Surrogate models

F.M. Velotti, F. Huhn, V. Baggiolini, R. Gorbonosov,
V. Kain, B. Rodriguez Mateos, M. Schenk, M. Sobieszek

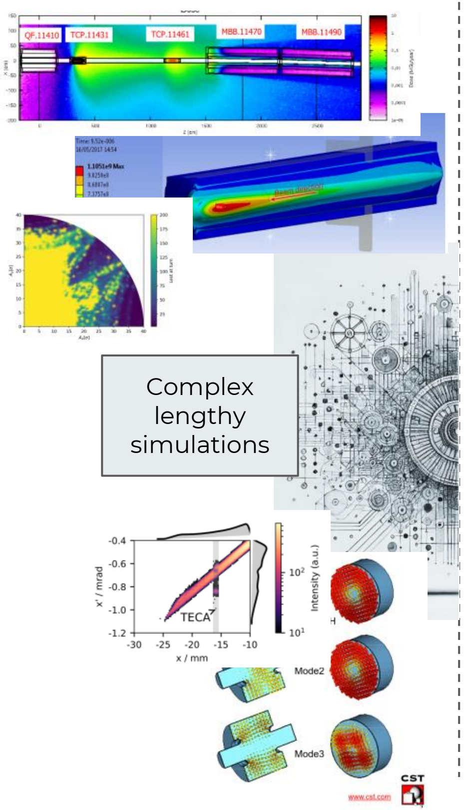
Outline

1. What are surrogate models?
2. Cherry picked CERN applications
3. Integration in the control system
4. Conclusions

TL;DR: What are surrogate models?

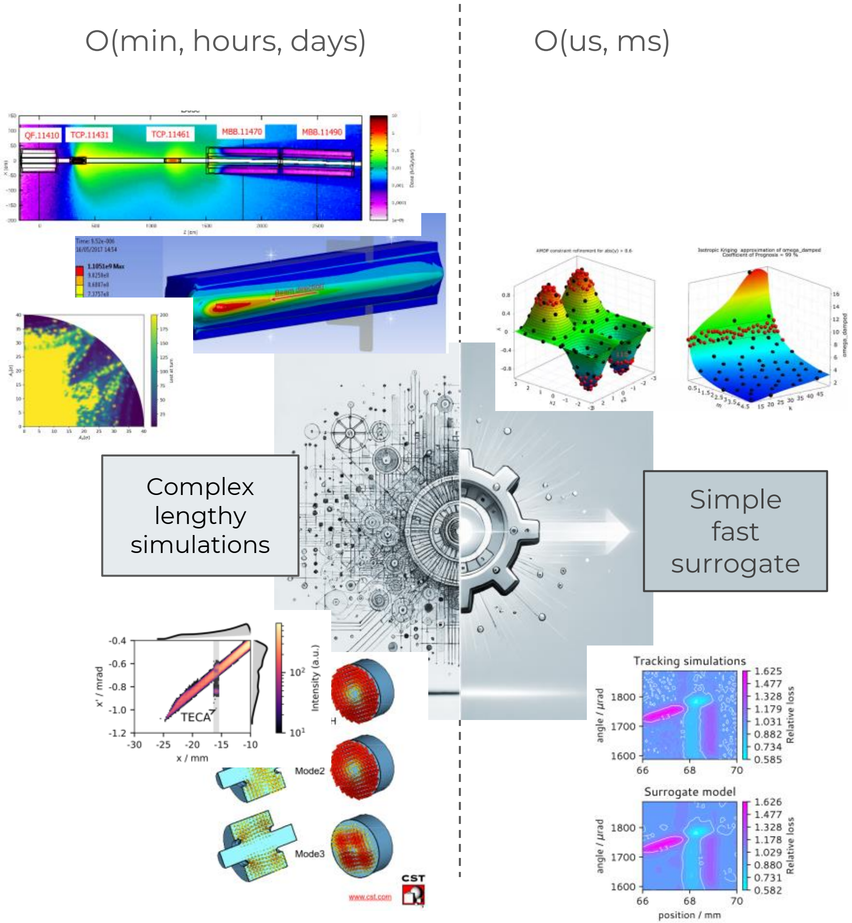
→ **Surrogate models (SM) are simplified mathematical models that approximate complex, computationally expensive simulations** - they can also include directly data from the real system

O(min, hours, days)



TL;DR: What are surrogate models?

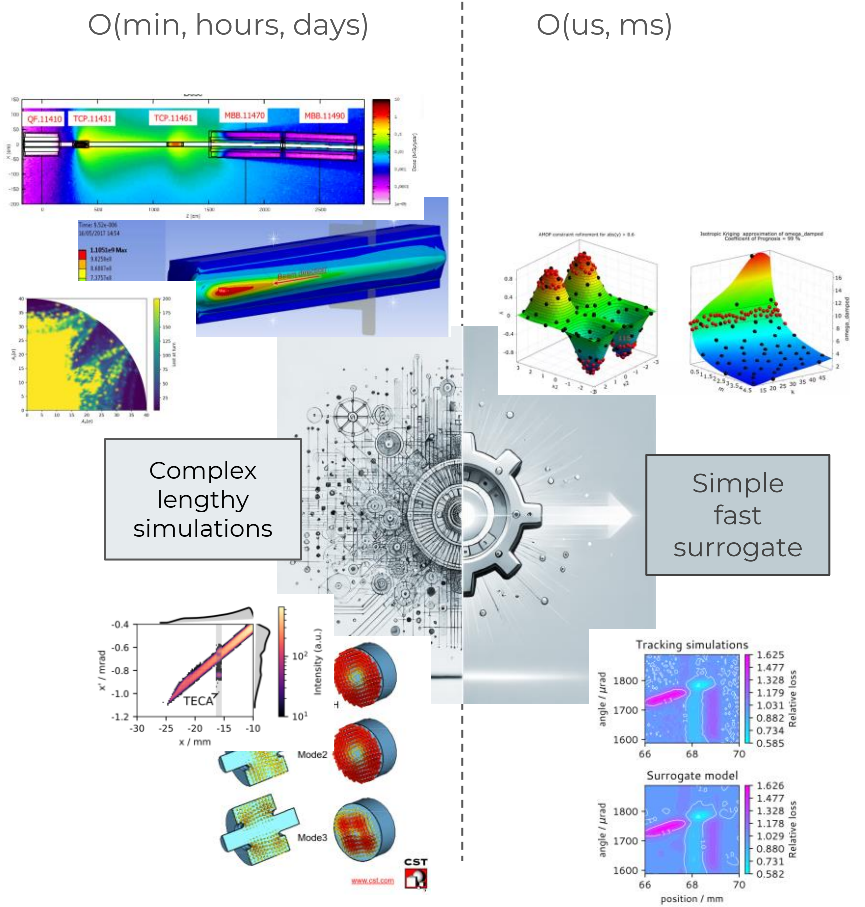
→ **<u>Surrogate models (SM) are simplified mathematical models that approximate complex, computationally expensive simulations</u>** - they can also include directly data from the real system



TL;DR: What are surrogate models?

→ **<u>Surrogate models (SM) are simplified mathematical models that approximate complex, computationally expensive simulations</u>** - they can also include directly data from the real system

→ They are commonly used in **<u>optimization, sensitivity analysis, and uncertainty</u>** quantification to reduce computation time

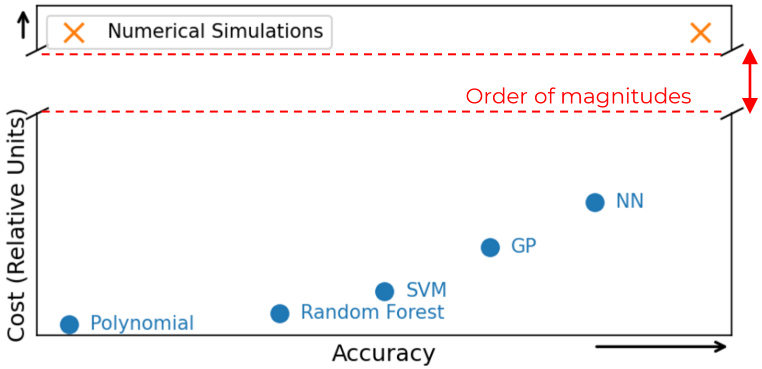


Why to use surrogate models

Main benefits:

➔ Enabling Large-Scale Optimization and Sensitivity Analysis

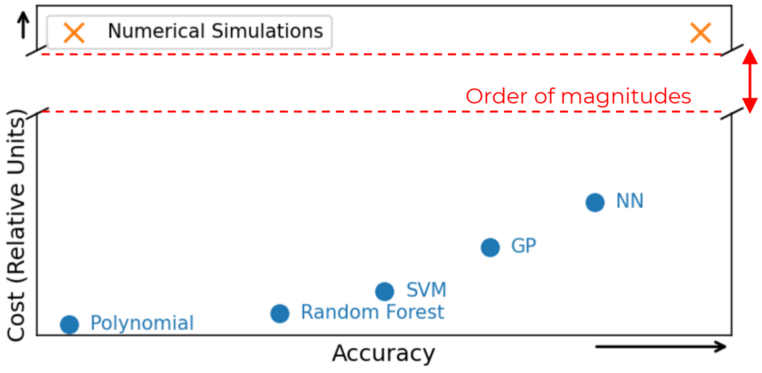
- Make feasible optimization of simulations that take long to run



Why to use surrogate models

Main benefits:

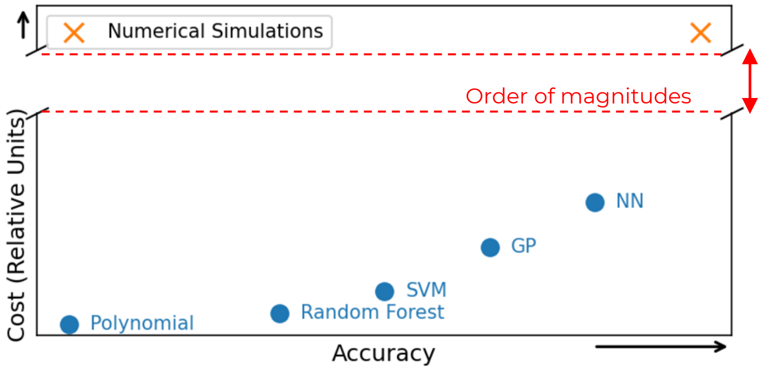
- ➔ Enabling Large-Scale Optimization and Sensitivity Analysis
  - Make feasible optimization of simulations that take long to run
- ➔ Facilitating Real-Time Decision Making and Control
  - Enable to have a real-time controller on arbitrary complex response
  - Possibility to study real system behavior to controllers and optimizers

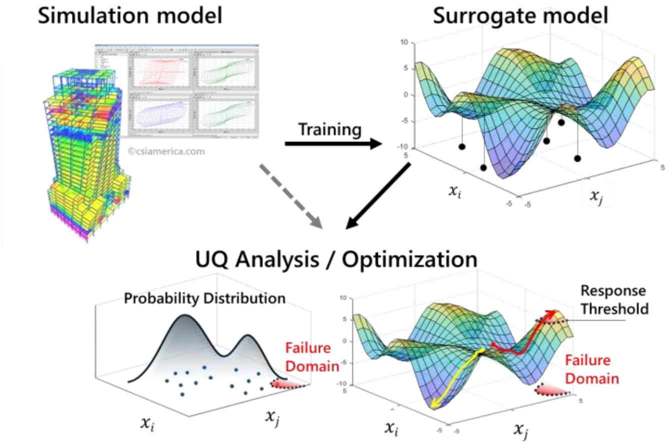


Why to use surrogate models

Main benefits:

- ➔ Enabling Large-Scale Optimization and Sensitivity Analysis
  - ◆ Make feasible optimization of simulations that take long to run
- ➔ Facilitating Real-Time Decision Making and Control
  - ◆ Enable to have a real-time controller on arbitrary complex response
  - ◆ Possibility to study real system behavior to controllers and optimizers
- ➔ Improved Interpretability and Understanding of Complex Systems
  - ◆ Enable the exploration of system parameter relationship and correlations





Why to use surrogate models

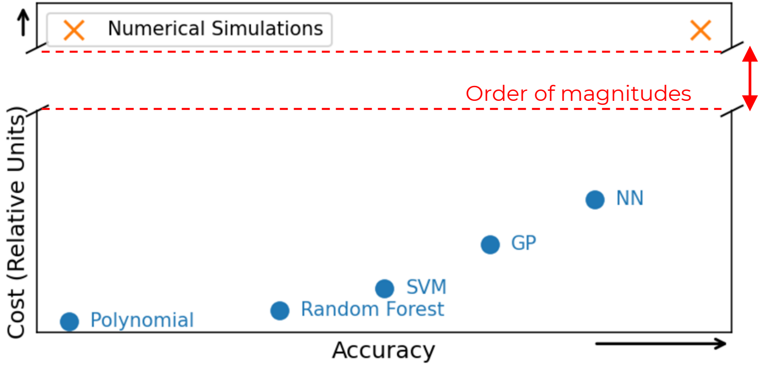
Main benefits:

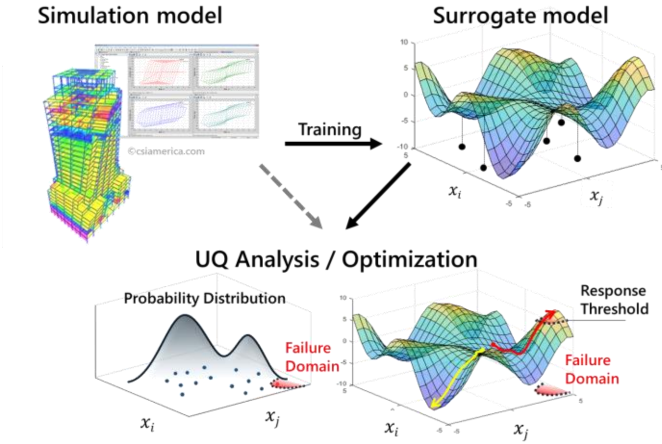
➔ Enabling Large-Scale Optimization and Sensitivity Analysis
- ◆ Make feasible optimization of simulations that take long to run

➔ Facilitating Real-Time Decision Making and Control
- ◆ Enable to have a real-time controller on arbitrary complex response
- ◆ Possibility to study real system behavior to controllers and optimizers

➔ Improved Interpretability and Understanding of Complex Systems
- ◆ Enable the exploration of system parameter relationship and correlations

➔ Enhanced Robustness and Simplicity
- ◆ Smooth functions to reduce sensitivity to noise

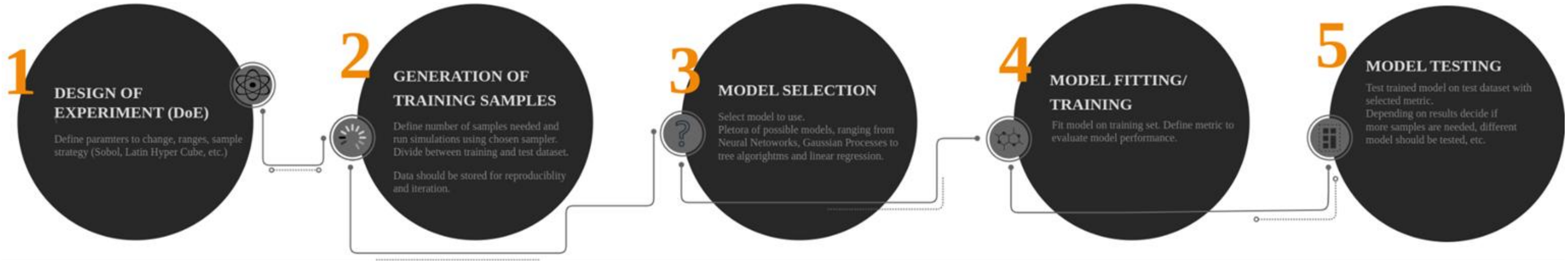




How to make a surrogate model

Surrogate model

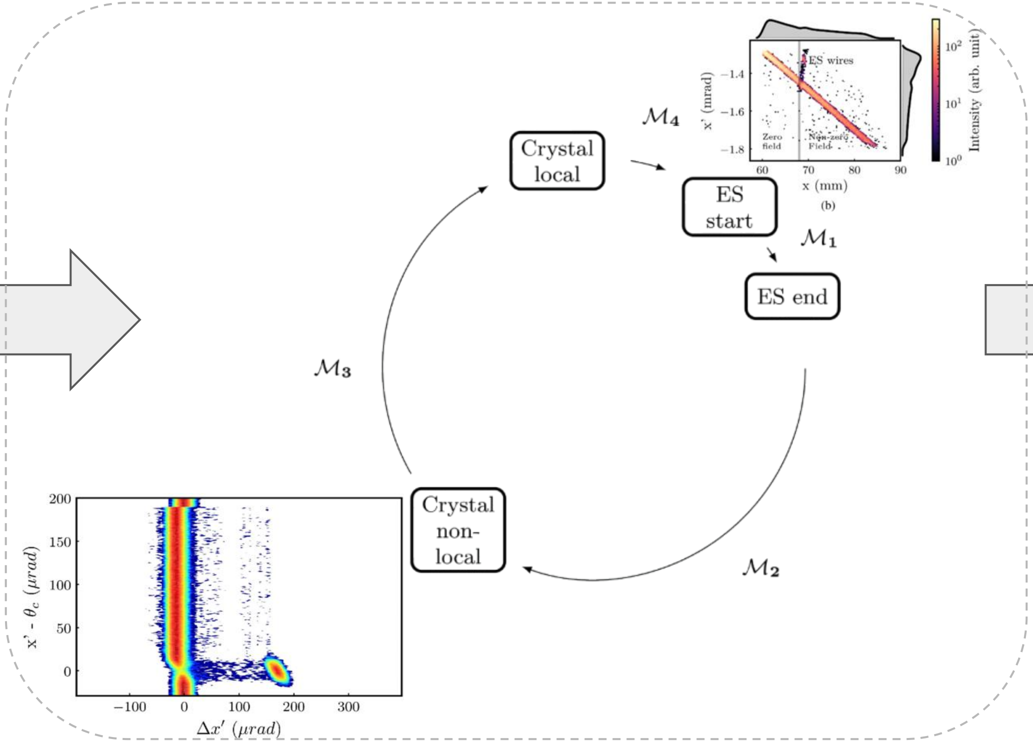
How to



1D example - crystal shadowing

- We have a simulation model (e.g. crystal shadowing simulations) that depends on a single parameter (e.g. angle)
    - Each simulation point takes some time O(10')

Input
Crystal angle

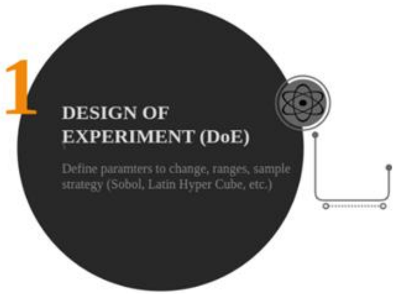


Output
Losses ZS

How to make a surrogate model

Surrogate model

How to



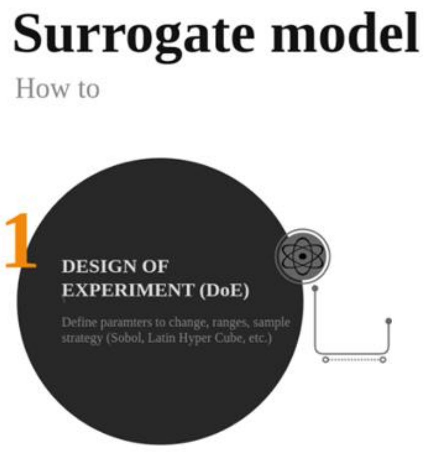
➔ Aim: **use the minimum number of samples to cover reasonably the space to explore**

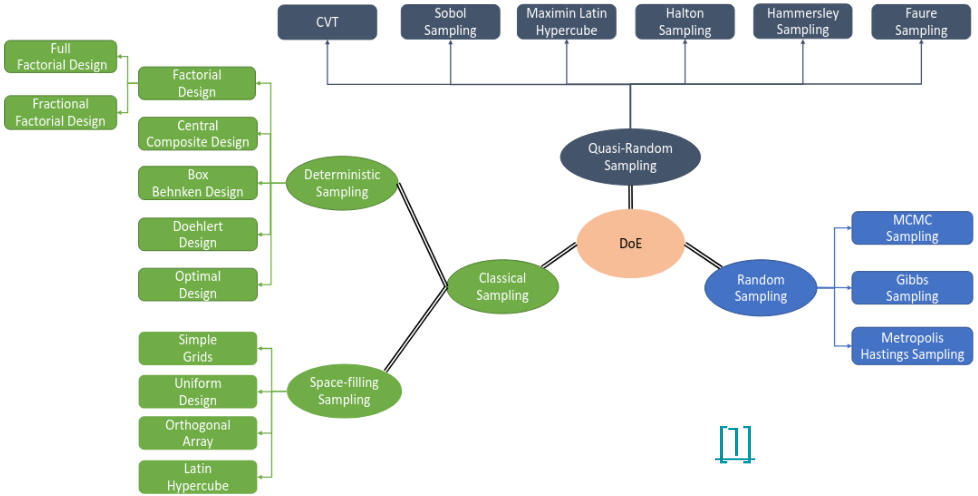
➔ Rather old concept (Fisher in 1926 [1])

Input
Crystal angle

```
np.linspace(-175, 175, 50)
```

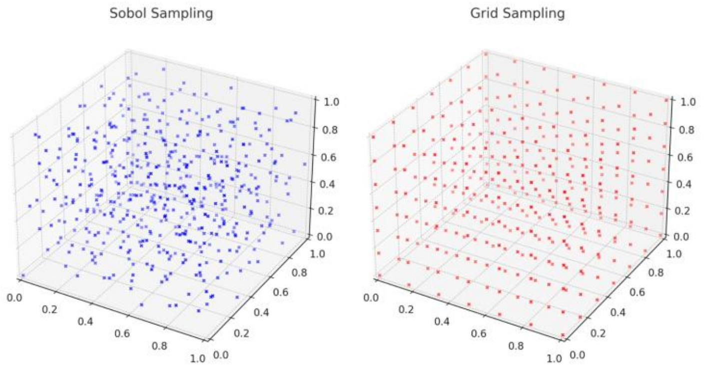
How to make a surrogate model



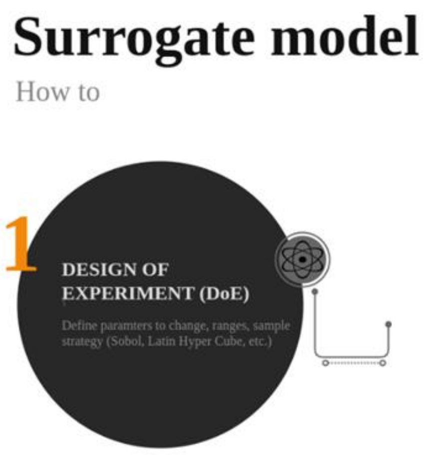


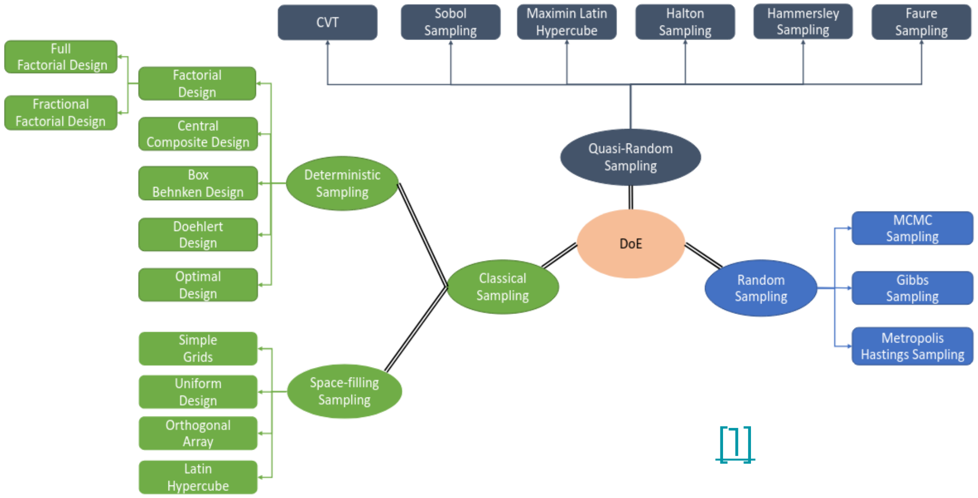
[1]

- ➔ Aim: **use the minimum number of samples to cover reasonably the space to explore**
- ➔ Rather old concept (Fisher in 1926 [1])



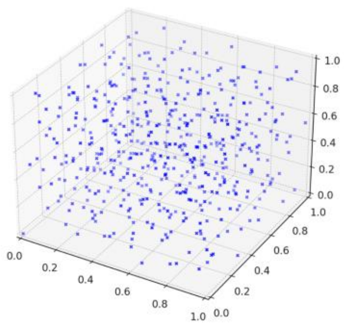
How to make a surrogate model

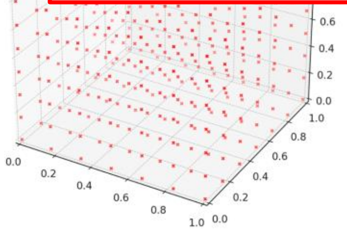




[1]

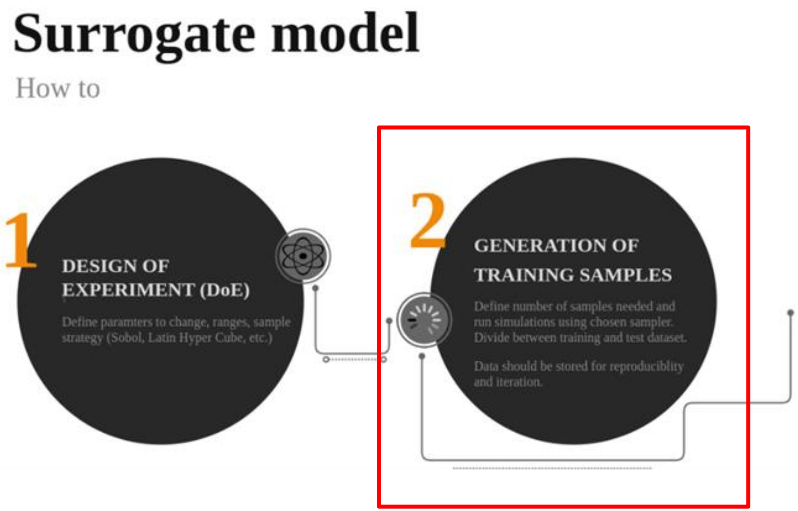
- ➔ Aim: **use the minimum number of samples to cover reasonably the space to explore**
- ➔ Rather old concept (Fisher in 1926 [1])

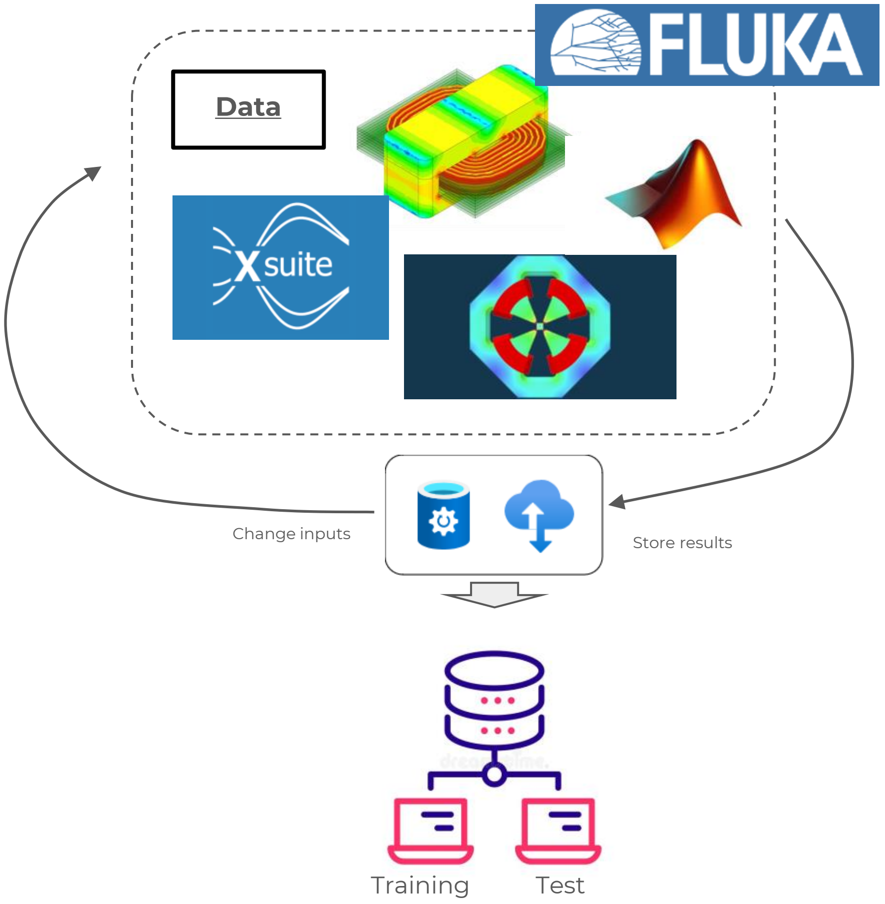




Grid scans are incredibly inefficient!!

How to make a surrogate model

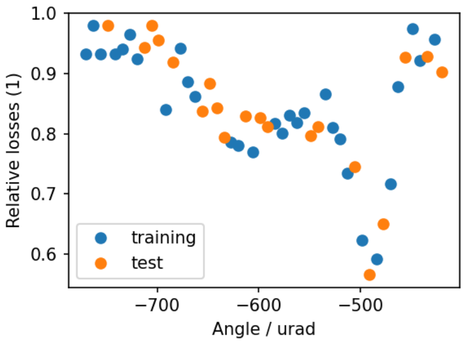




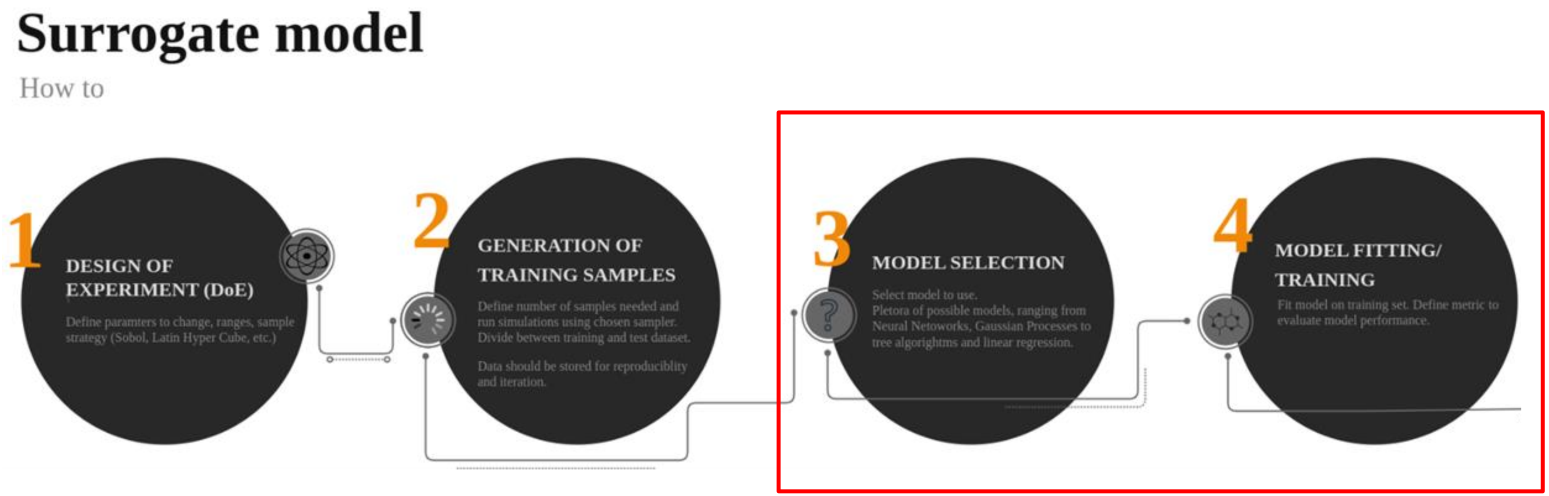
- ➔ Generate the datasets
- ➔ It can be iterated with the DoE to optimize the space coverage
- ➔ **Data can be from both simulations and data**

1D example - crystal shadowing

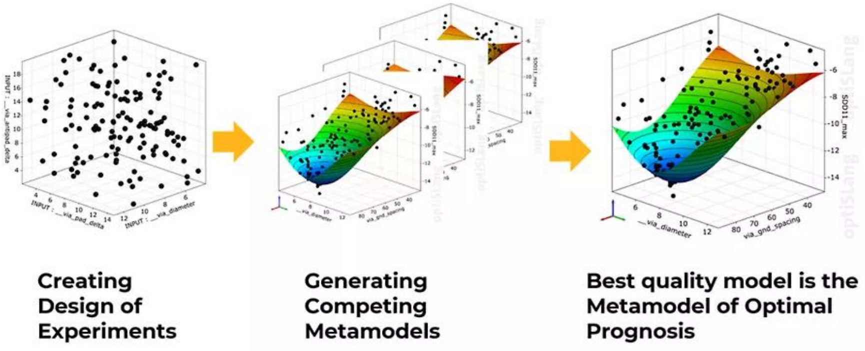
- ➔ We have a simulation model (e.g. crystal shadowing simulations) that depends on a single parameter (e.g. angle)
  - ◆ Each simulation point takes some time O(10')
- ➔ We collect simulations changing the angle



How to make a surrogate model



- ➔ Model selection and fitting go together
  - ◆ Usually very dependent on the problem at hand
  - ◆ One of the main decision is if uncertainty should be estimated too → need probabilistic model in case
- ➔ Models to be modified on testing of part of the training dataset

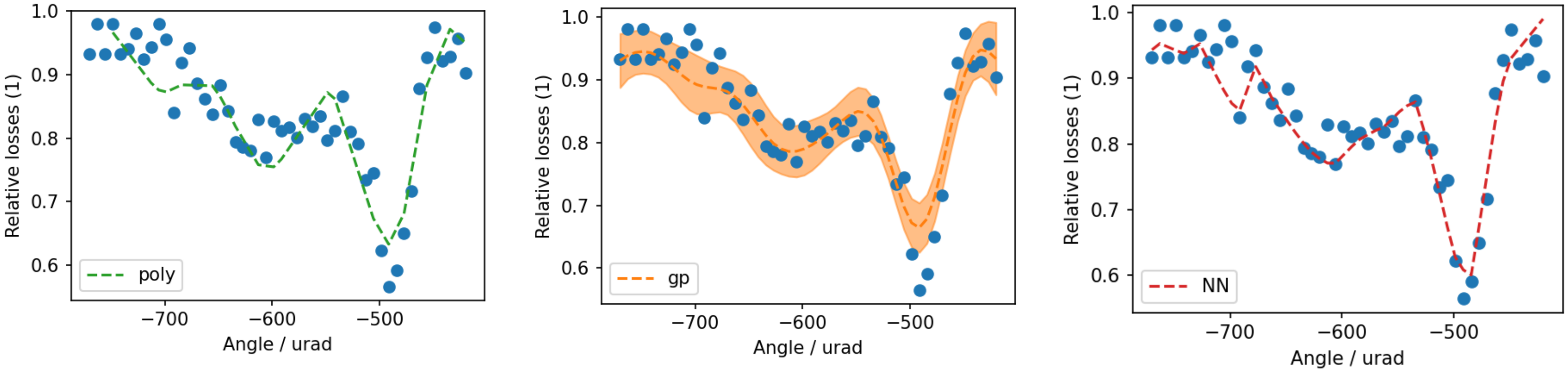


[ansys]

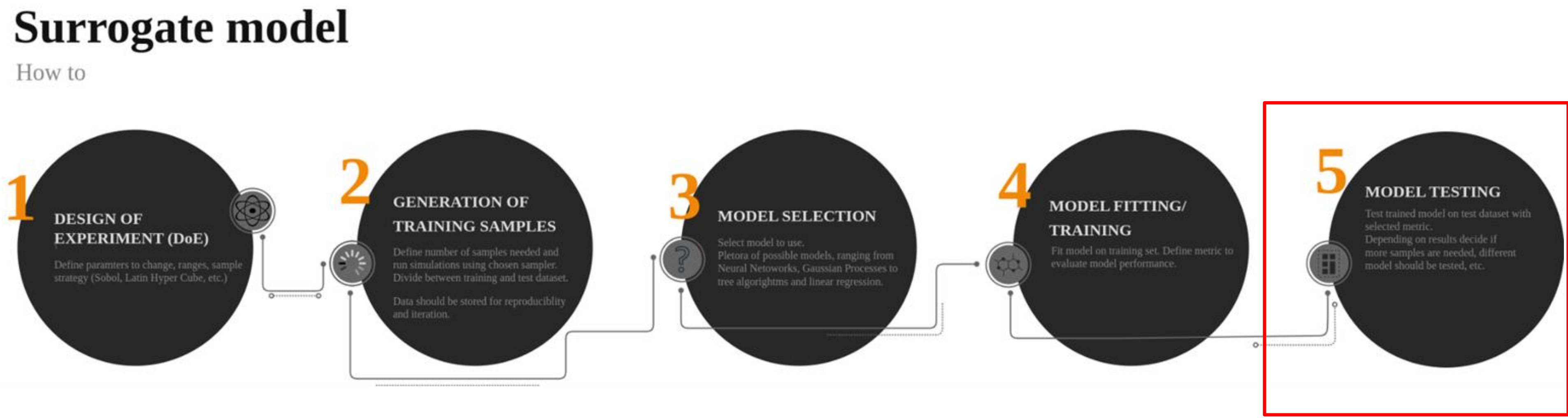
1D example - crystal shadowing

CERN

- ➔ We have a simulation model (e.g. crystal shadowing simulations) that depends on a single parameter (e.g. angle)
  - Each simulation point takes some time O(10’)
- ➔ We collect simulations changing the angle
- ➔ Then we fit a few models:
  - A 15 degree polynomial
  - A Gaussian Process [2]
  - A neural network...



How to make a surrogate model



➔ Model then tested on test set

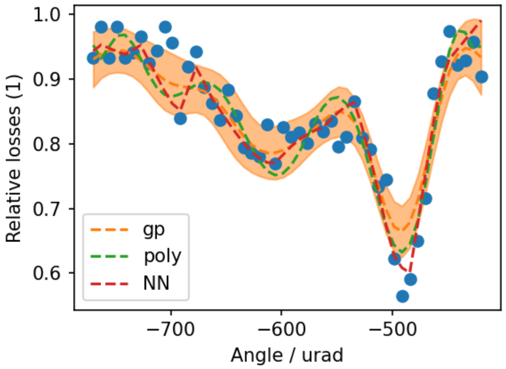
How to make a surrogate model

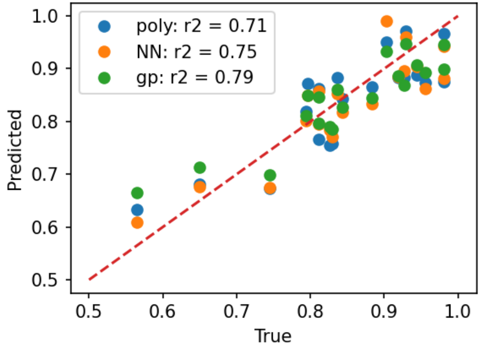
- ➔ Model then tested on test set
- ➔ if not good enough get more samples

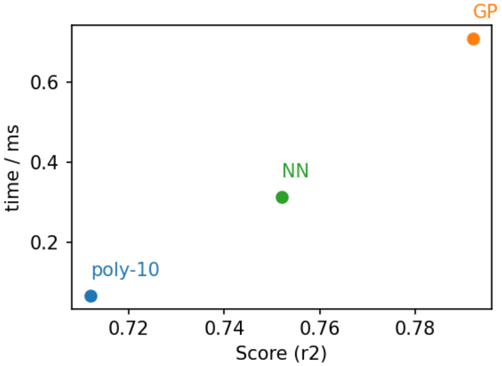
⇒ **<u>Ideally, samples from DoE or from optimization!</u>**

Real example 1D - crystal shadowing

- ➔ We have a simulation model (e.g. crystal shadowing simulations) that depends on a single parameter (e.g. angle)
  - ◆ Each simulation point takes some time O(10')
- ➔ We collect simulations changing the angle
- ➔ Then we fit a few models:
  - ◆ A 15 degree polynomial
  - ◆ A Gaussian Process [2]
  - ◆ A neural network...
- ➔ And finally we test them → if results are satisfactory, we have our SM!

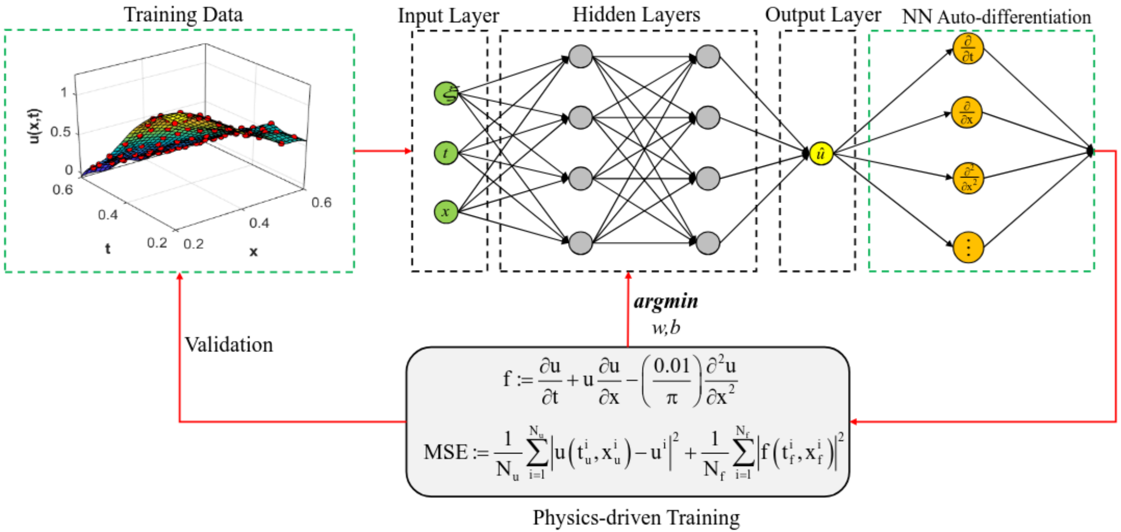


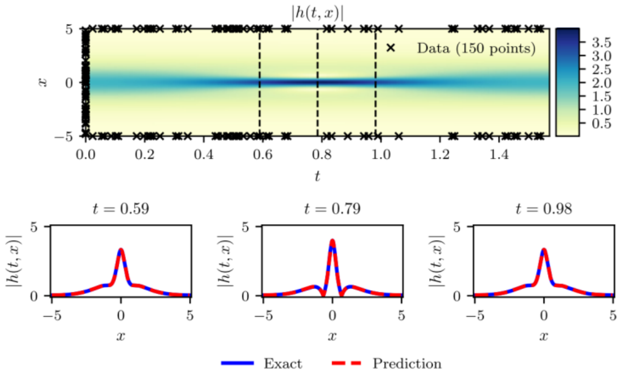




[Advanced] surrogate models: PINN

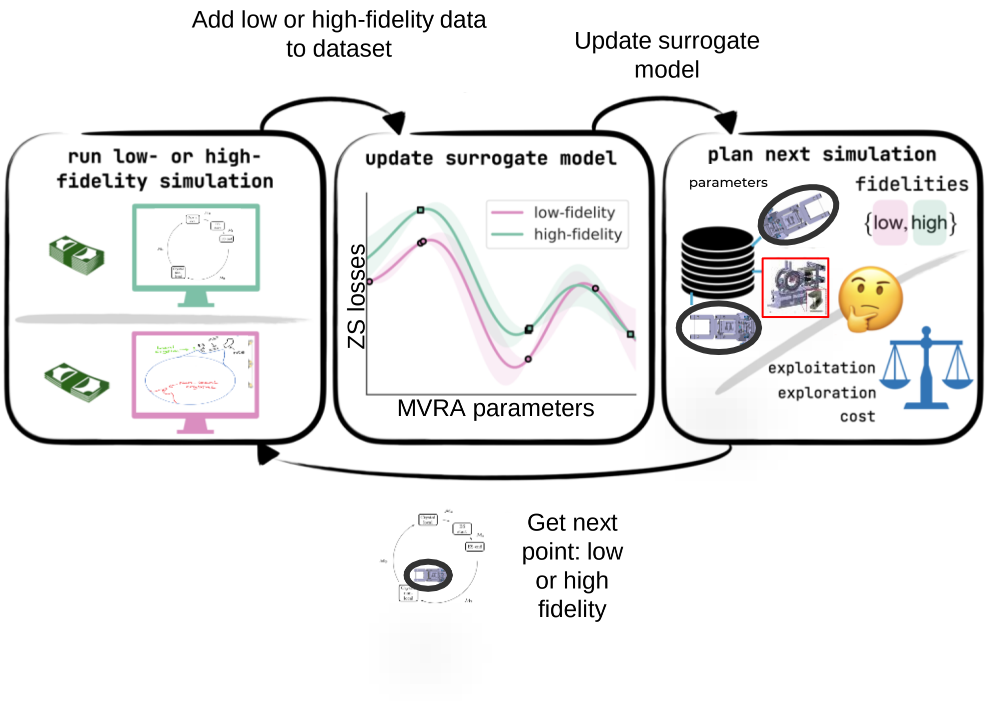
- ➔ To make SM more accurate, one can exploit known physics of the system
- ➔ Training can be constrained by imposing that the solution should satisfy a given physical law ⇒ Physics Informed Neural Networks (PINN)
  - ◆ Basically one can use a NN as a generic PDE solution...quite some applications!





[Advanced] surrogate models: multi-fidelity

- ➔ Simulations can be made more or less accurate → more or less fidelity[1]
- ➔ Data are the highest fidelity
- ➔ Fidelity can be treated as additional dimension and hence have the model be able to distinguish it and exploit it
  - ◆ High fidelity comes with a cost!
- ➔ More details in F. Huhn's talk [3]



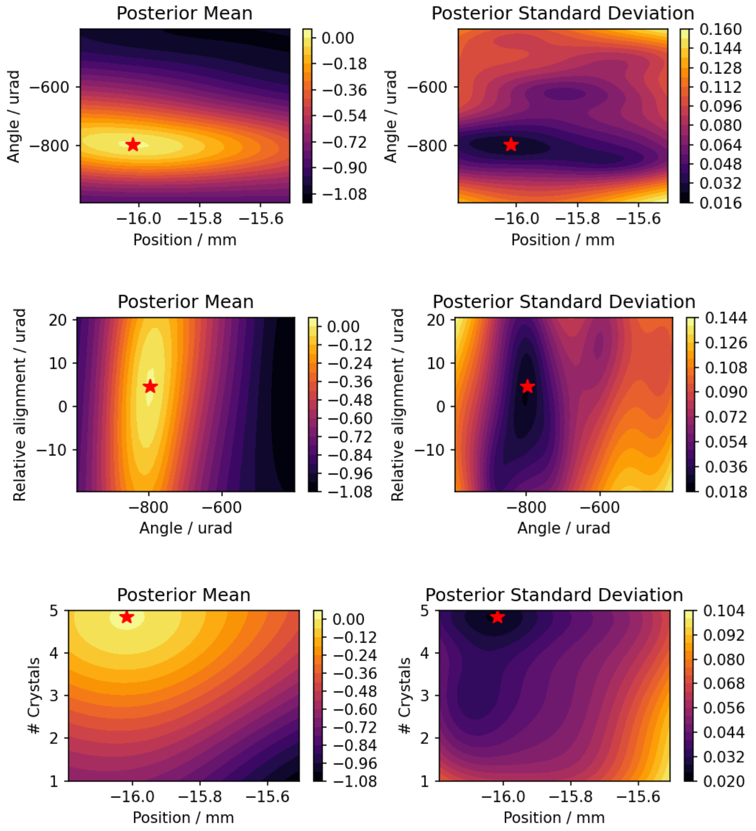
1: level of accuracy and detail with which a model represents a system or process. High-fidelity (HF) models provide precise and detailed simulations but are computationally intensive. Low-fidelity (LF) models, while less accurate, are computationally cheaper and faster to execute

Cherry picked CERN applications

Crystal shadowing → from 1D to 6D

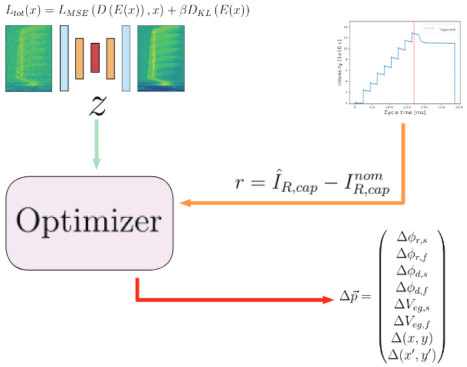
- ➔ New crystal technology can lead to **x10 loss reduction: Multi Volume Reflection Array (MVRA)**
- ➔ Series of N crystals → many parameters to optimize for in design phase
  - ◆ Angle
  - ◆ Position
  - ◆ # crystals
  - ◆ Relative alignment
  - ◆ Bending direction
  - ◆ Crystal width
- ➔ **Multi-fidelity Bayesian Optimization to find optimal configuration ⇒ hopefully to be tested in the SPS in 2026!**

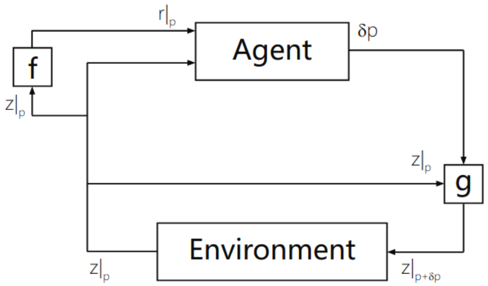
Direction	*Angle (urad)*	*Position (mm)*	*Width (mm)*	*Relative angle (urad)*	*N. Crystals (1)*
VR inside	-789	-16.03	1.4	-7	5
VR outside	-562	-15.96	1.3	-9	5



LEIR data-driven surrogate modeling

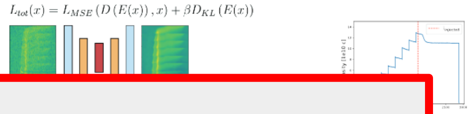
- ➔ **LEIR injection efficiency optimization**
  - Today done using VAE to encode Schottky spectrum in low dimensions and ring intensity → optimizer to correct energy ramping and debunching cavity phases, e-gun voltage, cooler and injection bump and BHN10
- ➔ **Move towards offline training of an RL agent on the surrogate model**
  - Data-driven surrogate model of the injection





[Borja Rodriguez Mateos]

LEIR data-driven surrogate modeling



→ **LEIR injection efficiency**
op

There is so much more!

- **Fully data-based models** (PFW [4], ZS-TT20 activation [5], etc.)
- **Virtual diagnostics** (hysteresis compensation [6], MKP temperature [7], etc.)
- **Simulation optimization** (FCC EM separator [3], MKDH model [3], etc.)
- **Digital twins**
- **Online models** (MTE optimization [10], spill optimization [11], etc.)

More details in F. Huhn talk [3] and A. Lu [6]

→ **Mo**
of
surrogate model

- Data-driven surrogate model of the injection

[Borja Rodriguez Mateos]

Integration in the control system

Infrastructure needed to scale it up

- ➔ In first approximations, nothing is needed → simulations outputs can be stored on local HDs (??) and on EOS/AFS/other could systems
  - Models can be “pickled” or “jsoned” and stored too

Infrastructure needed to scale it up

- ➔ In first approximations, nothing is needed → simulations outputs can be stored on local HDs (??) and on EOS/AFS/other could systems
  - Models can be “pickled” or “jsoned” and stored too
- ➔ Why do we need an “infrastructure” for SM?
  - **Continuity** → Ensure that a model “survives” the owner change
  - **Sharing** → easily share a SM with others that may need the same one
  - **Deployment** → SMs can be used as quick online models in operation

Storage of simulations data for SM

- ➔ Fundamental to preserve source data for SM

Storage of simulations data for SM

- ➔ Fundamental to preserve source data for SM
  - The SM is only good as its source data…

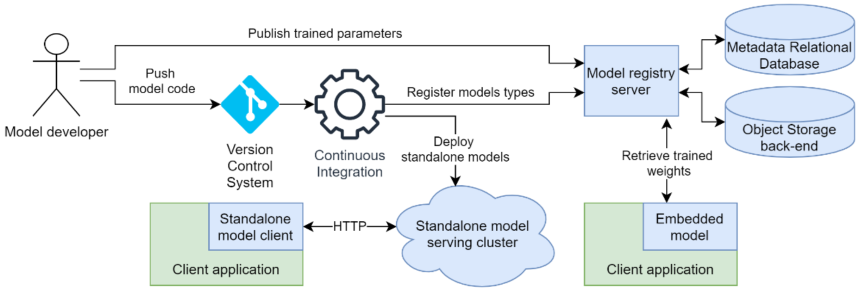
Storage of simulations data for SM

- ➔ Fundamental to preserve source data for SM
  - ◆ The SM is only good as its source data…
- ➔ Ideally we can use **NXCALS with the ingestion API in python**
  - ◆ Already available in Java
  - ◆ **Python native API should be possible and needed for most of our use-cases**
  - ◆ Timescale to be agreed upon given resources (action to follow up)

Serving and monitoring @ CERN

➔ **Serving:**

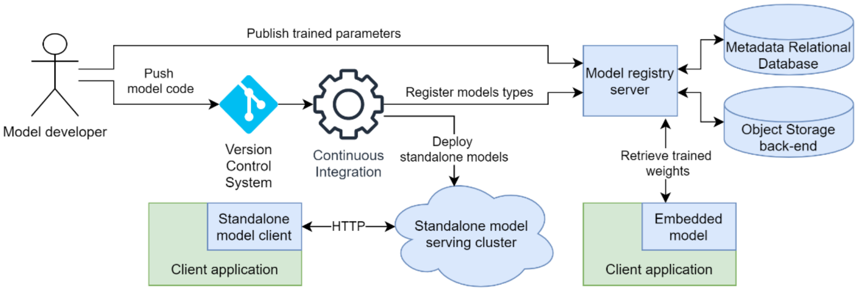
- We have the **Machine Learning Platform** [12] ⇒ used in operation in a few cases
- It offers both standalone and local inference
- Great for model versioning and integration in the control system ⇒ simplification needed to be streamlined



Serving and monitoring @ CERN

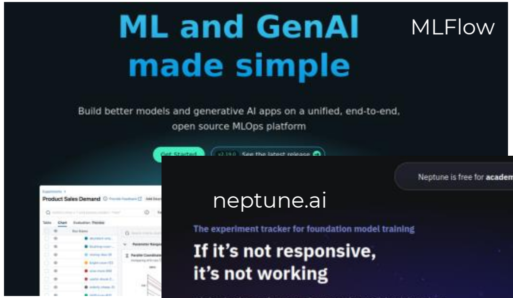
→ **Serving:**

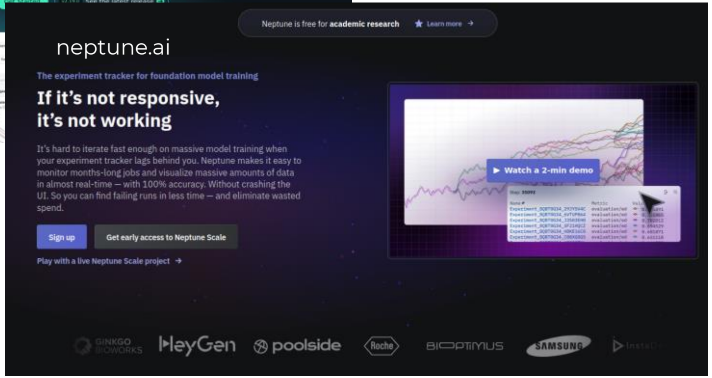
- We have the **Machine Learning Platform** [12] ⇒ used in operation in a few cases
- It offers both standalone and local inference
- Great for model versioning and integration in the control system ⇒ simplification needed to be streamlined



→ **Monitoring:**

- Need to use own tools (so many available open source)
- Hard then to close the loop on updating model in MLP...
- GitLab already offers way to monitor and register models via MLflow [13] - can this be a good way to explore? **⇒ need to solve this to be able to have up-to-date models automatically!**





Conclusions

➔ Surrogate modelling is a tool that can be used to tackle quite a few problems

Conclusions

- ➔ Surrogate modelling is a tool that can be used to tackle quite a few problems
- ➔ Still not fully exploited as in other domains/industry

Conclusions

- ➔ Surrogate modelling is a tool that can be used to tackle quite a few problems
- ➔ Still not fully exploited as in other domains/industry
- ➔ Infrastructure for full deployment potentially there
  - Missing NXCALS ingestion API in python ⇒ solution identified
  - MLP as is can be used for model versioning and standalone/local inference ⇒ done
  - Missing monitoring/performance tracking for continuous deployment ⇒ decision needed

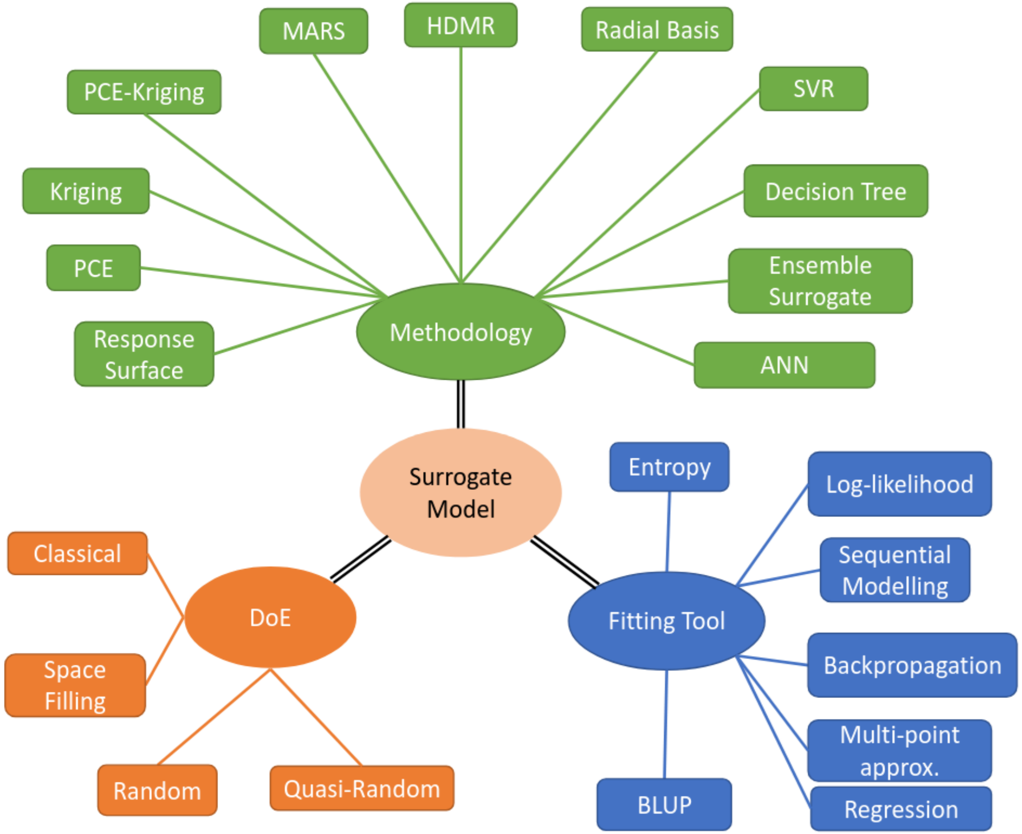
Thanks!

Definitions

- **Training/fitting**: In classic terms, training or fitting in machine learning is the process of finding the best mathematical function or model that matches (or "fits") a set of data points. Just like in classical curve fitting (e.g., fitting a line to a scatter plot), the goal is to adjust the model's parameters so it accurately captures the underlying patterns in the data
- **Bayesian optimization**: is a method for finding the best solution to a problem when evaluating each option is expensive or time-consuming. It uses a probabilistic model (like a Gaussian process) to predict the outcomes of different choices and focuses on testing the most promising ones to find the optimal solution efficiently.
- **Dataset:** is a collection of data, usually organized in a structured format like tables or files, that is used for analysis, training machine learning models, or drawing insights. It typically includes rows representing examples (like people, images, or events) and columns representing features or attributes (like age, color, or time).
- **Multi-fidelity model**: it combines information from multiple sources with varying levels of accuracy and cost to make predictions or solve problems more efficiently. It uses low-fidelity models (cheaper, less accurate) and high-fidelity models (more expensive, more accurate) together to balance cost and accuracy.
- **Physics-Informed Neural Network (PINN)**: is a type of neural network that incorporates the laws of physics (like equations or constraints) into its training process. This helps it solve problems more accurately and efficiently by combining data-driven learning with physical principles.

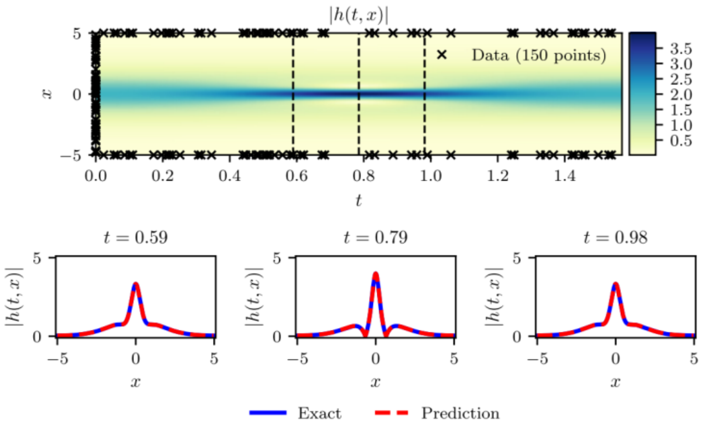
Common methods

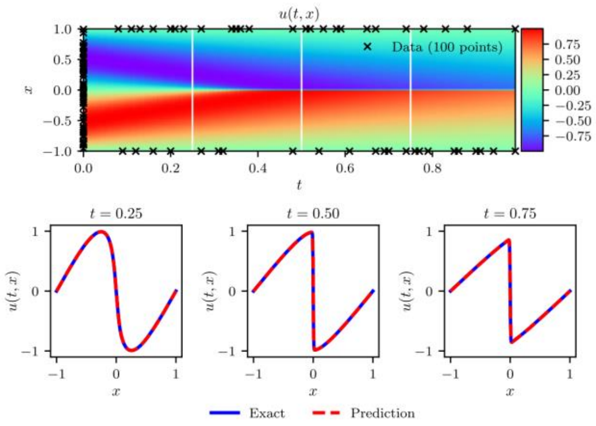
- ➔ Large possibilities on the choice of models, DoEs, fitting strategies…
- ➔ There is no solution that fits all problems!



Physics Informed Neural Networks

- ➔ First proposed to solve nonlinear PDE [10] (all plots from [10])
- ➔ Basically using boundary and initial conditions values, NN can interpolate the whole system dynamics "knowing" the PDE that describe the system
  - ◆ At the same time though, one can just use a physics loss term…it doesn't have to be a PDE system

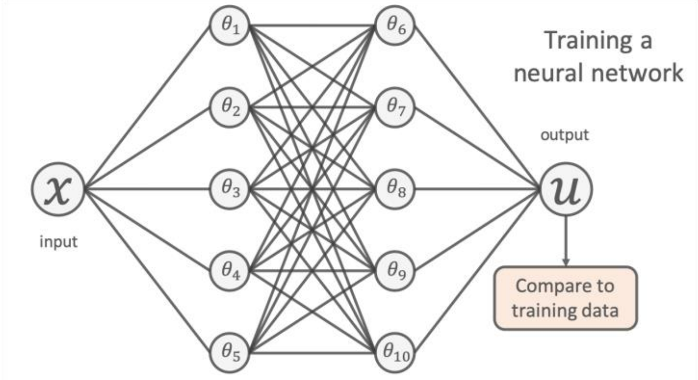




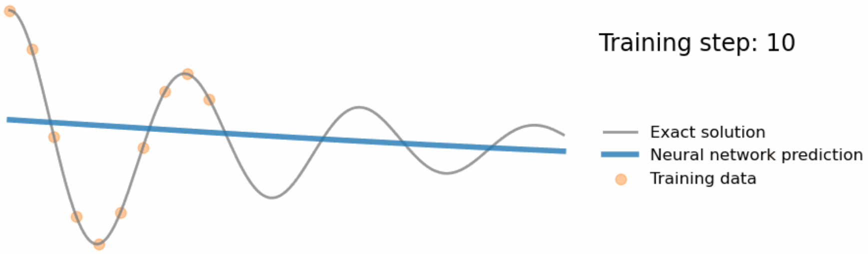
Physics Informed Neural Networks

➔ DNN cannot extrapolate beyond the training domain...which is exactly what we would expect from interpolation function

min(Loss) => Loss = Mean(data - prediction)^2



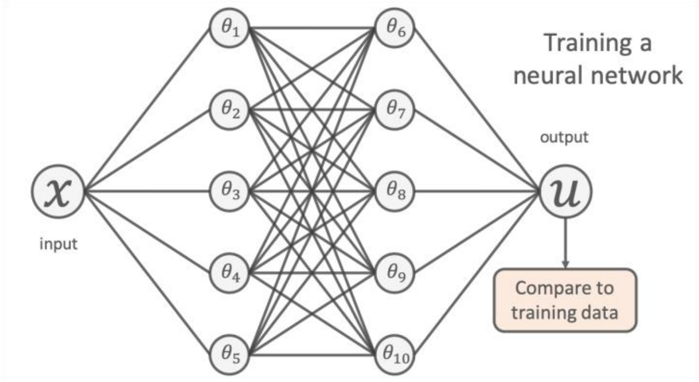
Source: [8]



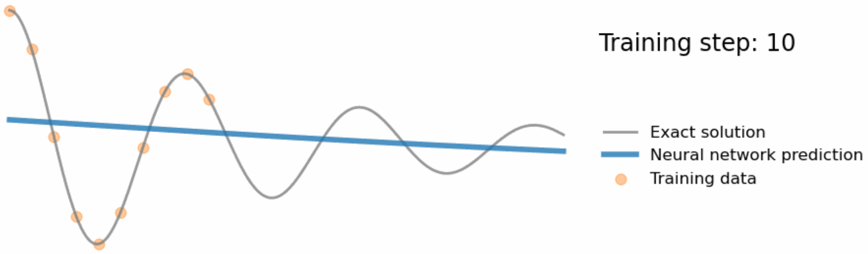
Physics Informed Neural Networks

➔ DNN cannot extrapolate beyond the training domain...which is exactly what we would expect from interpolation function

$$\mathcal{L} = \sum_{i}^{N} (u(x_i) - \hat{u}(x_i, \theta))^2$$



Source: [8]



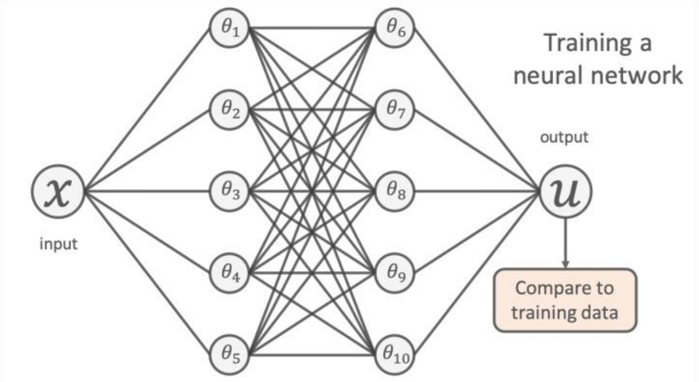
Physics Informed Neural Networks

➔ DNN cannot extrapolate beyond the training domain...which is exactly what we would expect from interpolation function

$$\mathcal{L} = \sum_{i}^{N} (u(x_i) - \hat{u}(x_i, \theta))^2$$

➔ Go beyond data domain => more information needed:

min(Loss) => Loss = Mean(data - prediction)^2
+ Additional_info(prediction)

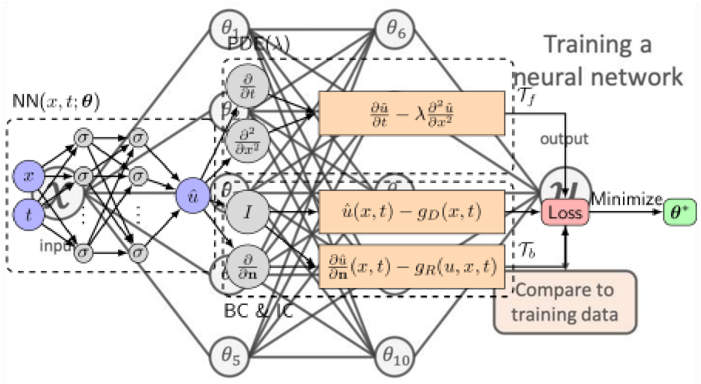


Source: [8]

Physics Informed Neural Networks

- ➔ DNN cannot extrapolate beyond the training domain…which is exactly what we would expect from interpolation function

$$\mathcal{L} = \sum_{i}^{N}(u(x_i) - \hat{u}(x_i, \theta))^2$$



Source: [8]

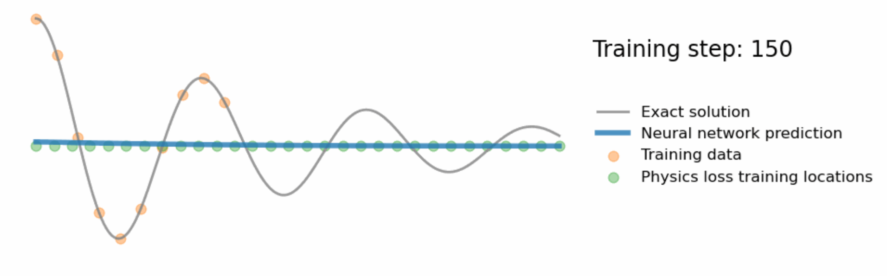
- ➔ Go beyond data domain => more information needed:

min(Loss) => Loss = Mean(data - prediction)^2
+ Additional_info(prediction)

$$\mathcal{L}_2 = 1/M \sum_{j}^{M} \left(\frac{\partial^2 \hat{u}}{\partial x^2} - \frac{\partial \hat{u}}{\partial t}\right)^2$$

$$\mathcal{L}_3 = \hat{u}(x, t=0) - f(x)$$

$$\mathcal{L}_4 = \hat{u}(x=0, t) - u_0$$



$$\mathcal{L}_{tot} = \alpha\mathcal{L}_1 + \beta\mathcal{L}_2 + \gamma\mathcal{L}_3 + \eta\mathcal{L}_4$$