Efficient adaptive meshing
in Finite Element Methods using Graph Neural Networks

James Rowbottom*[1], Georg Maierhofer*[1], Teo Deveney[2], Eike Hermann Müller[2], Alberto Paganini[3], Katharina Schratz[4], Pietro Liò[1], Carola-Bibiane Schönlieb[1], and Chris Budd[2]

*joint first-author (gam37@cam.ac.uk); [1]University of Cambridge, [2]University of Bath, [3] University of Leicester, [4]Sorbonne Université

Finite Element Methods (FEM): Powering Modern Simulation

- **Most widely used numerical methods** for solving PDEs in science and engineering;

$$\mathcal{F}(u_t, u, \nabla u, \nabla^2 u, \dots) = f.$$

- **Reliable simulation enabling** "certification by analysis";
- Global FEM market forecast: **£10 billion by 2033**;
- 25%-40% of development cost in new engineering products

  $\rightarrow$ **efficiency is paramount** for accelerating innovation.

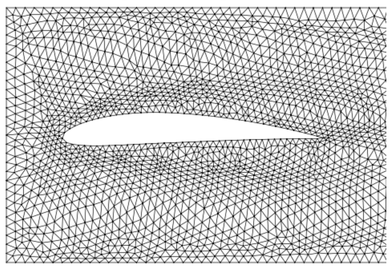


Figure 1: Example of G-Adaptive mesh.

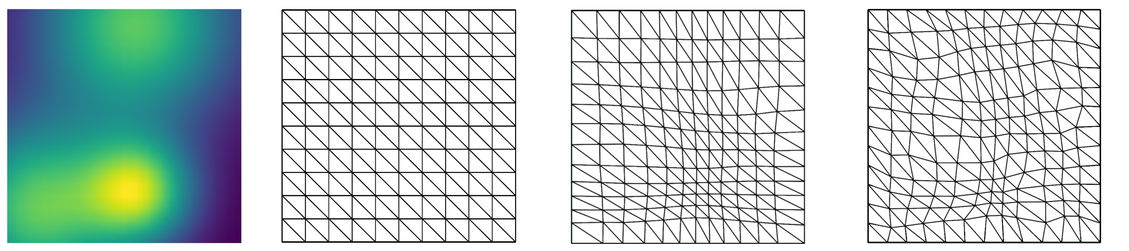
It all starts with the mesh...

Central ingredient: Solution quality depends on mesh $\mathcal{Z}$ and corresponding FEM basis space.

Goal in relocation-based (r-adaptive) meshing is to minimise the solution error

$$\min_{|\mathcal{Z}|=N_x} \mathcal{E}(\mathcal{Z}, U_{\mathcal{Z}}), \quad \text{where} \quad \mathcal{E}(\mathcal{Z}, U_{\mathcal{Z}}) := \|U_{\mathcal{Z}} - u\|_{L^2(\Omega)}.$$

The **key evaluation metrics** in this problem are **mesh relocation time** and **FEM error reduction**.



Solution field
FEM cost: 251ms

Standard mesh

28.74% Error reduction
(Conventional - **5183ms**)

39.99% Error reduction
(G-Adaptivity - **120ms!**)

Figure 2: Performance comparison of conventional and modern relocation techniques.

G-Adaptivity: efficient AI-based mesh relocation

Our **novel approach** achieves $25\times$ speed-up and significantly lower error.

Three main **G-adaptivity** components:

i. A stable, easy-to-train and low-cost **diffusion based graph neural network diffformer**;

ii. Training with **a differentiable FEM solver** for direct optimisation based on the FEM error;

iii. An **Equidistribution Regulariser**.

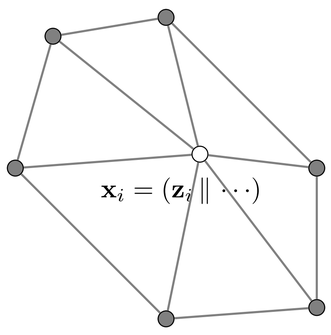
Bridging Geometric Deep Learning and Finite Elements

Represent mesh as graph:

- Mesh with points $\mathcal{Z} = \{\mathbf{z}_i\}_{i=1}^N$ induces a graph $(\mathcal{V}, \mathcal{E})$:

$$\mathcal{V} = \mathcal{Z}, \quad \mathcal{E} = \{(\mathbf{z}_i, \mathbf{z}_j) \mid \exists \Delta \in \text{Mesh}, \mathbf{z}_i, \mathbf{z}_j \in \Delta\}$$

- Node features $\mathbf{x}_j$ contain coordinates, source/solution information.



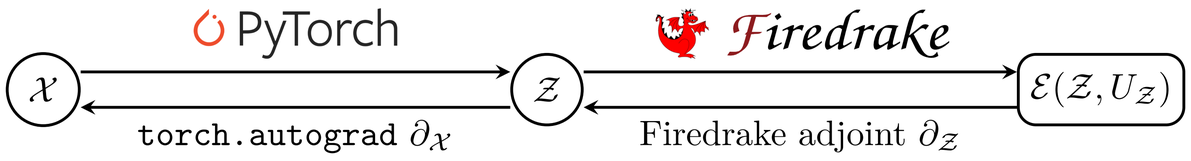
Deformation mechanism:

- Update features on each graph vertex:

$$\mathbf{x}_i^{k+1} = \Phi\left(\mathbf{x}_j^k, (i,j) \in \mathcal{E}\right)$$

Training mechanism:

- Interface between **advanced Python frameworks**: machine learning and FEM.

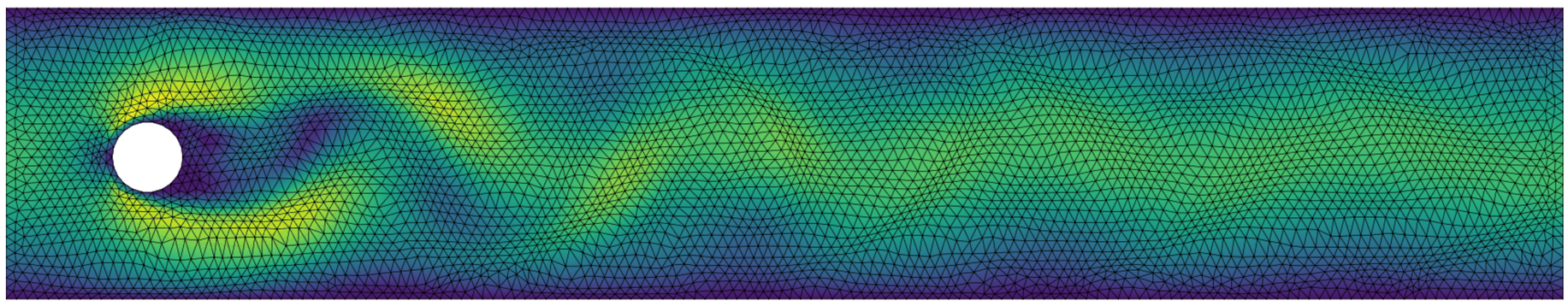


AI-Driven Adaptive Meshing for Fluid Flows

The model:

$$\frac{\partial \mathbf{u}}{\partial t} + (\mathbf{u} \cdot \nabla)\mathbf{u} - \nu \nabla^2 \mathbf{u} + \nabla p = \mathbf{f}.$$

Navier–Stokes equation		
Model	Error Red. (%)	Time (ms)
Conventional (MA)	NA	-
Prior-AI	1.34 $\pm$ 0.57	44
G-Adapt	**26.36±1.37**	49



Insights: Delivers $\sim$26% error reduction at neural-network speed with negligible overhead, making adaptive computational fluid dynamics (CFD) viable in practice, even when conventional methods fail.

The diffformer architecture

Graph Neural Diffusion: At each layer, node positions evolve via:

$$\dot{\mathcal{Z}}(\tau) = (\mathbf{A}_\theta(\mathbf{X}^k) - \mathbf{I})\mathcal{Z}(\tau), \quad \mathcal{Z}(0) = \mathcal{Z}^k,$$

which is solved to time τ_{end} to yield $\mathcal{Z}^{k+1}$.

Mesh Update Operator (Diffformer): The G-adaptive relocation operator is composed of multiple blocks, updating both geometry and features:

$$\begin{pmatrix} \mathcal{Z}^{k+1} \\ \mathbf{X}_\lambda^{k+1} \end{pmatrix} = \begin{pmatrix} \mathcal{Z}(\tau_{\text{end}}) \\ \sigma_\lambda\left(\mathbf{A}_\theta(\mathbf{X}^k)\mathbf{X}_\lambda^k \mathbf{W}_\lambda\right) \end{pmatrix}.$$

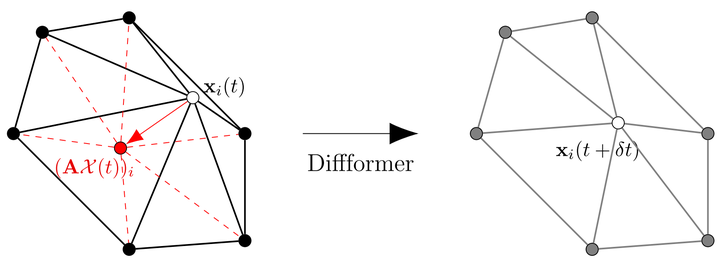


Figure 3: Graph diffusion pulls nodes in learned direction inside the convex hull of their neighbours.

Equidistribution Loss Regulariser

We introduce also the **equidistribution regulariser**:

$$\mathcal{L}_{\text{equi}}(\mathcal{Z}) = \sum_{\Delta^{(j)} \in \mathcal{T}} \left| \int_{\Delta^{(j)}} m(x)dx - \overline{m} \right|^2,$$

reducing **variation against the average monitor distribution** $\overline{m} = |\mathcal{T}|^{-1} \sum_{\Delta^{(j)} \in \mathcal{T}} \int_{\Delta^{(j)}} m(x)dx$.

Advantages of the G-Adaptivity approach

- **Guaranteed high-quality meshes:** no tangling and high mesh regularity assured;
- **Lightweight architecture:** easy to train and fast to evaluate (all experiments possible on laptop).

Tracking shock formation in Burgers' equation

The model:

$$\frac{\partial \mathbf{u}}{\partial t} + (\mathbf{u} \cdot \nabla)\mathbf{u} - \nu \nabla^2 \mathbf{u} = 0$$

- Trains for **optimal meshes for a chosen time-stepping method**.
- **Naturally incorporates remeshing frequency**.

Viscous Burgers' equation		
Model	Error Red. (%)	Time (ms)
Conventional (MA)	25.78 $\pm$ 0.00	18884
Prior-AI	−11.24 $\pm$ 2.52	314
G-Adapt	**27.17** $\pm$ **0.34**	717

Insights: Enables accurate shock-resolving simulation $\sim 25\times$ faster than classical moving-mesh methods.

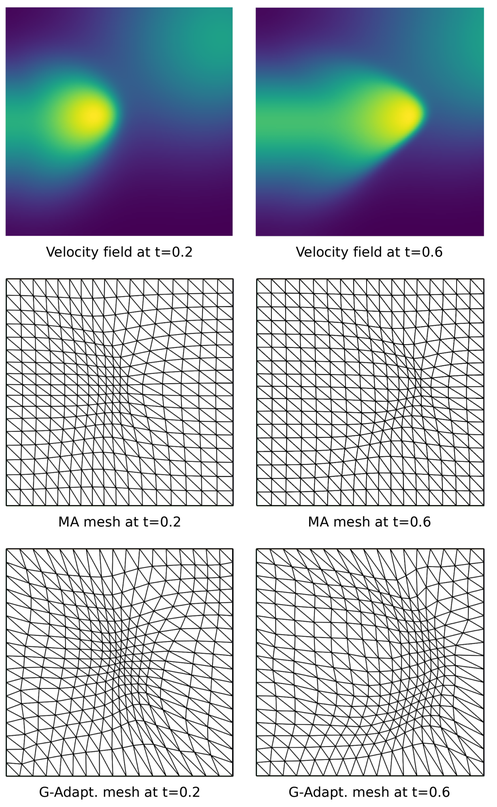


Figure 4: Accurate tracking of the shock front.

Impact and Future Directions

- Our **G-Adaptivity framework** performs mesh relocation at roughly **50% of FEM cost**, making adaptive meshing viable for real-world engineering (e.g. aerospace) simulations for the first time.
- **Fast, stable, and fully compatible with existing FEM infrastructure**, G-Adaptivity enables routine integration of relocation-based adaptive meshing into existing workflows.
- **Future Work:** Adaptive meshing on surfaces (Planet Earth) for weather and climate modelling.