UNIVERSITY OF CAMBRIDGE

Lecture 2 – Simulation

Machine Learning and the Physical World

Nicolas Durrande (Monumo) – 15th of October 2025

Lecture outline

Real world data

Simulation

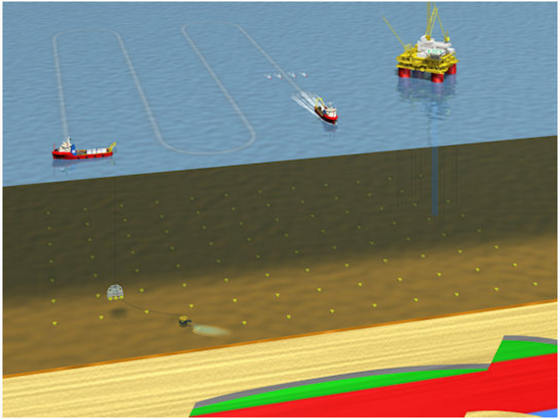
Same model, different numerics

What about ML?

Wrap up

Real world data

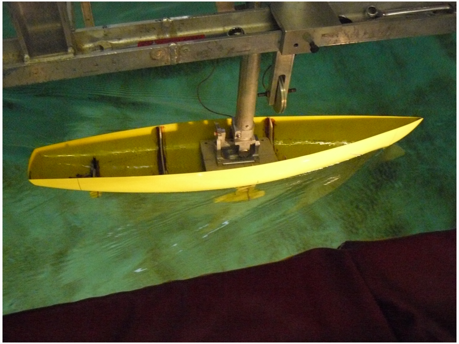
Example 1: mining exploration



Data collection

Example 2: Destructive testing

Example 3: Prototyping of a boat shape



In a **supervised learning** setting, data comes as pairs $\{x_i, y_i\}_{i=1}^n$.

1. Mining: x is the location, y is the oil quantity.
2. Crash test: x is the car design, y is the impact score.
3. Boat shape: x is the hull design, y is the drag force.

In the examples above:

- y is costly to obtain (weeks to months) $\Rightarrow$ n is small.
- x can be high-dimensional (10s to 100s).
- y is noisy (measurement errors, variability of the physical system).

Challenges in the low data regime:

- Computing integrals

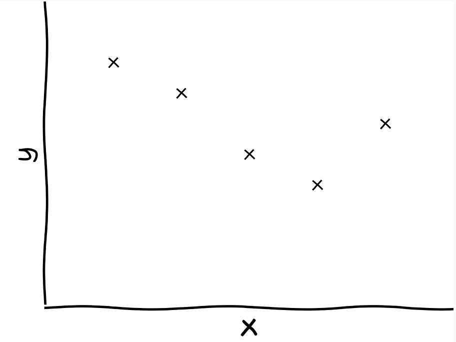
  What is the reservoir oil content?

- Uncertainty Propagation

  What is the impact score distribution?

- Optimisation

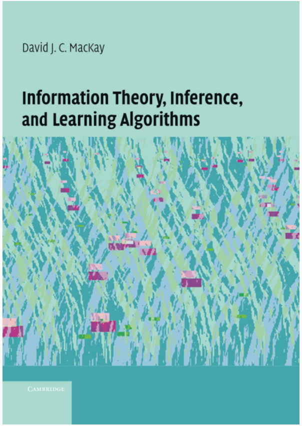
  What is the best hull design?



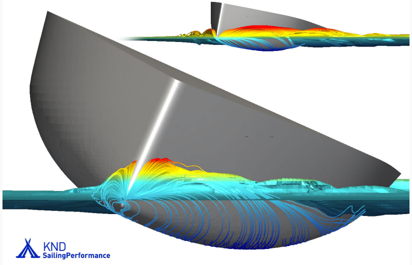
$$Data + Model \xrightarrow{\text{compute}} Prediction$$

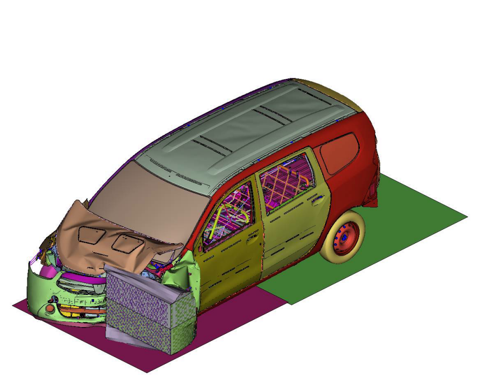
"You cannot do inference without making assumptions"

– David J. C. MacKay



Numerical experiments are an invaluable way to make predictions.





Simulation

The principle behind simulation is to introduce a **mathematical model** of a physical phenomenon, and to use **computation power** to solve it.

Two extremely common types of models are:

- Physics models based on **partial differential equations** (PDEs) (e.g. fluid dynamics, elasticity, electromagnetism).
- Discrete event models: The physical system is described by discrete events (e.g. queuing systems, network traffic).

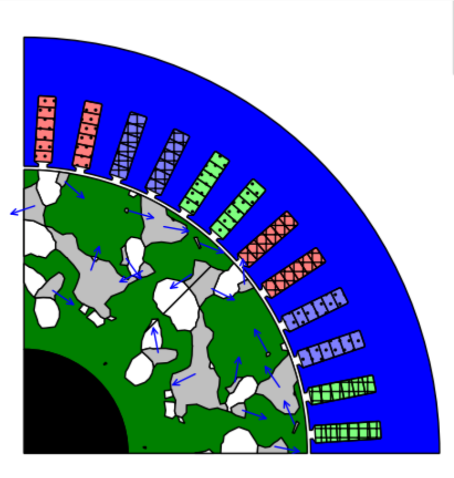
There is often a trade-off between the **accuracy** of the model and its **computational cost**.

Case study: electric motor design

What is the torque produced by this motor?

Is it mechanically safe?

How hot does it get?



Maxwell's equations (magnetostatic)

$$\nabla \times \nu \mathbf{B} = \mathbf{J}, \qquad \nabla \cdot \mathbf{B} = 0$$

The divergence-free condition can by guaranteed by working with the vector potential $\mathbf{B} = \nabla \times \mathbf{A}$ which yields:

$$\nabla \times \left(\nu \nabla \times \mathbf{A}\right) = \mathbf{J}$$

In a 2D problem only the z component of $\mathbf{A}$ and $\mathbf{J}$ is non-zero, which yields:

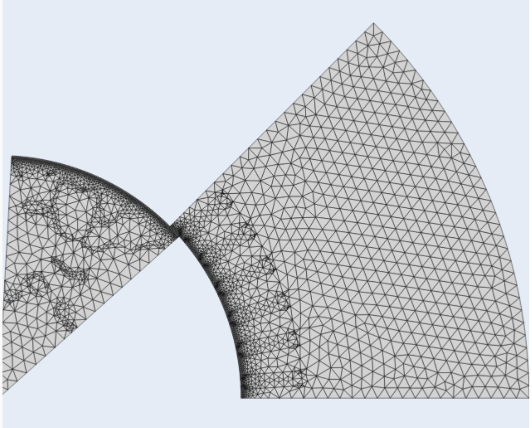
$$-\nabla \cdot \left(\nu(x) \nabla a(x)\right) = j(x)$$

Here, the **data** is the reluctivity $\nu(x)$ and the current density $j(x)$.

This PDE does not have an analytical solution for non trivial geometries

$\Rightarrow$ **Numerical methods** are required to obtain approximate solutions.

The **Finite Element Method (FEM or FEA)** is a popular choice for solving PDEs. It works by discretizing the domain into small elements to transform the PDE into a system of algebraic equations.



FEM (in one slide!)

Given that $-\nabla\cdot\big(\nu(x)\,\nabla a(x)\big) = j(x)$ holds for all x in Ω, then for any test function v:

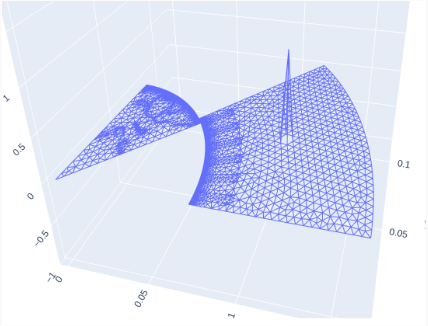
$$\int_\Omega -\nabla\cdot(\nu\,\nabla a)v\,\mathrm{d}x = \int_\Omega j(x)\,v(x)\,\mathrm{d}x.$$

A basis comes naturally with meshing Ω. We use it as test functions and to represent a:

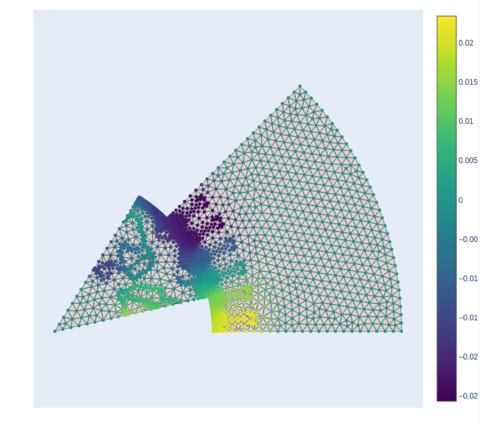
$$v(x) = \phi_i(x), \quad i = 1,\ldots,N$$

$$a(x) = \sum_{i=1}^{N} a_i\phi_i(x).$$

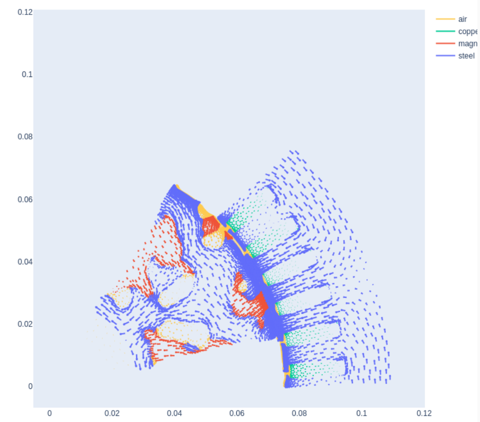
This makes the integrals tractable and leads to a linear system $Ka = F$ which is solved numerically.



Computing $a = K^{-1}F$ for a mesh with 3000 nodes takes $\sim$1s!



A-field solution



B-field

Case study conclusion

Computing the performance metrics of the motor requires $\sim$900 FEM solves:

- Different rotor angles: 30 positions
- Different current excitations: 30 configurations
- Other physics (mechanical, thermal): 1-2 solves each

All in one, evaluating the performance of one motor design takes $\sim$10 minutes.

As a comparison, building a physical prototype takes weeks to months!

Case study conclusion

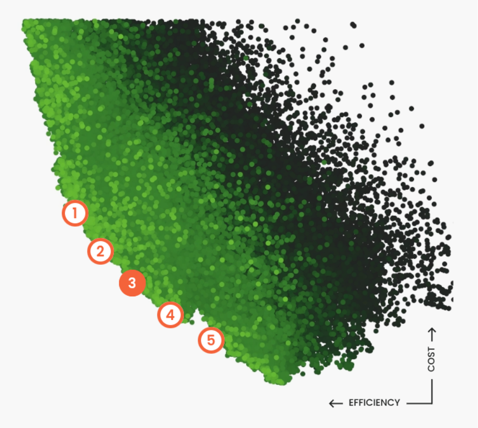
These simulations are used in an optimisation loop, to find the best motors.

Multiple objectives:

- Minimise material cost
- Minimise losses over drive cycle

Many constraints:

- Mechanical stress
- Demagnetisation
- Max temperature, etc.



Genetic Algorithms (NSGA-II) are typically used for such optimisation problems.

Where are my uncertainties?

Error in Data

- Material properties (permeability $\nu(x)$)
- Operating conditions (current density $j(x)$)
- Manufacturing tolerances (geometry)

Error in Model Simplifying assumptions (2D vs 3D, steady-state vs transient)

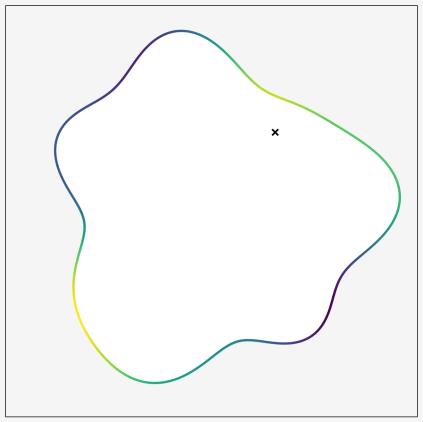
Computation Error Mesh resolution, numerical errors, solver tolerances, etc.

Expect a 10% discrepancy between simulation and test bed measurements!

Same model, different numerics

Steady state heat equation (Laplace) for a bounded domain Ω with Dirichlet boundary conditions:

$$\begin{aligned} \Delta u(x) &= 0 && \text{for } x \in \Omega, \\ u(x) &= g(x) && \text{for } x \in \partial\Omega. \end{aligned}$$



What is the temperature at x?

The *same solution* can be computed using several different numerical paradigms:

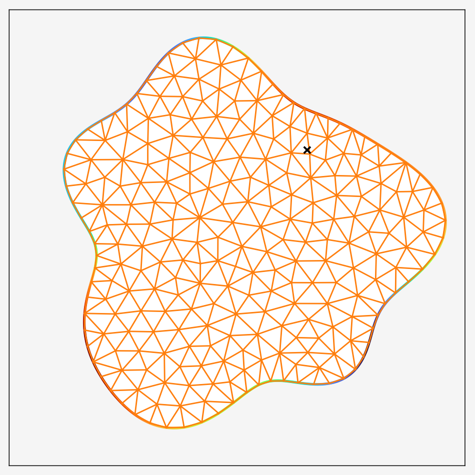
- **Finite Element Method (FEM)** on a mesh of Ω.
- **Probabilistic / Brownian motion Monte Carlo (MC)** via Feynman–Kac.
- **Boundary Element Method (BEM)** on $\partial\Omega$.
- **Finite Difference Method (FDM)** on a grid covering Ω.

We will focus on the first two, which have complementary strengths and weaknesses…

FEM solution

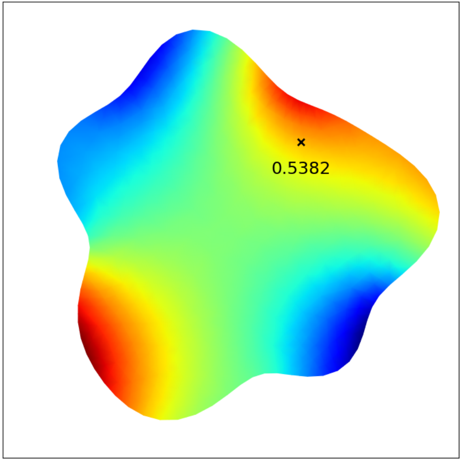
We follow the standard FEM workflow:

1. Generate a mesh of Ω.
2. Assemble the linear system $\mathbf{K}\,\mathbf{u} = \mathbf{F}$.
3. Solve the linear system.
4. Evaluate $u(x)$ at desired points using linear interpolation.

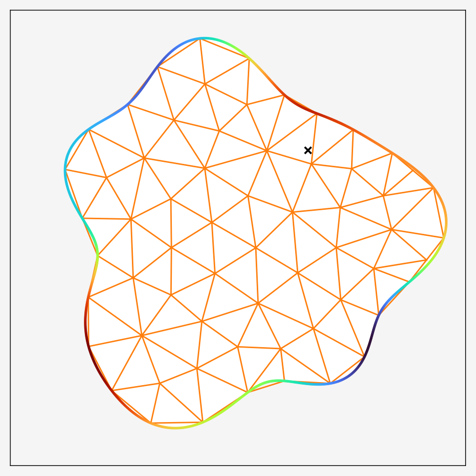


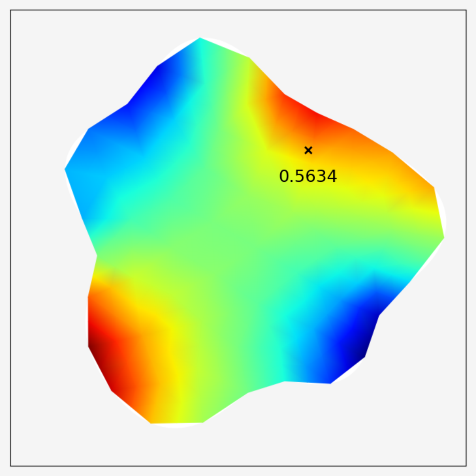
We follow the standard FEM workflow:

1. Generate a mesh of Ω.
2. Assemble the linear system $\mathbf{K}\,\mathbf{u} = \mathbf{F}$.
3. Solve the linear system.
4. Evaluate $u(x)$ at desired points using linear interpolation.



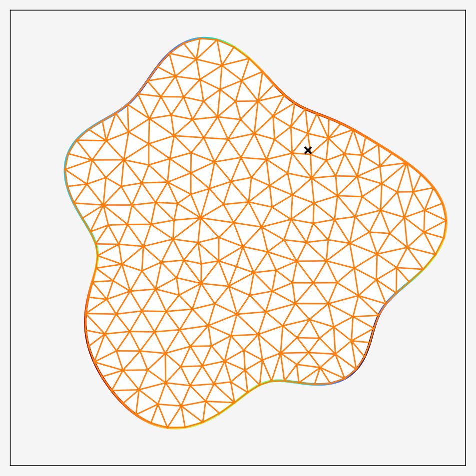
Influence of the mesh size *h* on the solution quality:

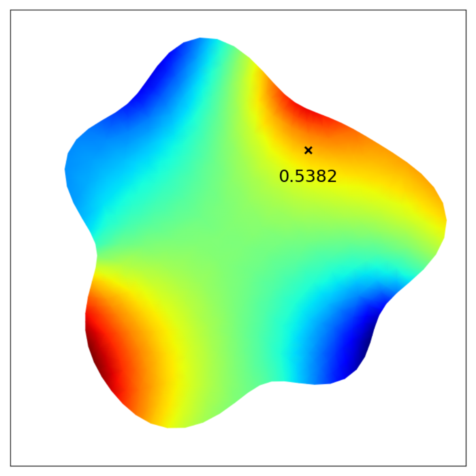




FEM solution

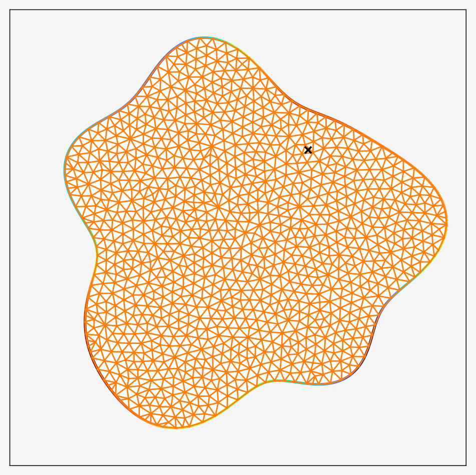
Influence of the mesh size *h* on the solution quality:

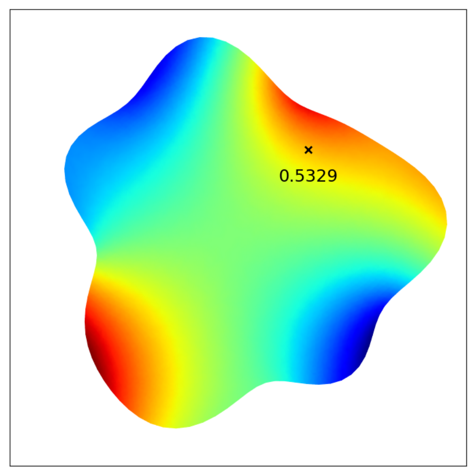




FEM solution

Influence of the mesh size h on the solution quality:





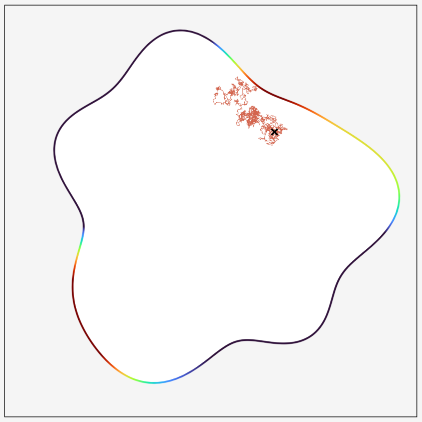
Monte Carlo solver

Let B_s be Brownian motion started at x, and let $\tau = \inf\{s \geq 0 : B_s \in \partial\Omega\}$ be the exit time from Ω. Then the heat solution equation can be written as:

$$u(x) = \mathbb{E}\Big[g\big(B_\tau\big)\Big]$$

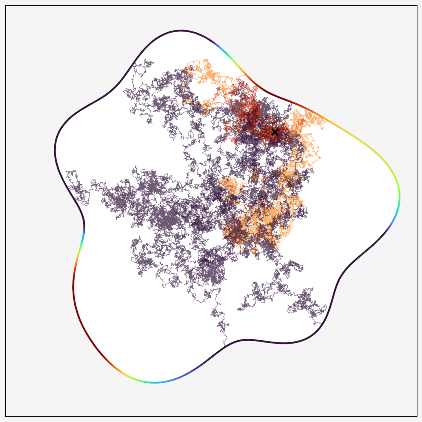
Key idea: estimate $u(x)$ at selected points by averaging independent Brownian paths until they hit $\partial\Omega$.

Example with one Brownian path:



10 paths:

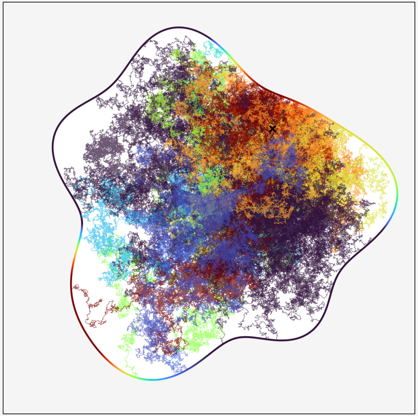
- $u(x) \approx 0.21$
- standard deviation: 0.20



Monte Carlo solver

100 paths:

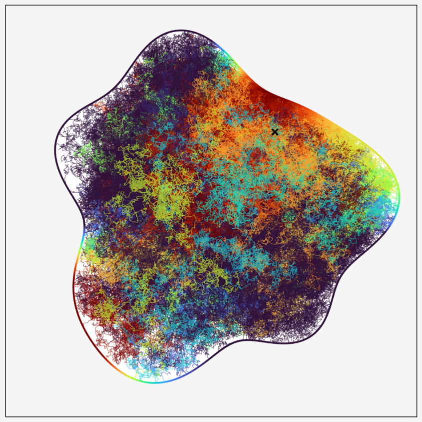
- $u(x) \approx 0.47$
- standard deviation: 0.06



Monte Carlo solver

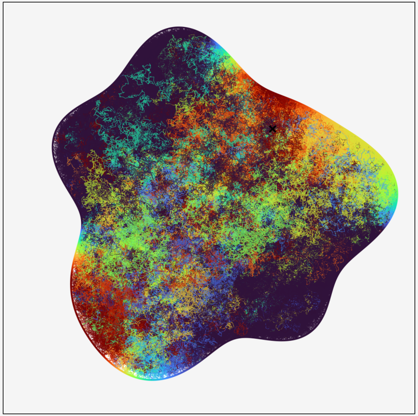
1000 paths:

- $u(x) \approx 0.55$
- standard deviation: 0.02



10,000 paths:

- $u(x) \approx 0.53$
- standard deviation: 0.00



FEM solver

- Computes the solution ”everywhere”.

  Consider BEM if this isn’t desirable on the problem at hand!

- Makes error quantification difficult!

Monte Carlo solver

- Computes the solution at selected points.
- Error quantification is straightforward!
- Convergence is slow (but dimension-independent).

  Can be accelerated with Walk on Spheres!

What about ML?

What about Machine Learning?

Same as before:

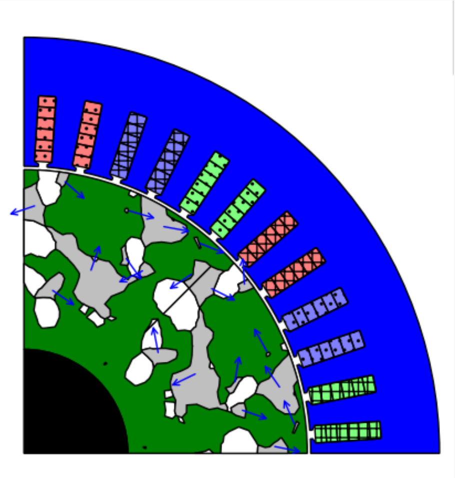
$$Data + Model \xrightarrow{\text{compute}} Prediction$$

just pushed further...

ML models are sometimes referred to as **surrogate models**, **meta-models** or **emulators** in the simulation community.

Case study: electric motor design

Let's go back to our case study...



Simulation pipeline

At a high level, the simulation pipeline is

ML to replace Simulation pipeline

From a ML perspective, it is tempting to collect data and to replace it with

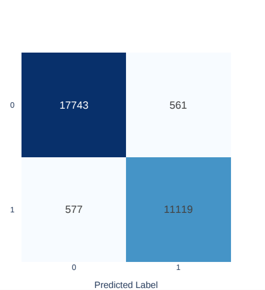
However... it is extremely difficult to collect data that is balanced and diverse!

- If using random sample, virtually no motor satisfies all the constraints
- If using an optimisation algorithm to collect data, the dataset severely lacks diversity

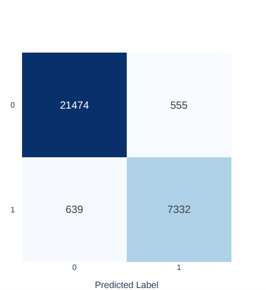
End-to-end classifier accuracy

Even if mixing both data collections scheme, class imbalance remains a problem!

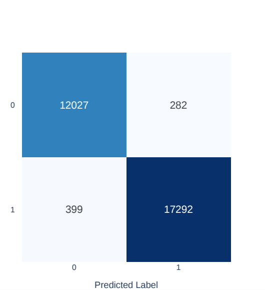
Example Consider an optim problem with three emag constraints. Using 1M simulations, we can train 3 classifiers and obtain good prediction accuracies:



confusion torque



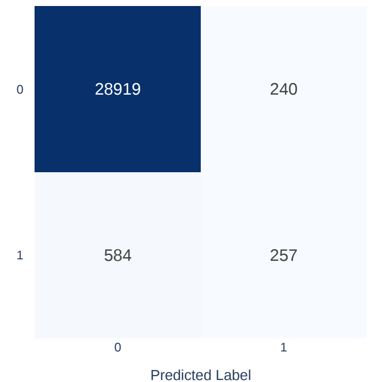
confusion torque ripple

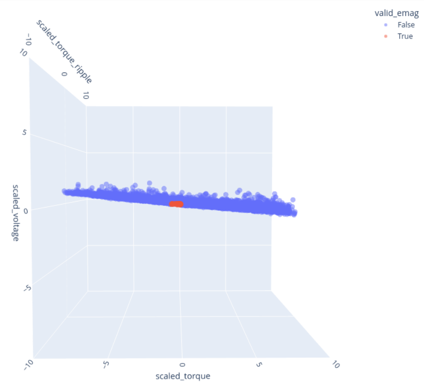


confusion voltage

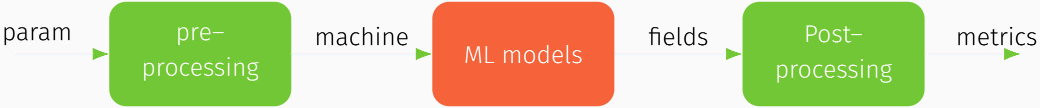
Example (continued)

However combining the predictions of these three classifiers results in very poor predictions





To solve the data imbalance issue, we considered the following approach



The ML models can be seen as drop-in replacement for FEM solvers, they take triangular meshes as input, and return field values at vertices and cells.

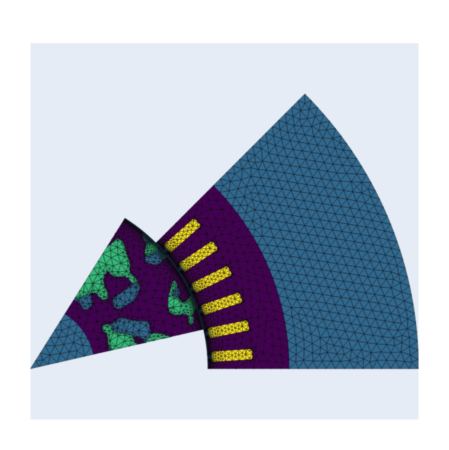
Modelling task

Models

- Operate on mesh/graphs
- Predict directly field values

Benefits

- No data imbalance
- Same granularity as FEM
- Gradients
- Simpler task?



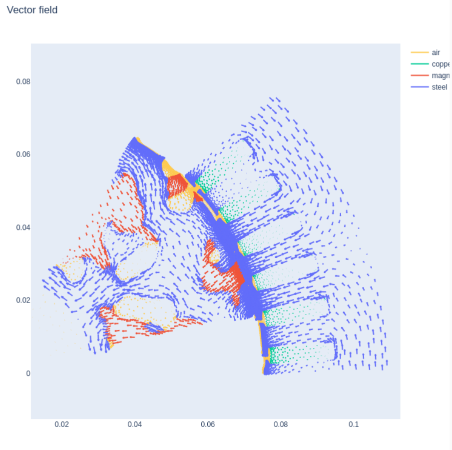
Machine represented as a mesh

Models

- Operate on mesh/graphs
- Predict directly field values

Benefits

- No data imbalance
- Same granularity as FEM
- Gradients
- Simpler task?



Outputs: A- and B-field

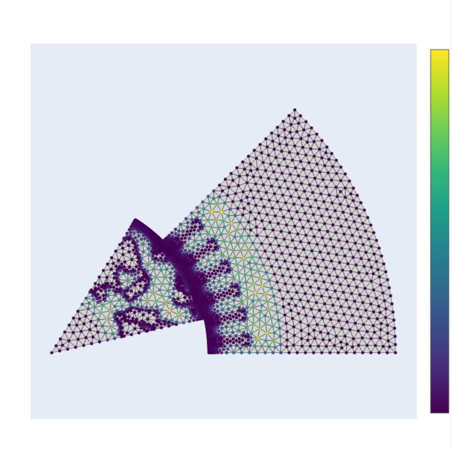
We use a simplified Graphormer architecture, with mesh vertices as inputs.

Input features

- Vertex location (Cartesian + polar, 4 features)
- Materials at vertex (4 features)
- Magnet polarisation (2 features)
- Current (1 feature)

Output features

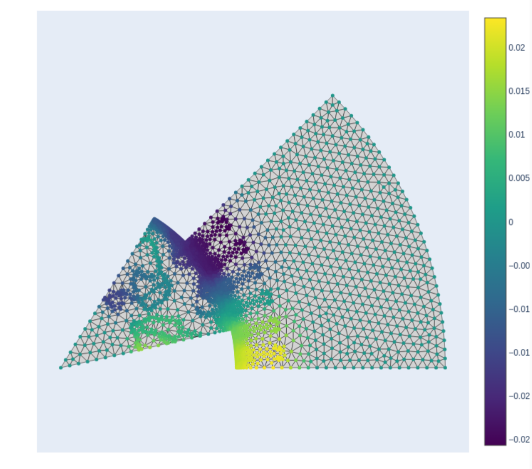
- A-field at mesh vertices
- B-field at mesh faces (curl of A)



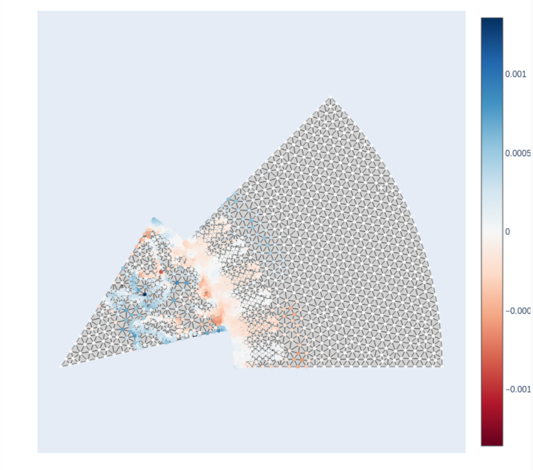
Input: Amount of steel at vertices

Model accuracy: A-field

GNN predictions

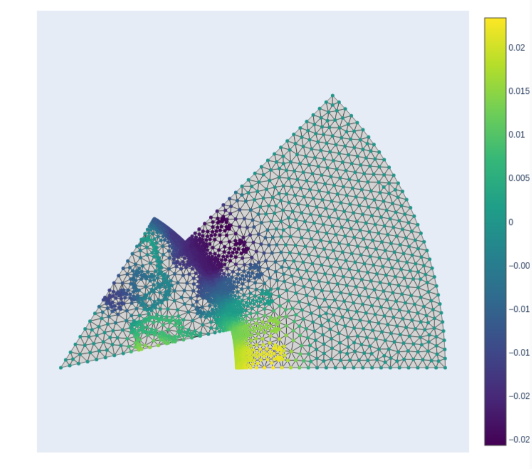


Error (actual - pred)

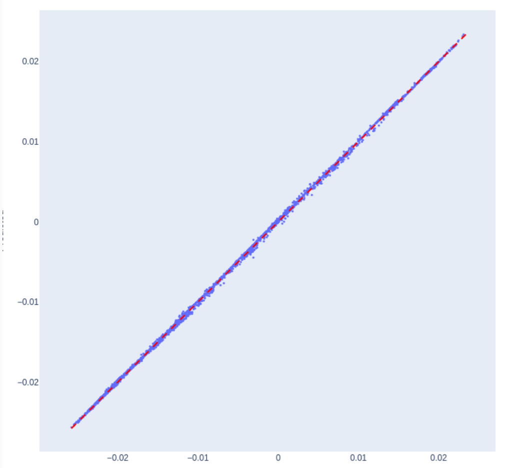


Model accuracy: A-field

GNN predictions



Pred vs True

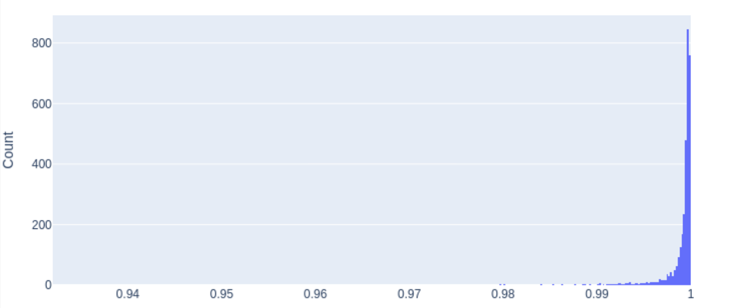


Model accuracy: A-field

We can now look at accuracy over the validation test (3k datapoints)

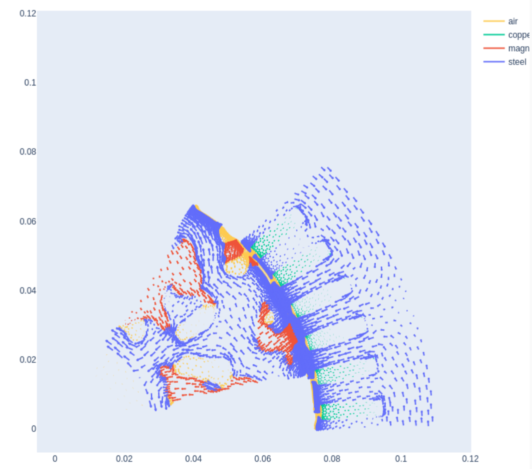
Key metrics

- Mean $R^2 = 0.9988$
- Median $R^2 = 0.9996$

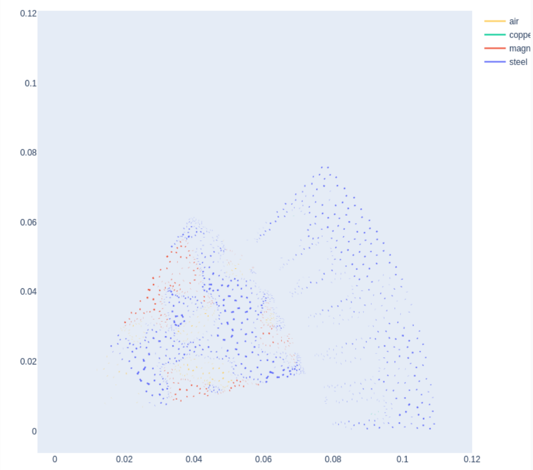


Model accuracy: B-field

GNN predictions

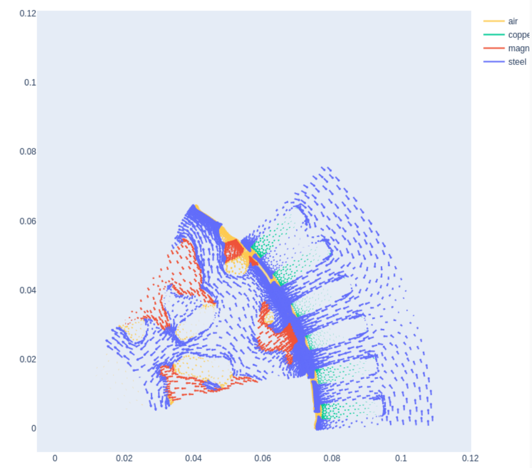


Error (actual - pred)

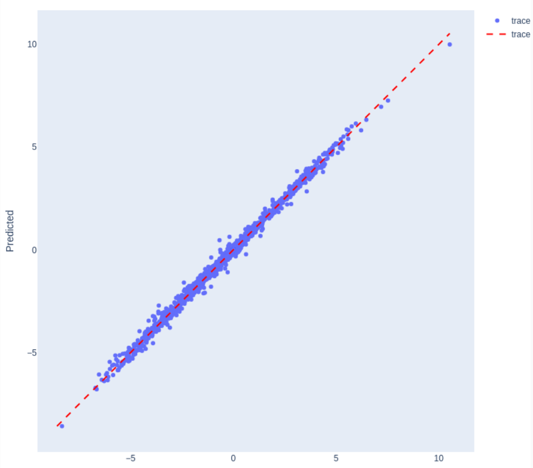


Model accuracy: B-field

GNN predictions

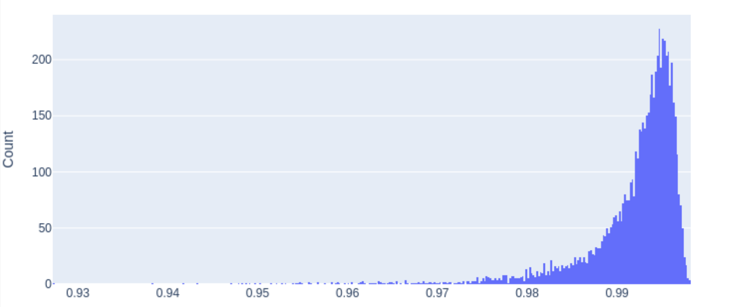


Pred vs True



Model accuracy: B-field

R^2 distribution over the validation test



Key metrics

- Mean $R^2 = 0.9918$
- Median $R^2 = 0.9936$

Wrap up

Real world data is typically extremely costly to obtain. It often includes noise or confounding factors.

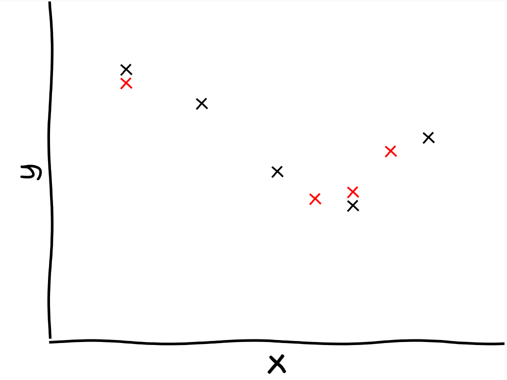
Simulation is a great alternative to real world experiments. More data can be collected, but search spaces are still too big to be explored exhaustively!

Further topics

Can ML models benefit from physical knowledge?

$\Rightarrow$ **Physics-informed Neural Networks**

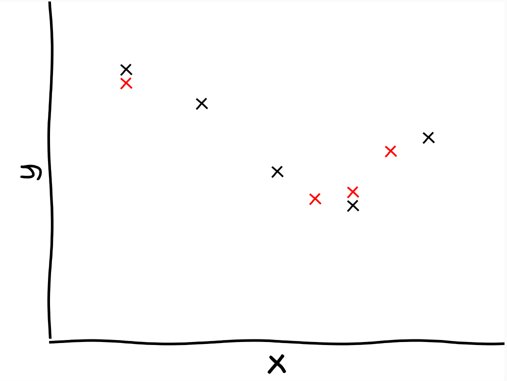
$\Rightarrow$ **Kernels encoding PDE knowledge**



Further topics

Can ML models help to combine real world data and simulation data?

⇒ **Multi-fidelity models**



What are design space points I should simulate?

$\Rightarrow$ Design of Experiments