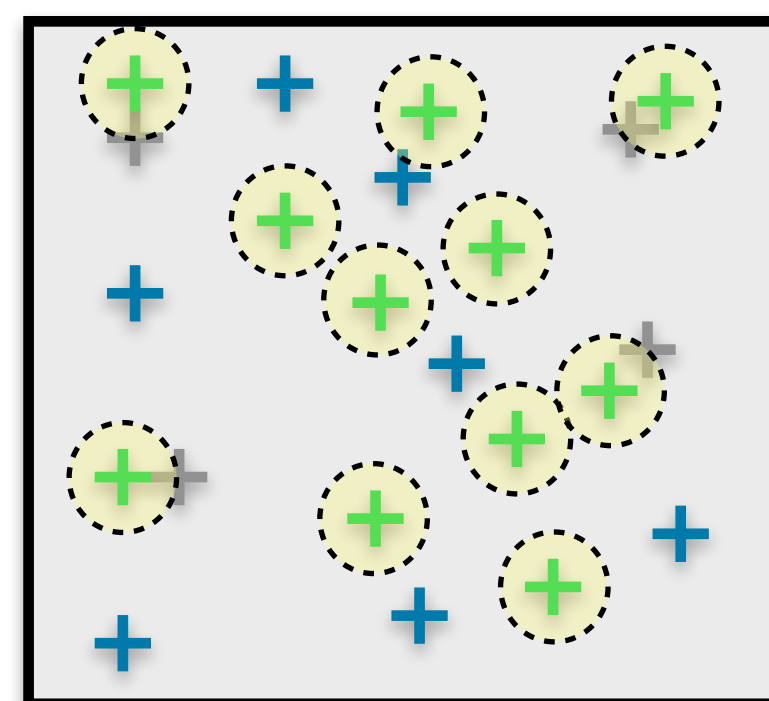
- + Existing sample points at iteration k
- + Random sample points at iteration $k + 1$
- + Candidate points
- + Rejected points
- + Final points
- Repulsion radius
- △ High residual area

4 Choosing collocation points

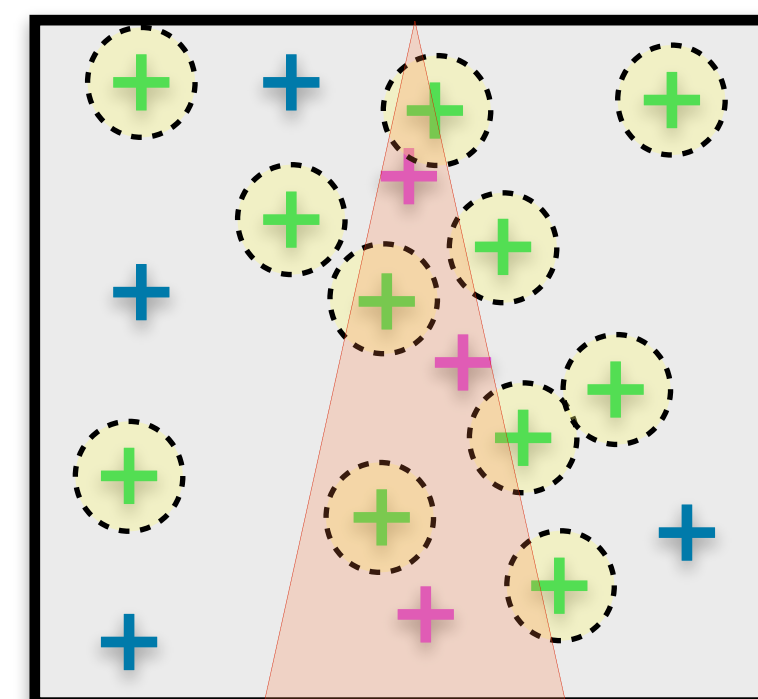
Repulsion Masks - FBOAL +



Sample randomly a second set of points +

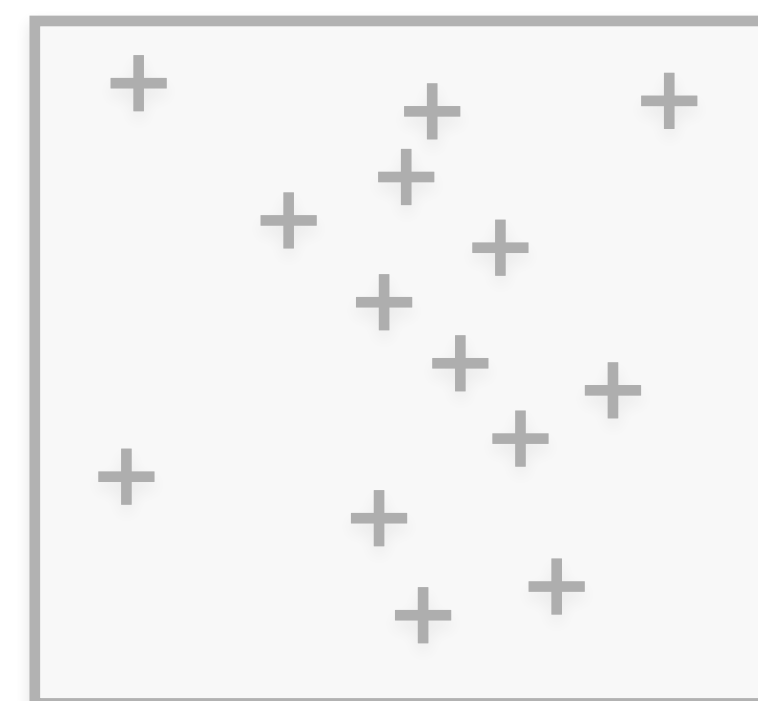


For each collocation point +, reject the points + that fall within the radius



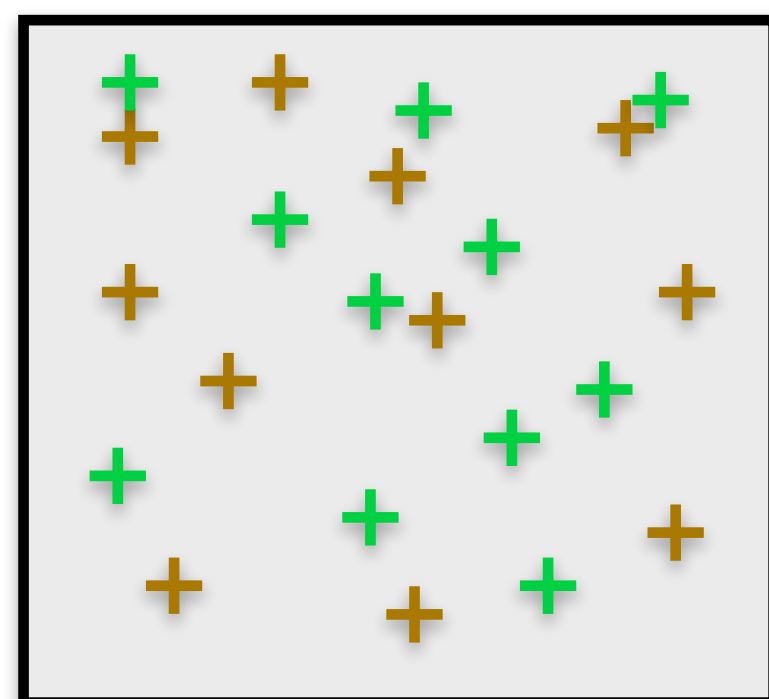
- + Existing sample points at iteration k
- + Random sample points at iteration $k + 1$
- + Candidate points
- + Rejected points
- + Final points
- Repulsion radius
- High residual area

For the remaining candidates + select $m = 3$ points + that give the highest residual values

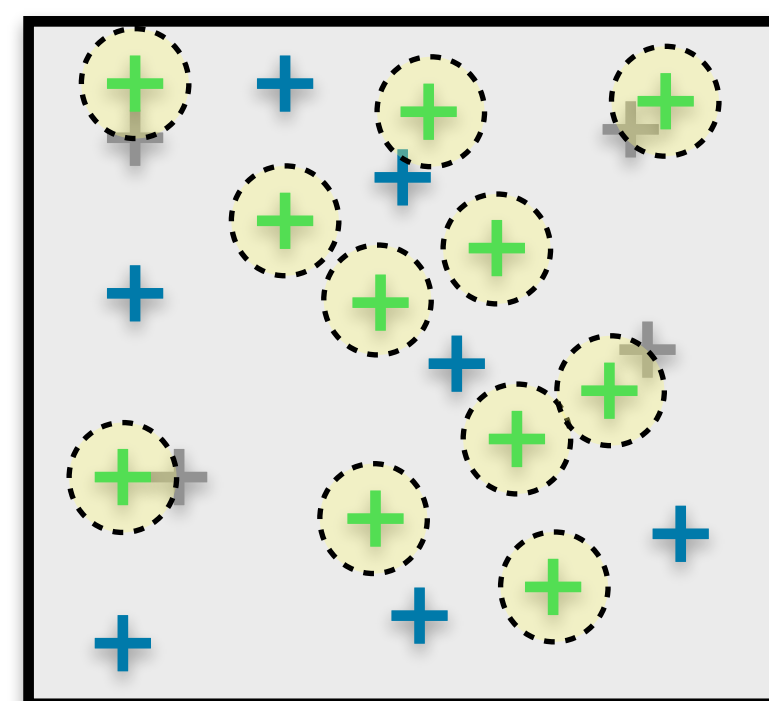


4 Choosing collocation points

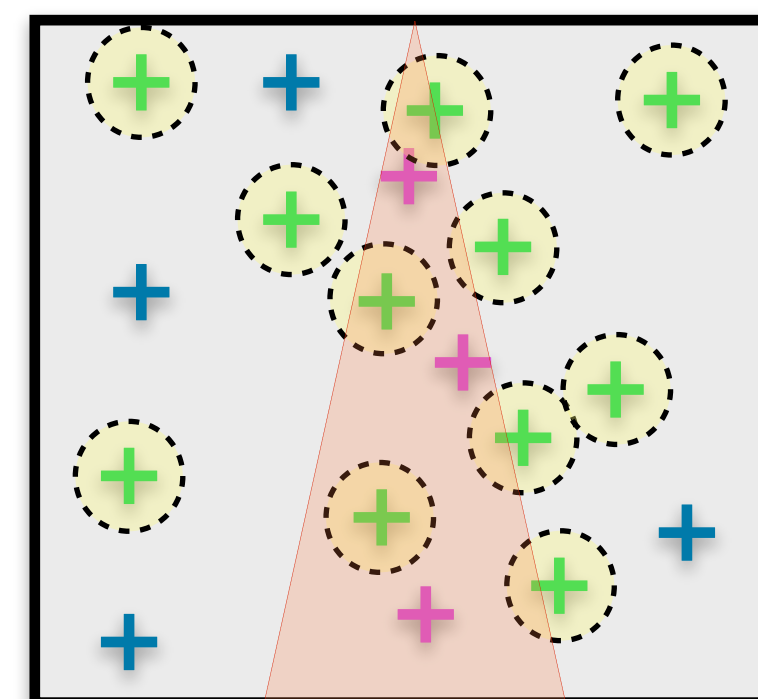
Repulsion Masks - FBOAL +



Sample randomly a second set of points $+$

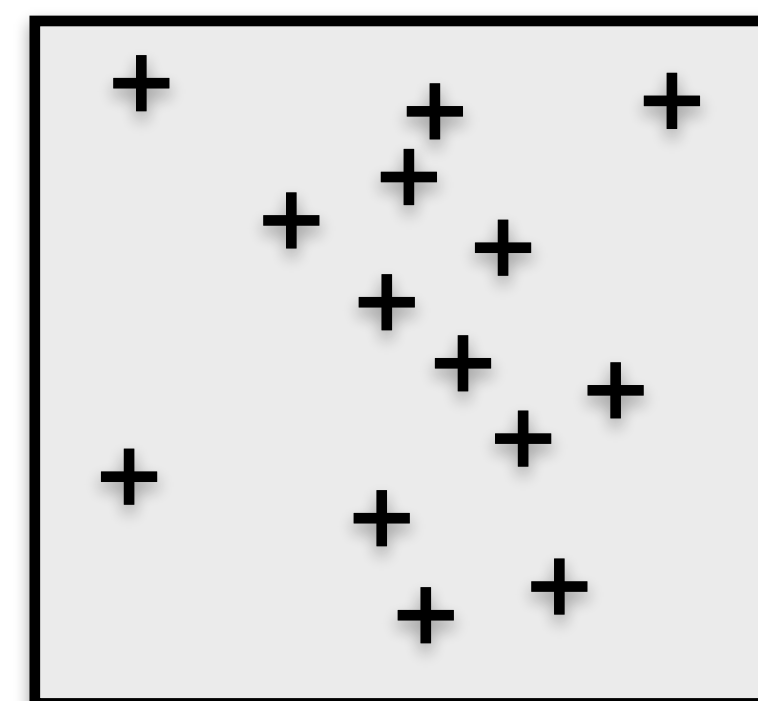


For each collocation point $+$, reject the points $+$ that fall within the radius $\odot$



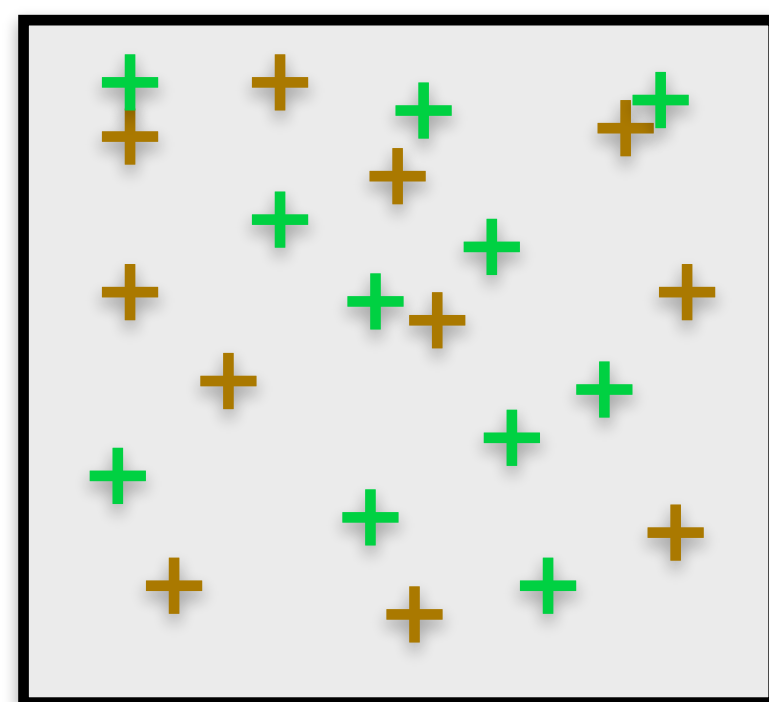
- $+$ Existing sample points at iteration k
- $+$ Random sample points at iteration $k+1$
- $+$ Candidate points
- $+$ Rejected points
- $+$ Final points
- $\odot$ Repulsion radius
- $\triangle$ High residual area

For the remaining candidates $+$ select $m = 3$ points $+$ that give the highest residual values $\triangle$



4 Choosing collocation points

Adaptive m - FBOAL +

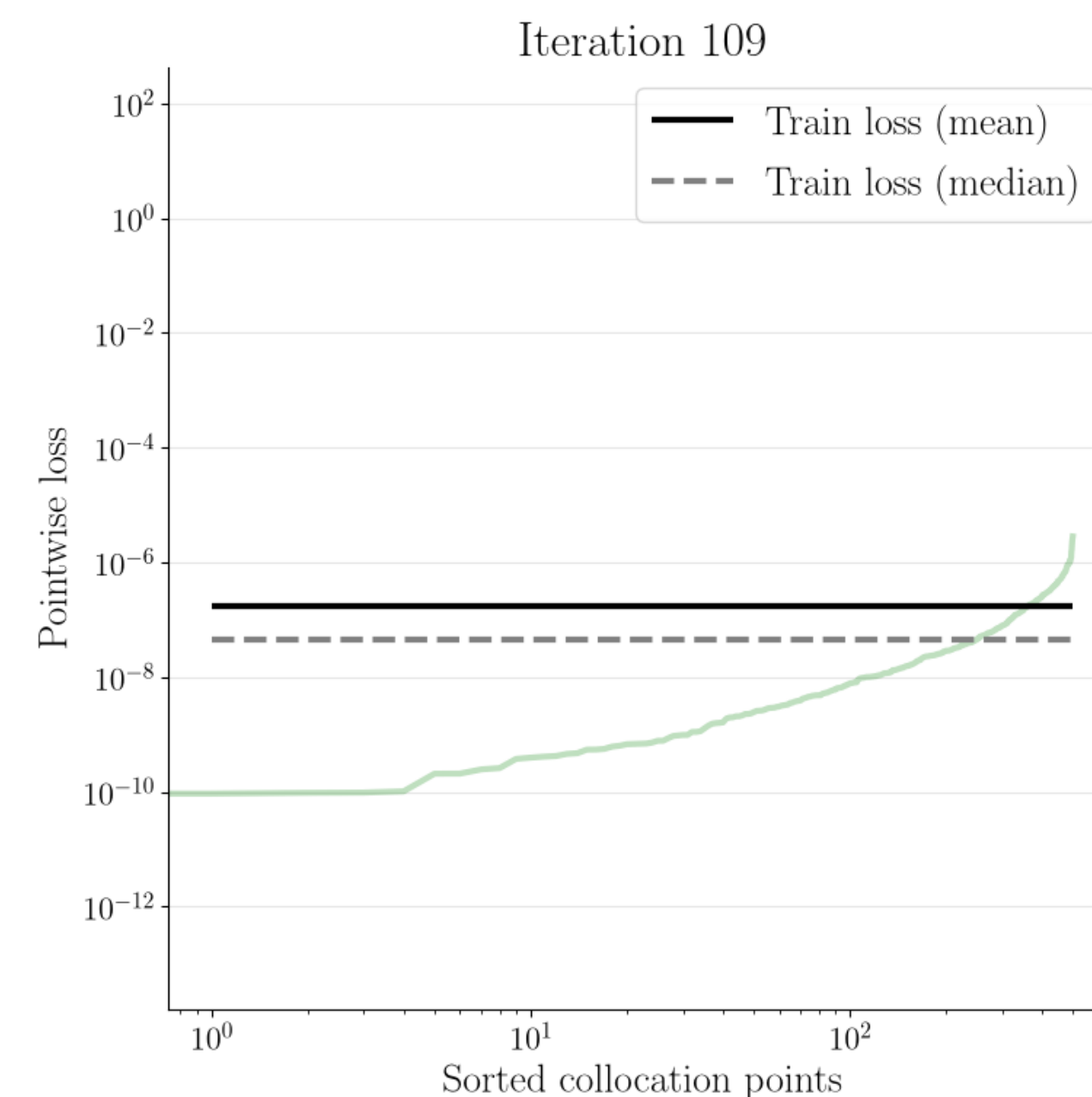


+ Randomly Sampled points

The value m of points to be replaced will vary during training as $m(\theta)$

$$m(\theta) = \left\lceil \sqrt{\#\{i \in \{1, \dots, n\} : \tilde{\ell}_{(i)} > \ell_{(i)}\}} \right\rceil$$

The worst-performing $m(\theta)$ points are then replaced by the highest-error $m(\theta)$ points.



5

Boundary Conditions

Weakly-enforced

Through the collocation points

Accuracy depends on the sample size

$$L_{pinn}(\theta) = L_{data}(\theta) + L_{pde}(\theta) + L_{bc}(\theta)$$

↪ Enforced through the **loss** on **sampled points** at the **boundaries**

5

Boundary Conditions

Strongly-enforced

Through the network

Exact

Output of the Neural Network
(estimates **PDE solution+ boundary conditions**)

$$\hat{u}_{\theta}(x) = \phi(x) * NN(x; \theta) + b(x)$$

↪ Boundary function

5

Boundary Conditions

Strongly-enforced

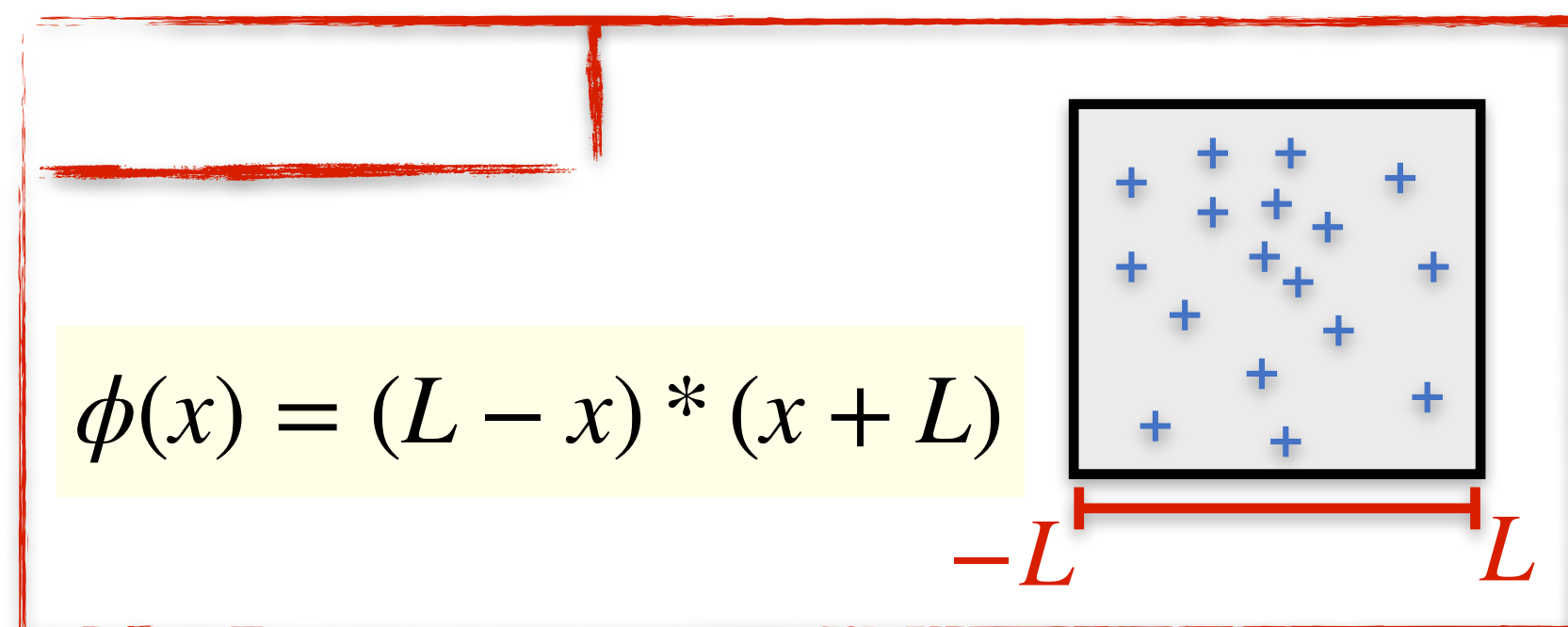
Through the network

Exact

Output of the Neural Network
(estimates **PDE solution+ boundary conditions**)

$$\hat{u}_{\theta}(x) = \phi(x) * NN(x; \theta) + b(x)$$

↪ Boundary function



6

Viscous Burgers 1D



Use a PINN to approximate the solution $\hat{u}(x, t; \theta) \approx u(x, t)$

$$\frac{\partial u}{\partial t} + u \frac{\partial u}{\partial x} - \nu \frac{\partial^2 u}{\partial x^2} = 0$$

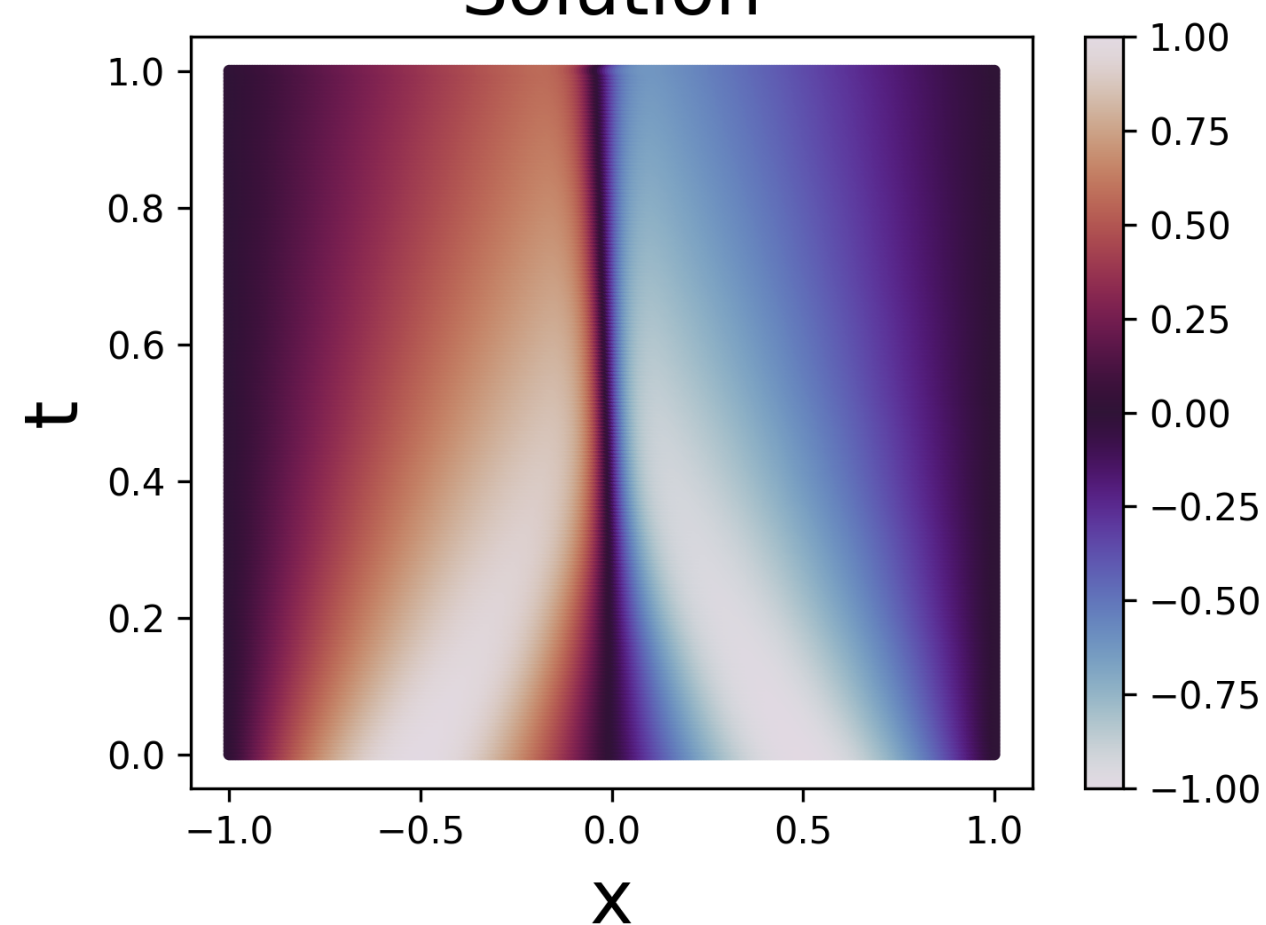
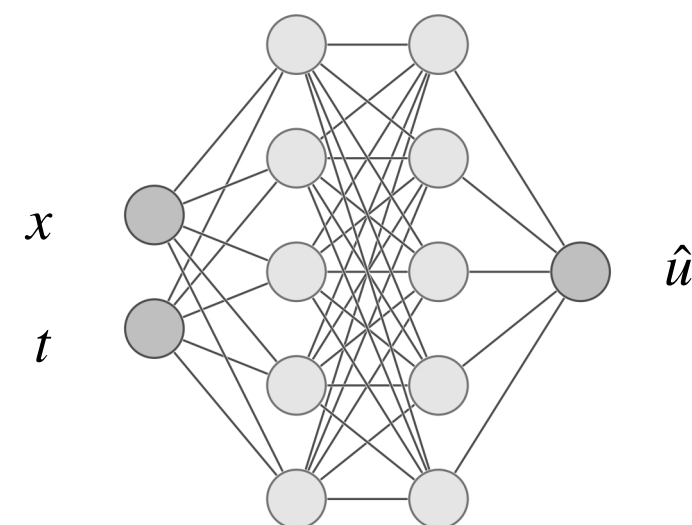
Initial and boundary conditions

$$u(x, 0) = -\sin(\pi x)$$

$$u(-1, t) = u(1, t) = 0$$

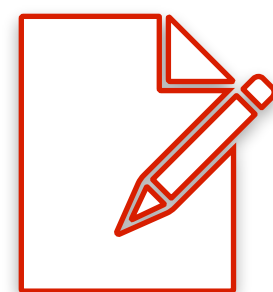
u = fluid velocity
 ν = constant viscosity (0.025)

Solution

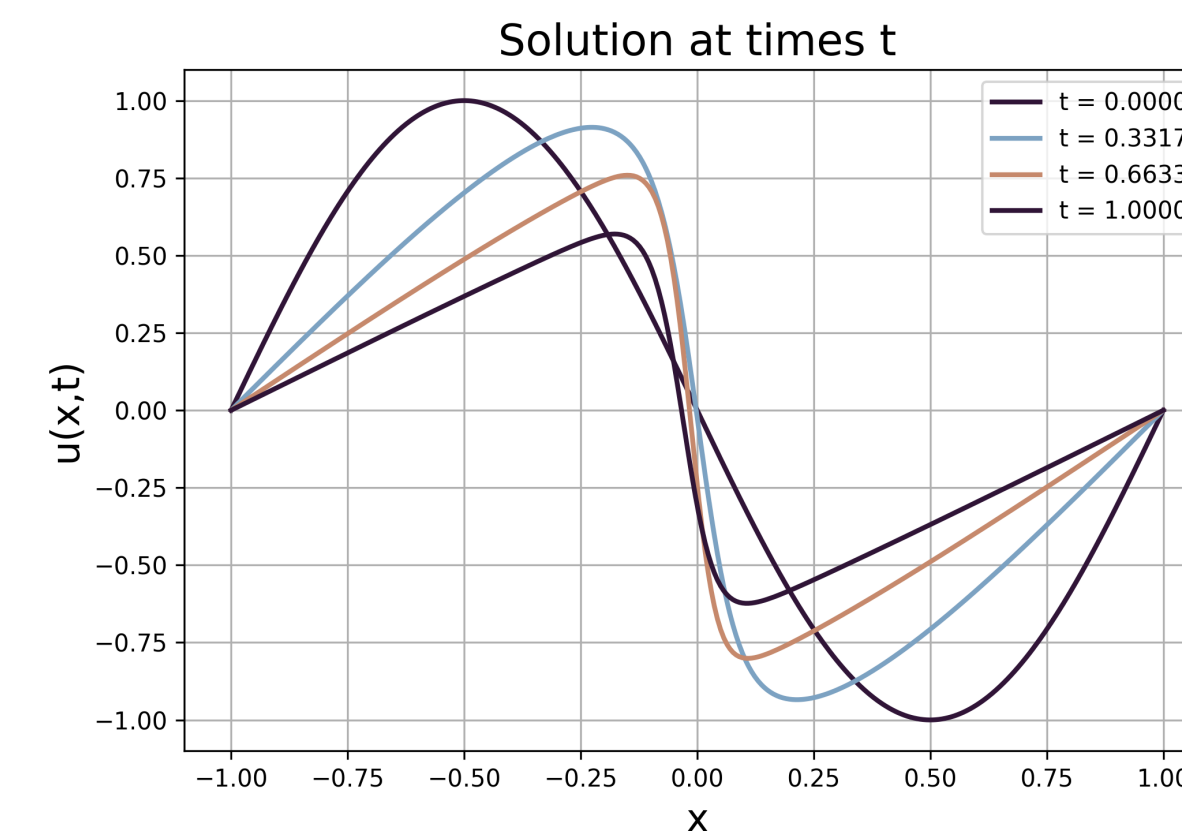


Train the PINN using the loss function

$$L(\theta) = \frac{1}{N} \sum_{i=1}^N \left[\frac{\partial \hat{u}_{\theta}(x_i, t_i)}{\partial t} + \hat{u}_{\theta}(x_i, t_i) \frac{\partial \hat{u}_{\theta}(x_i, t_i)}{\partial x} - \nu \frac{\partial^2 \hat{u}_{\theta}(x_i, t_i)}{\partial x^2} \right]^2 + \frac{1}{N_{ic}} \sum_{i=1}^{N_{ic}} [\hat{u}(x_i, 0) + \sin(\pi x_i)]^2 + \frac{1}{N_{bc}} \sum_{i=1}^{N_{bc}} |\hat{u}(x_{bc}, t_i)|^2$$



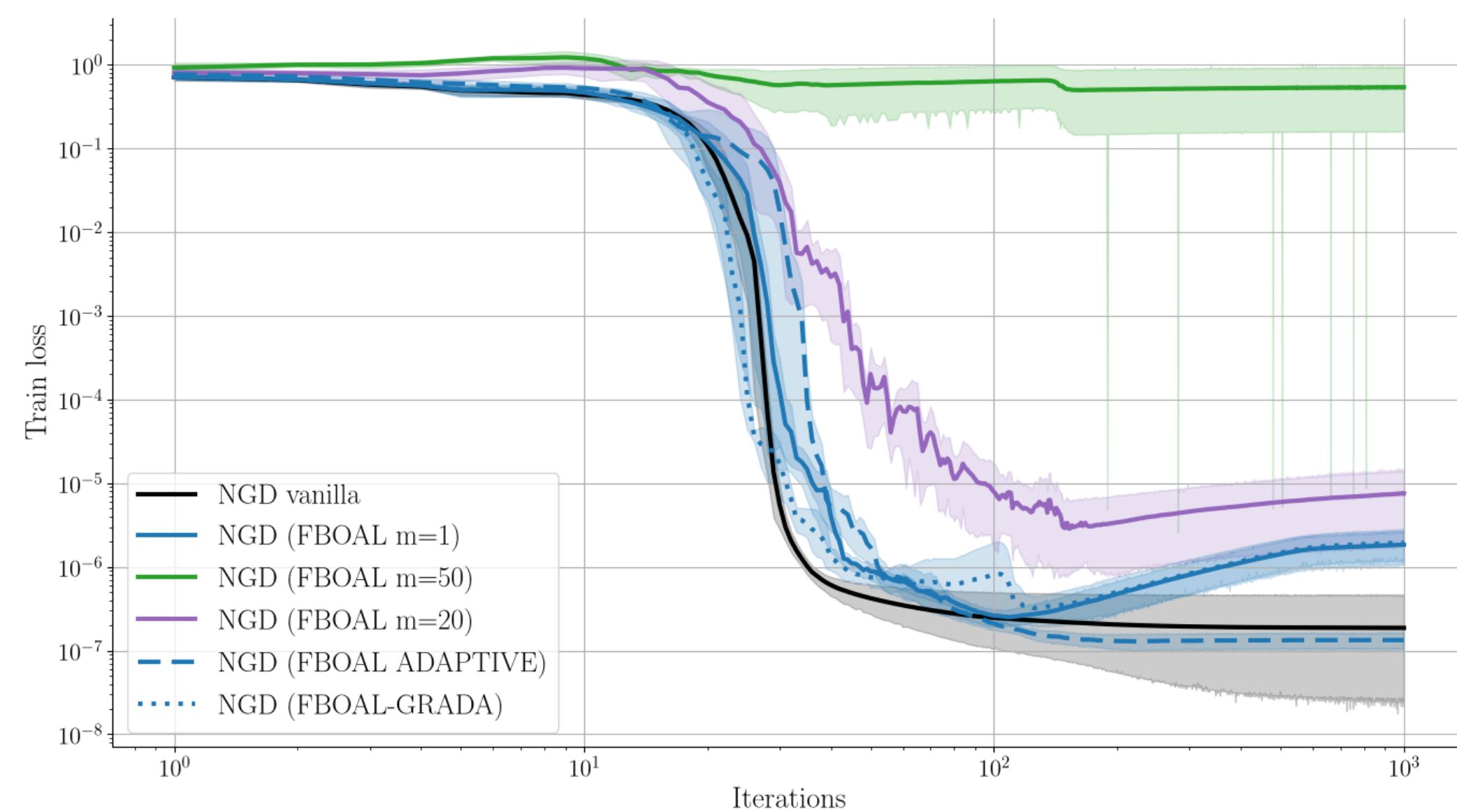
The MLP treats the time variable t as a space variable !



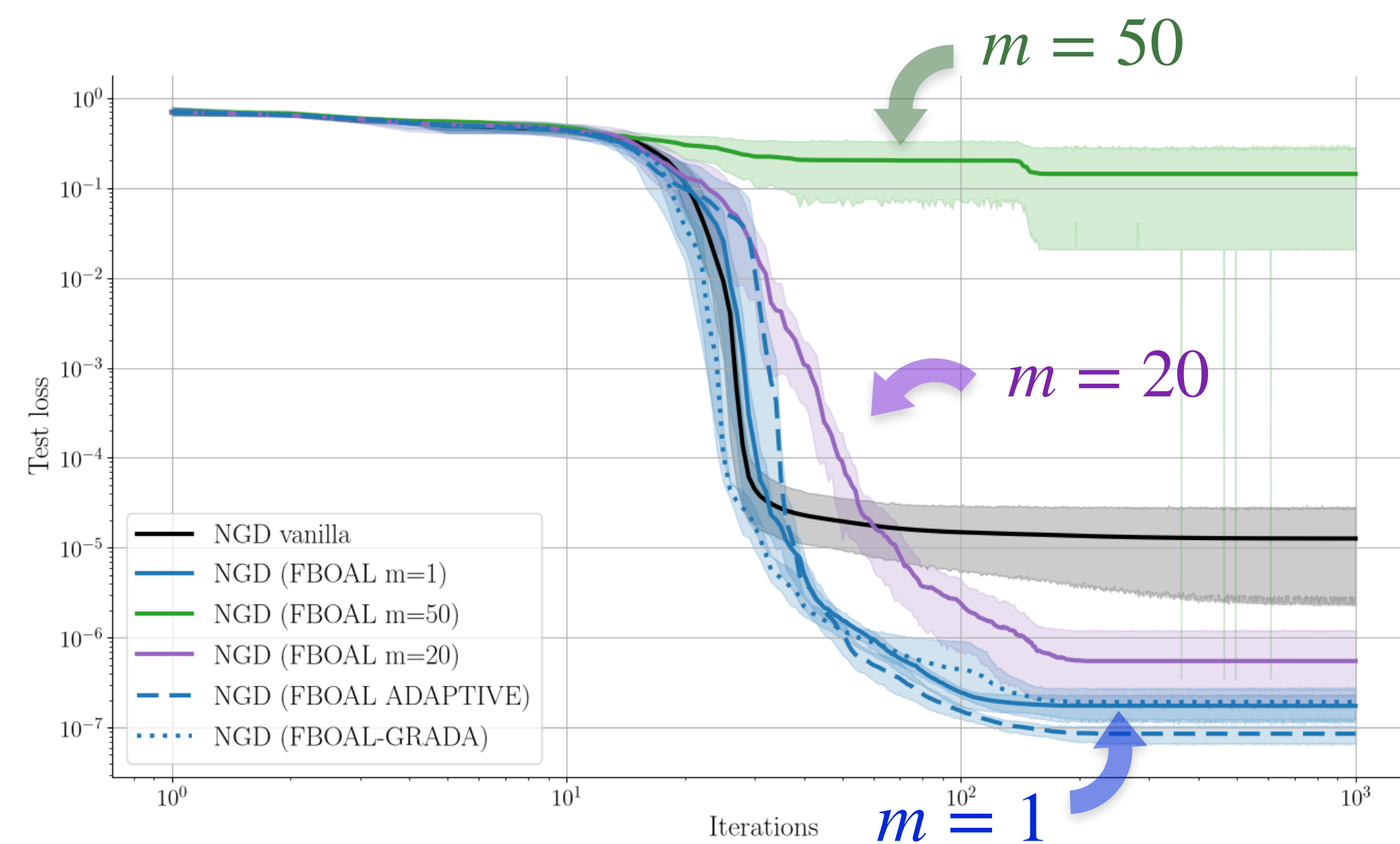
6

Viscous Burgers 1D

What happens when we move many points at once with NGD ?



Loss over the Collocation Points during training

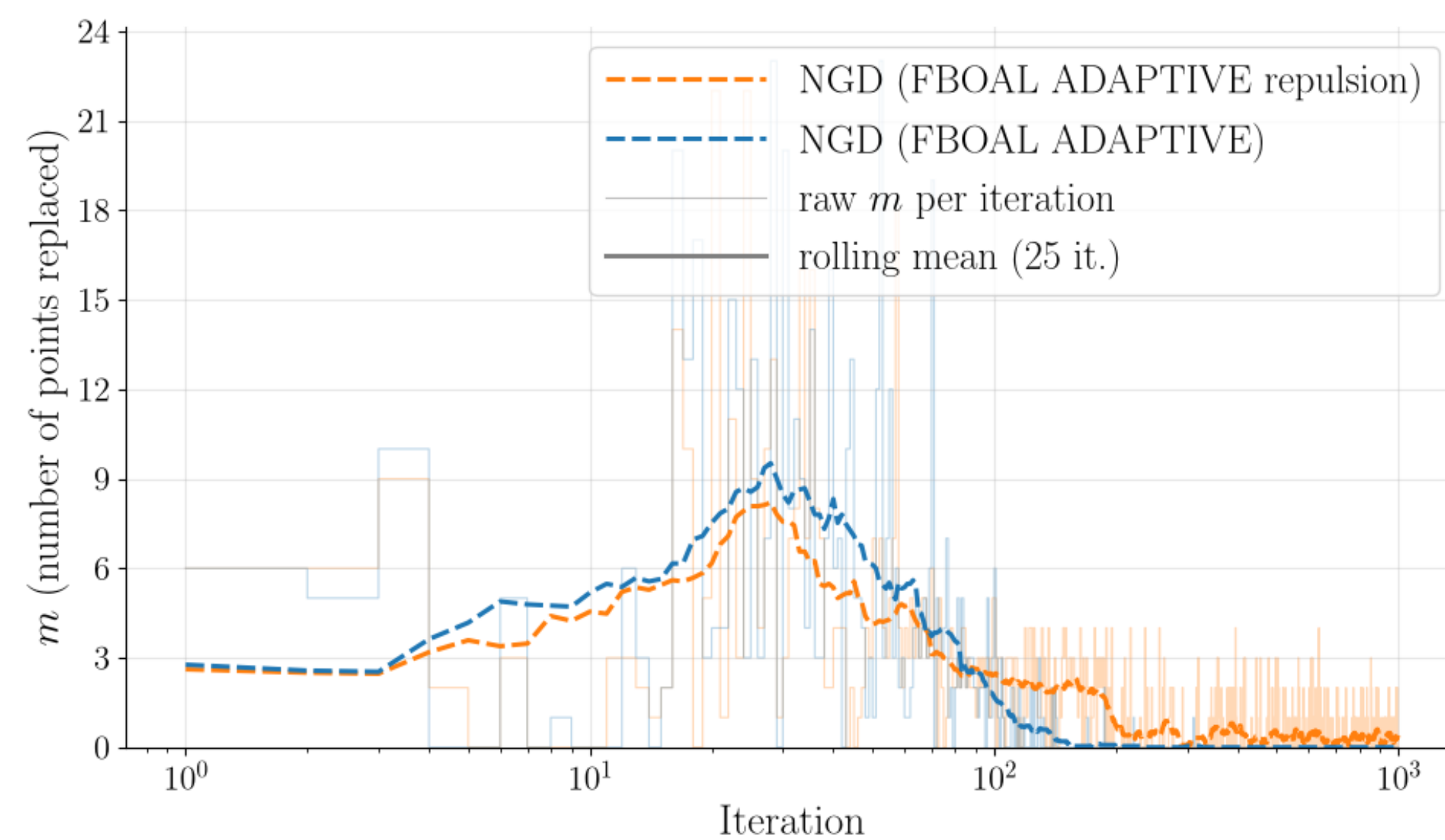


Loss over the Test Points during training

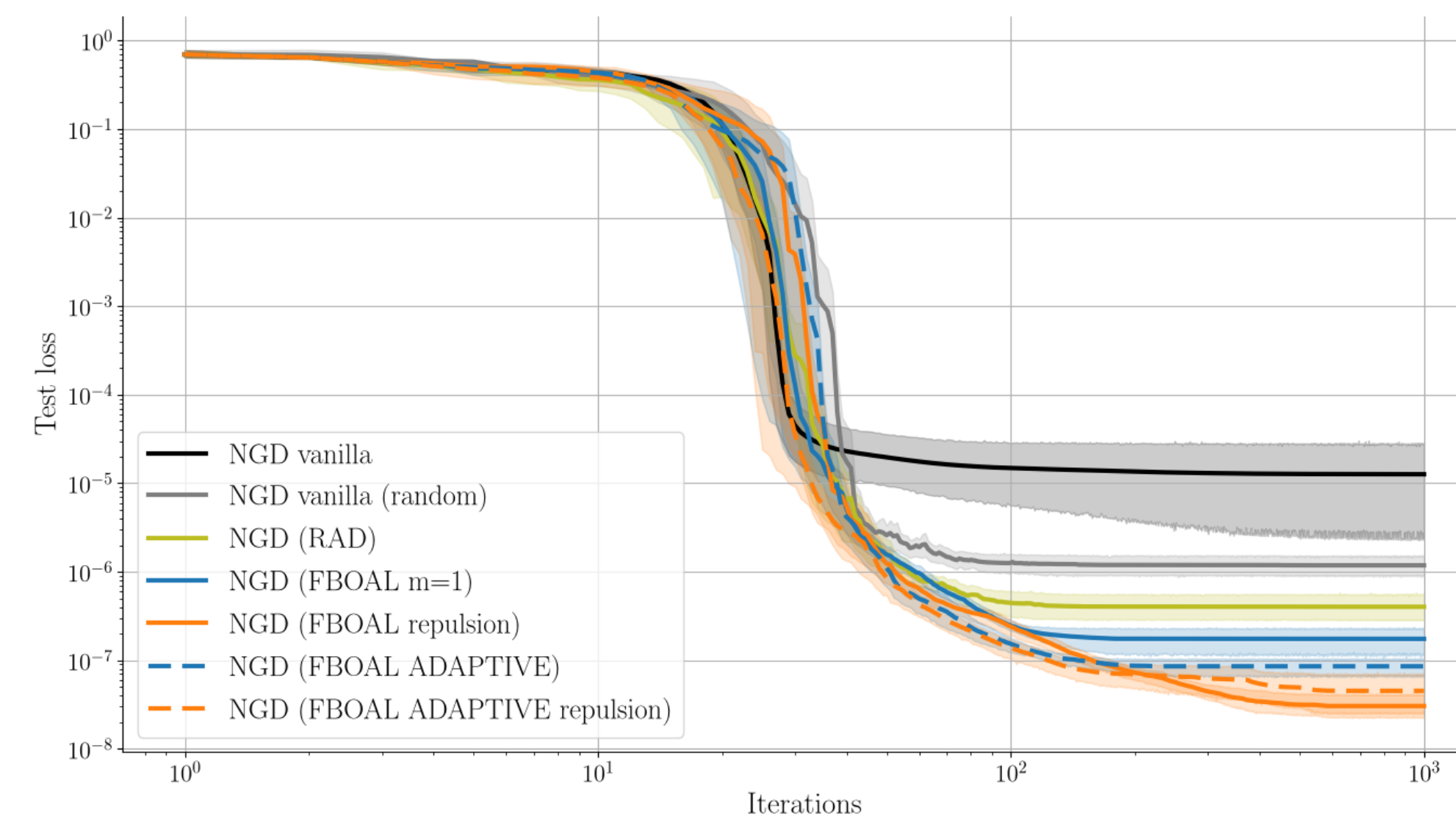
6

Viscous Burgers 1D

Repulsion and Adaptive m



Adaptive choice for m during training

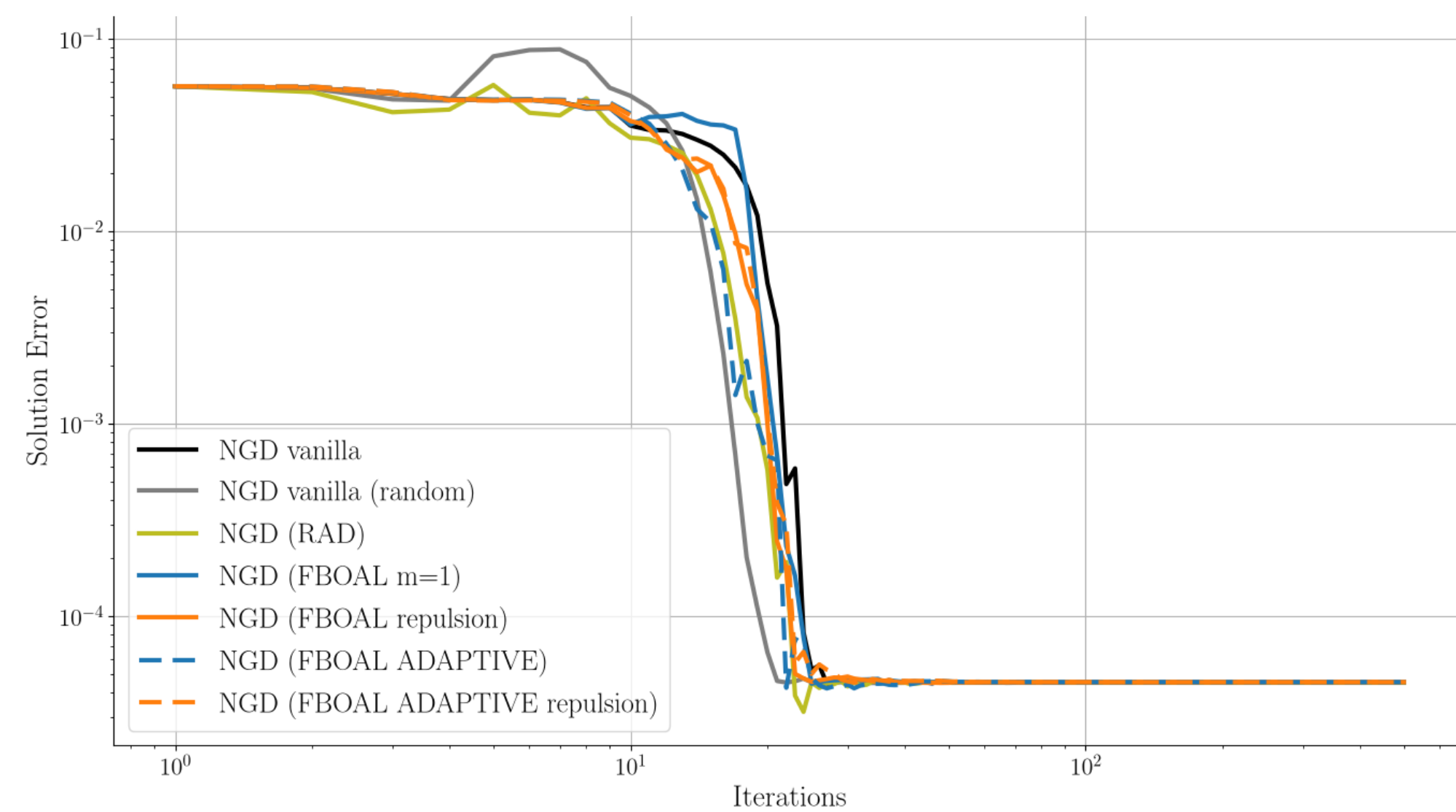


Loss over the Test Points during training

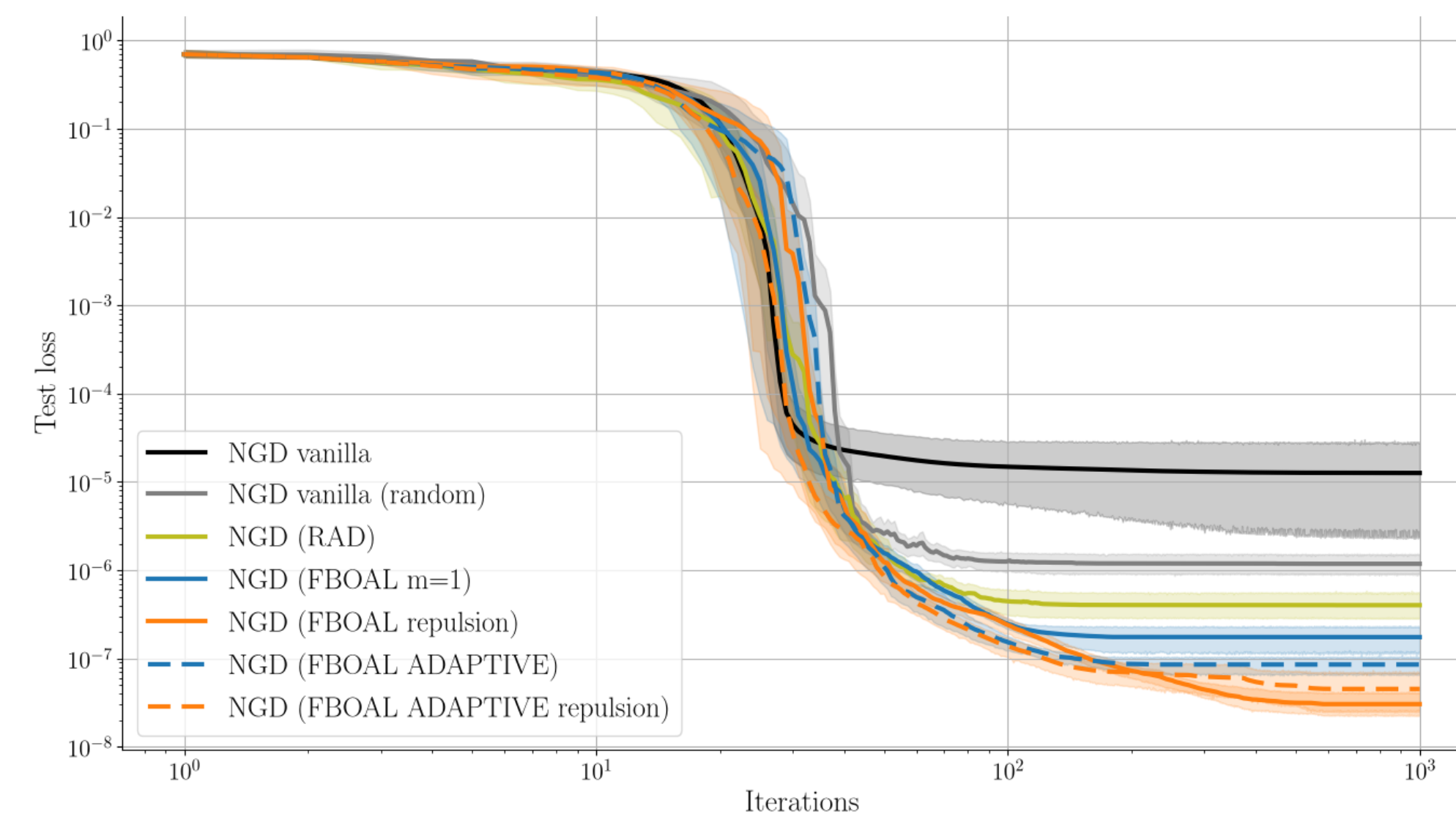
6

Viscous Burgers 1D

Repulsion and Adaptive m



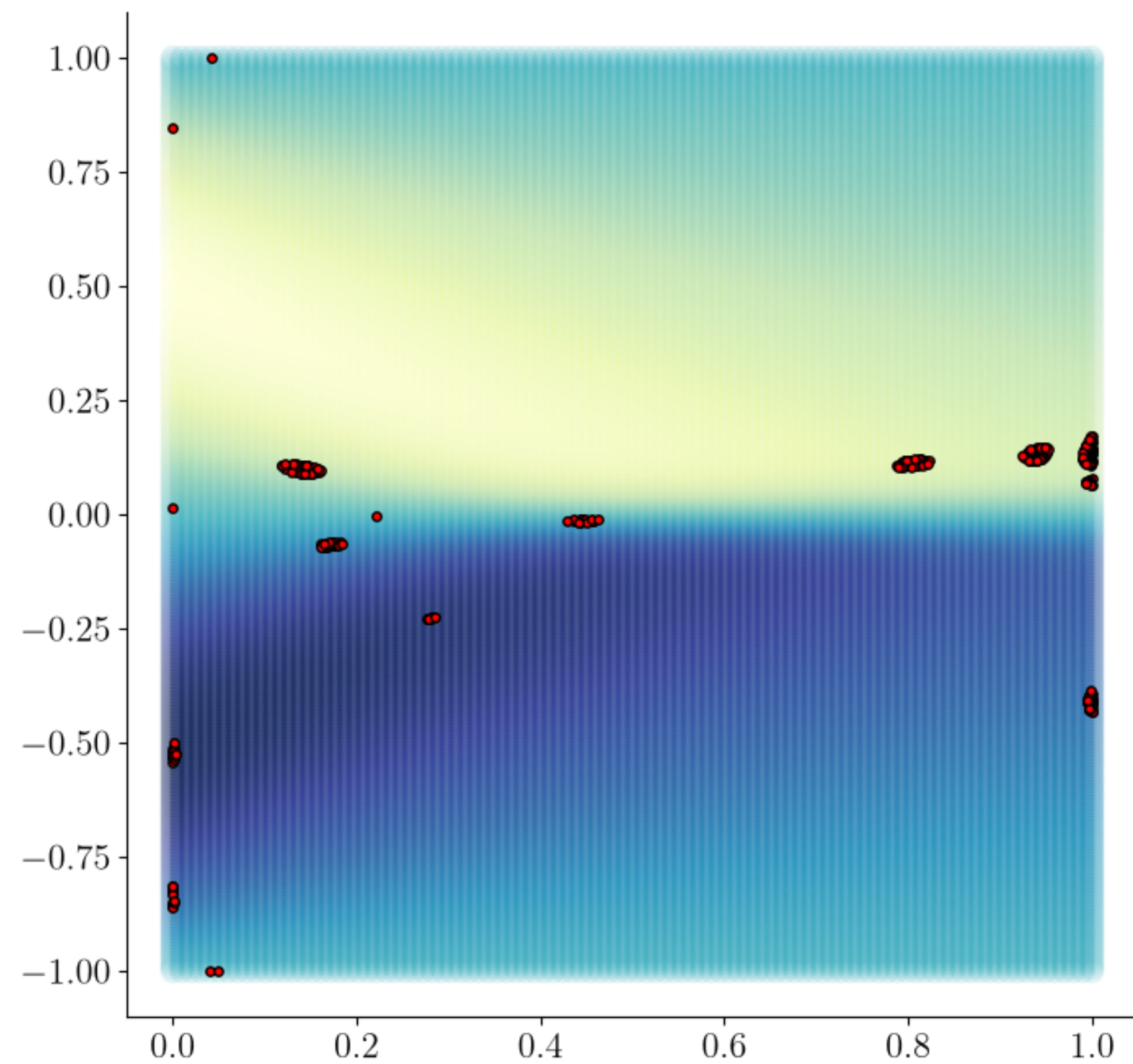
Prediction Error



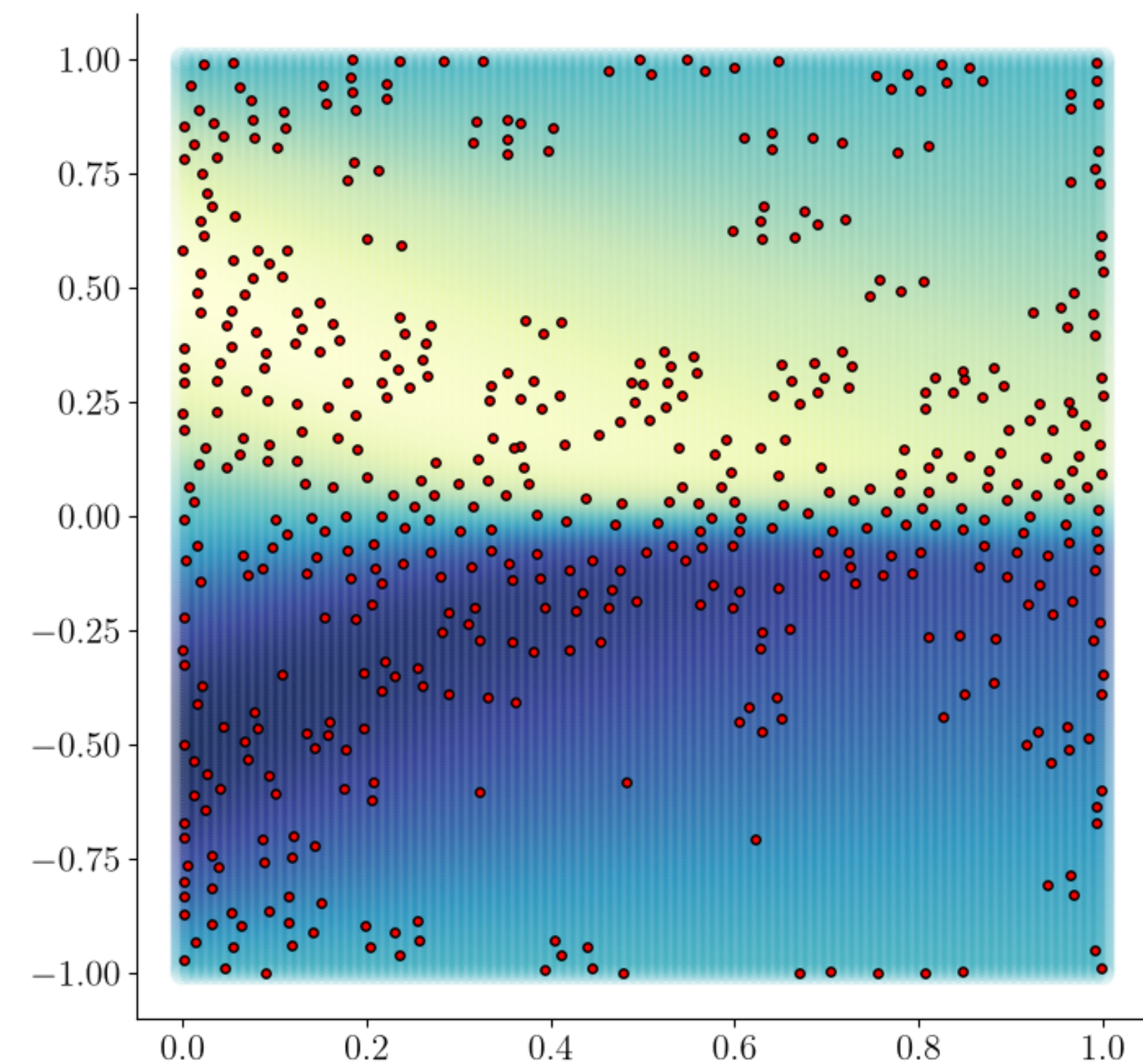
Loss over the Test Points during training

6

Viscous Burgers 1D



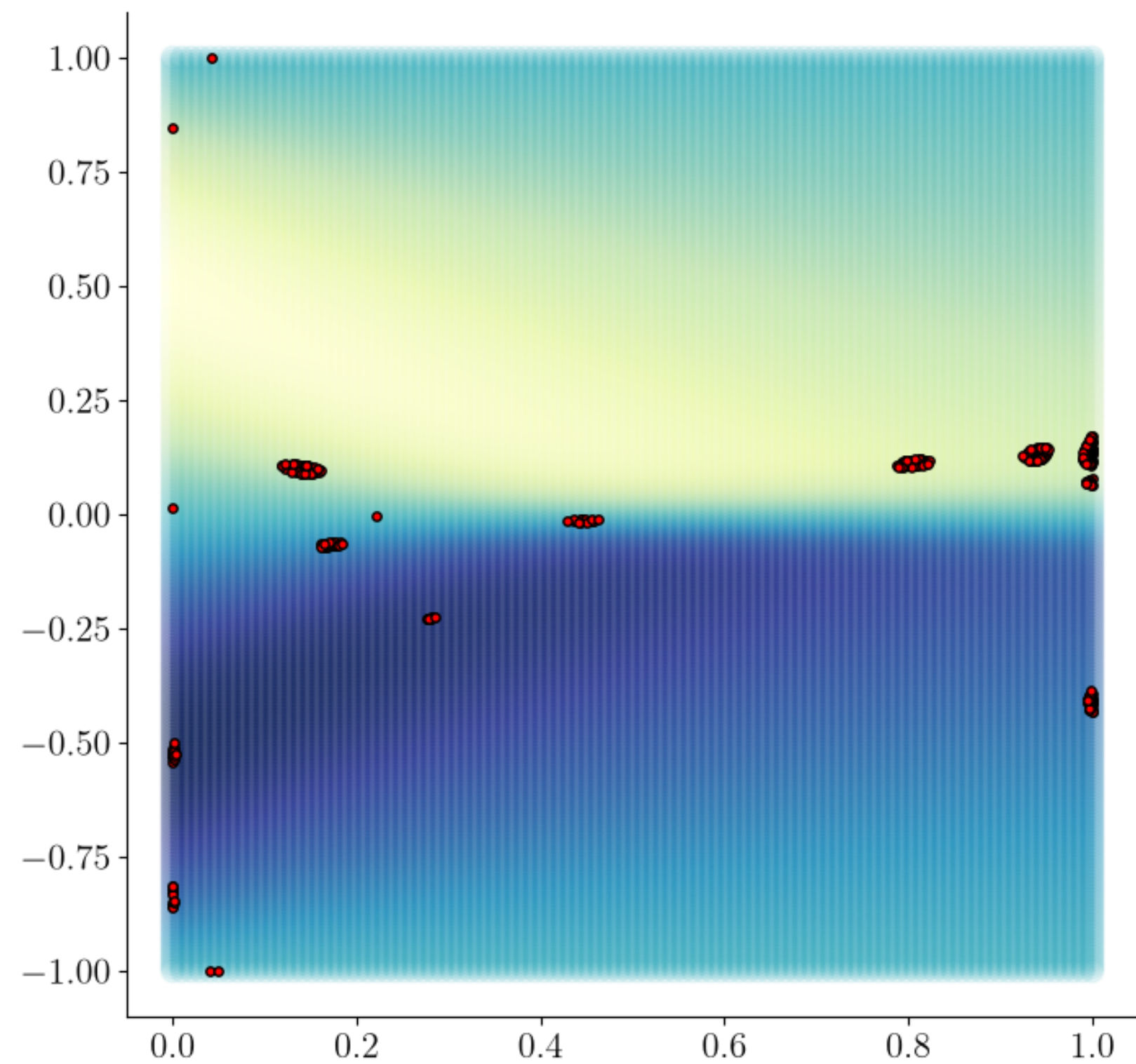
Collocation Points at the end of FBOAL



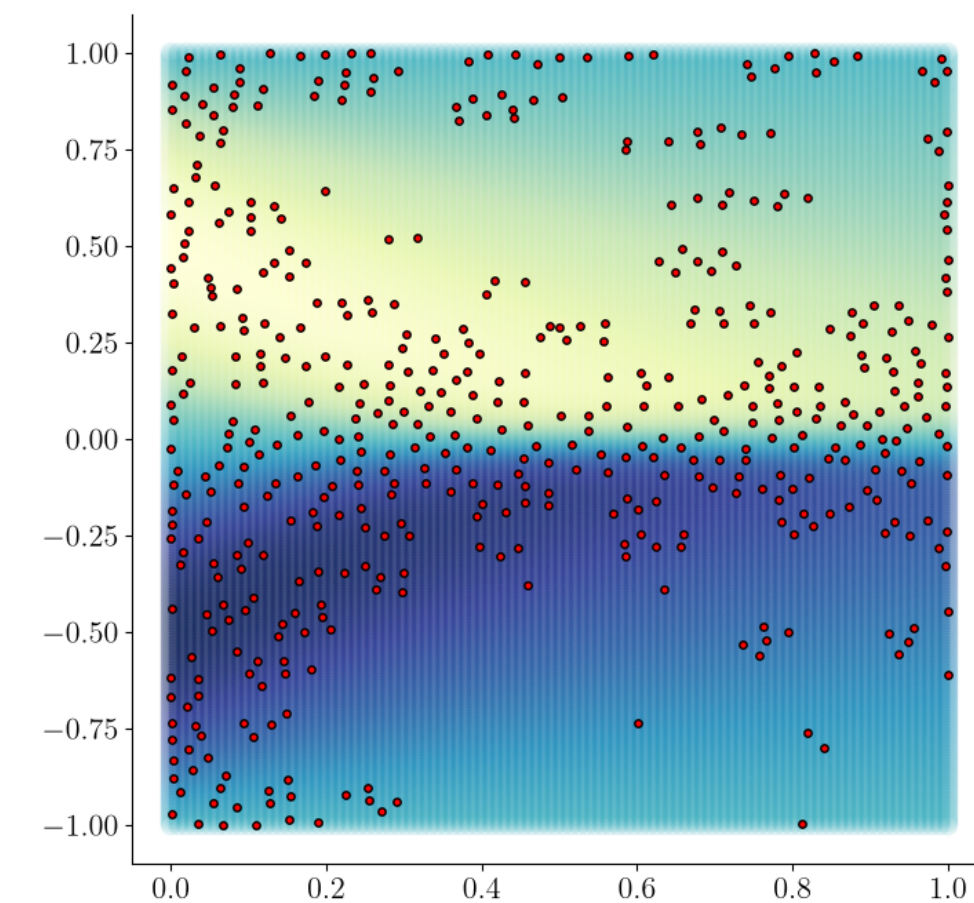
Collocation Points at the end of FBOAL
with repulsion

6

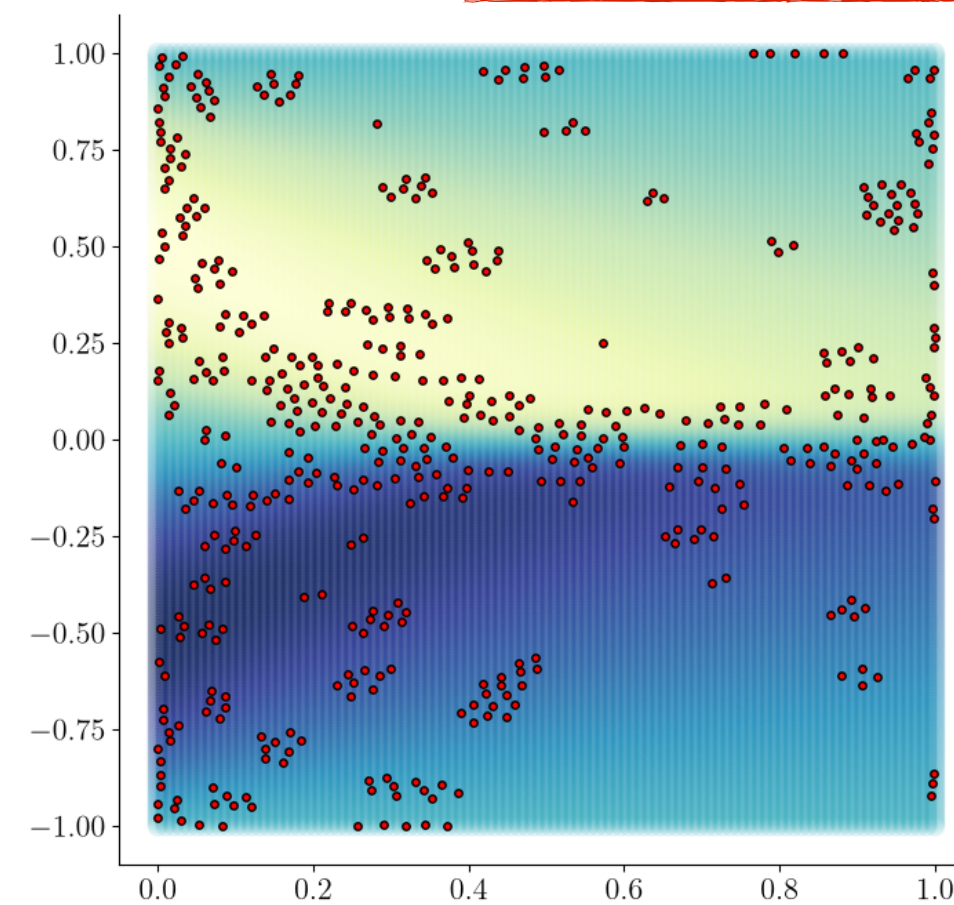
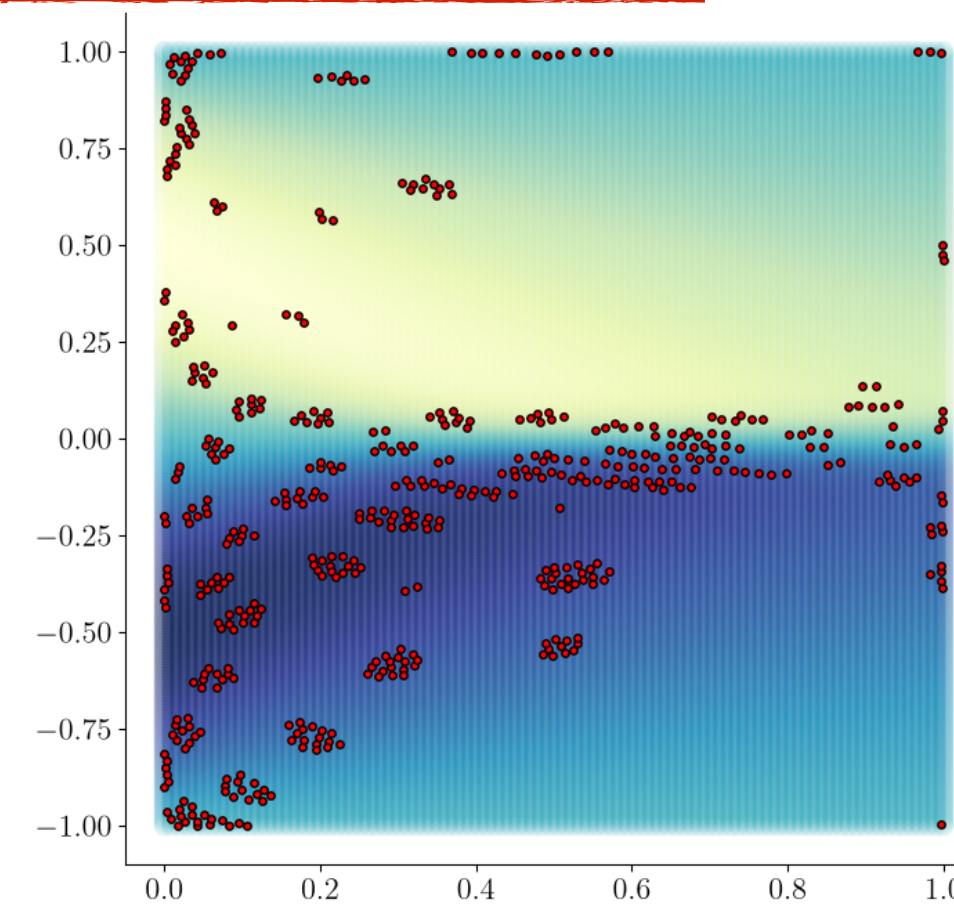
Viscous Burgers 1D



Collocation Points at the end of FBOAL



Varying the repulsion radius



Thank you for your attention !

Questions ??