Double Master's Degree Program

Università degli Studi di Padova

Dipartimento di Ingegneria Civile, Edile ed Ambientale

Mathematical Engineering

Mathematical Modelling for Engineering and Science

Universitat Politècnica de Catalunya

Departament d'Enginyeria Civil i Ambiental

Numerical Methods in Engineering

Final Master's Thesis

Enhancement of a structural optimization simulation tool using Machine Learning

Candidate:	Zamberlan Giorgio
UniPd Supervisor:	Larese Antonia
UniPd Co-supervisor:	Putti Mario
UPC Supervisor:	Giacomini Matteo

Academic year 2022/2023

Abstract

Additive manufacturing has opened unexplored possibilities in the fabrication of elastic structures not manufacturable via traditional moulding and machining processes. These new fabrication techniques found immediate synergies with topology optimization problems, allowing for the printing process of optimal shapes. Topology Optimization routines uses a wide variety of methodologies to obtain strongly defined configurations, and the one taken into consideration in this work used an anisotropic mesh adaptation technique to do so.

The focus will be centered around the results of a Topology Optimization routine in the context of a linear elastic problem to determine support structures under external loads.

The goal of this project is to show that Machine Learning algorithms could make a substantial contribution to reducing the computational cost of topological optimisation procedures, by generating quasi-optimal meshes as a starting point for the already existing pipelines. Said triangulations will need to reproduce the anisotropic elements to successfully reduce the computational burden of the the considered optimization procedure.

This work will make use of Graph Neural Networks instead of more traditional ones, aiming to show the inherent correlation between Finite Elements Methods and Graph Theory, possibly enticing further research in potential synergies between the two fields.

Contents

Acknowledgements

I would like to express my appreciation for my thesis tutors, Professor Matteo Giacomini at Universitat Politècnica de Catalunya and Professoressa Antonia Larese and Professor Mario Putti at Università degli Studi di Padova. I have deeply valued the time and effort that they have placed in me, effectively allowing me to be part of an international academic environment and guiding me through the difficulties of it.
A special recognition goes to my colleague Enrico Savona, which shared with me the endeavours of being one of the first students taking part in a novel Double Master's Degree program.
A special recognition goes to two people in particular, my father Marino and my girlfriend Alba Nor. They are my biggest supporters, and I could not have asked for better moral support, either from a distance, or here in Barcelona.

Chapter 1

Introduction

The main objective of a topology optimization procedure is to determine the layout of an object, by placing and removing material from the domain of interest, in order to obtain the best performances, measured according to a specific target objective. Applications of interest span a wide range of problems and fields, from aerodynamics [13], to electromagnetic [25] and structural design [1].

These optimization problems can be formalized as a PDE-constrained problem for a domain-dependent functional. The most prevalent industrial codes for topology optimization are developed around density-based methods, level-set and phase-field procedures [40]. The leading density-based approach is the Simplified Isotropic Material with Penalization (SIMP), which emphasizes the material properties through a penalization factor. The main drawback of this method, is its mesh dependency, which leads many solutions to tend towards a chequered pattern. Numerous solutions to this common issue have been developed, like filtering techniques [37] and penalization methods [42].
This work will use a modified version of the method proposed in [29], by combining a Reaction Diffusion Equation (RDE) update scheme [8] with a Double Well Potential (DWP) technique [39]. These methodologies are solved by means of continuous Finite Elements Method (FEM), effectively allowing to develop error estimation indicators and grid refinements techniques.

This combination of methods is synergised with an anisotropic mesh refinement procedure [2], based on a recovery-based error estimator [12], which allows to reduce the computational cost and time required to obtain the final optimized solution.

Although well established, these topology optimization pipelines are inherently complex and computationally demanding. The need to reduce computational effort has been a staple of the engineering world, but in recent years the growth in popularity of Machine Learning techniques, and in particular Neural Networks, gave a new impulse to research in these technologies. A prime example of the possible advantages of implementing Neural Network into pre-existing pipelines was image processing, which has been adopted into manufacturing processes [43] to improve productivity. A further step forward in Machine Learning was achieved with the introduction of Deep Learning. This new improved methodology immediately found itself in applications for complex problems that could not be tackled before, such as health monitoring [26], disease detection [10], ocean data inference via satellites [5] and active weed plants recognition during soil sowing in large scale agriculture [20].

Machine Learning undoubtedly changed significantly how many industries approach technological challenges. Even though it may seems like Neural Networks could potentially solve any problem, a lot of challenges have to be overcome before these tools can becomes useful and practical. Traditional Machine Learning techniques do not allow much flexibility on the dimension of their input, and they intrinsically require exhaustive training before becoming viable. In order to overcome these issues, another class of Neural Network grew in popularity in recent years, Graph Neural Networks. This novel approach to Machine Learning aims to reduce the computational strain of the training associated to traditional networks, by associating the input data with connections between single datapoints, effectively interpreting the inputs as graphs [6]. Moreover, these network have been designed with the goal of allowing a much a broader flexibility when it comes to input data [9].

Information propagation methodologies [22], helped Graph Neural Network in becoming a viable tool for complex problems such as inductive learning [14], neighborhood sampling and network scalability [15]. Recently the research in graph based learning produced very advanced and flexible architectures[24], potentially capable of tackling an incredible variety of tasks and problems. Employment of Graph Neural Networks for industrial equipment residual lifetime forecast [34], estimation for thermodynamics effects in industrial processes [18] and

anomaly detection [41] represents just a few of the possible usages of this new type of Machine Learning technologies as useful tools for real life industrial applications.

Nowadays most simulation tools used for industrial applications relies on FEM, which requires a triangulation of the domain under study to be implemented. Such discretization could be interpreted as a graph, and the connection between these methodologies and graph based learning is intuitive.
This project aims to apply Graph Neural Networks to Finite Elements Methods based simulations, to potentially reduce the computational demand of the latter thanks to the inherent correlation between the meshes used in FEM and the graph utilized by GNN. Such task is accomplished by developing a novel GNN architecture, which expands the capabilities of models present in the literature today [11]. The potential of this new model is tested on a well known problem, allowing to perform a numerical quantitative validation of its results before testing it inside the context of an optimization pipeline.

Chapter 2

Topology optimization of a linear elastic structure

This chapter introduces the optimal design problem of an elastic structure with minimum compliance and a volume restriction.

2.1 The elastic Problem

The elastic problem is studied under the hypothesis of linear elasticity. Let $\Omega \in \mathbb{R}^d$ where $d = 2, 3$ be a connected bounded domain. The deformation of Ω under a volume force $\boldsymbol{f}$ is described by the conservation of momentum law

$$-\nabla \cdot \boldsymbol{\sigma} = \boldsymbol{f} \text{ in } \Omega, \tag{2.1}$$

where $\boldsymbol{\sigma} = \boldsymbol{\sigma}^T$ is the Cauchy stress tensor. We now define the linearized strain tensor, under the hypothesis of small displacements [19], as

$$\boldsymbol{\varepsilon}(\boldsymbol{u}) = \frac{1}{2}\left(\nabla \boldsymbol{u} + \nabla \boldsymbol{u}^T\right). \tag{2.2}$$

Furthermore, we consider the linear relation between stress and strain known as Hook's law

$$\boldsymbol{\sigma} = \boldsymbol{\sigma}(\boldsymbol{u}) = \boldsymbol{C}\boldsymbol{\varepsilon}(\boldsymbol{u}), \tag{2.3}$$

where $\boldsymbol{C} = \boldsymbol{C}(\boldsymbol{x})$ is the fourth-order elastic tensor. Under the assumption of an homogeneous and isotropic material, the mechanical properties of the material are described by the Young's modulus E and the Poisson's ratio ν. To maintain a more compact notation, we define the two Lamé constants

$$\lambda = \frac{E\nu}{(1+\nu)(1-2\nu)}, \quad \mu = \frac{E}{2(1+\nu)}. \tag{2.4}$$

Equation 2.3 can we reformulated using these constants as

$$\boldsymbol{\sigma} = \boldsymbol{C}\boldsymbol{\varepsilon}(\boldsymbol{u}) = 2\mu\boldsymbol{\varepsilon}(\boldsymbol{u}) + \lambda \mathrm{tr}(\boldsymbol{\varepsilon}(\boldsymbol{u}))\mathbb{I}, \tag{2.5}$$

where $\mathbb{I}$ is the $d \times d$ identity matrix. Finally, is possible to fully describe the linear elastic model underlying the deformation of Ω under a volume force $\boldsymbol{f}$ as

$$\begin{cases} -\nabla \cdot \boldsymbol{\sigma}(\boldsymbol{u}) = \boldsymbol{f} \text{ in } \Omega \\ \boldsymbol{u} = \boldsymbol{0} \text{ on } \Gamma_D \\ \boldsymbol{\sigma}(\boldsymbol{u})\boldsymbol{n} = \boldsymbol{g} \text{ on } \Gamma_N \\ \boldsymbol{\sigma}(\boldsymbol{u})\boldsymbol{n} = \boldsymbol{0} \text{ on } \Gamma_F \end{cases} \tag{2.6}$$

In the above equation, the domain boundary $\partial\Omega$ is divided into three portions disjoint by pairs such that $\partial\Omega = \Gamma_D \cup \Gamma_N \cup \Gamma_F$. They represent respectively the Dirichlet, Neumann and traction free sections of the boundary.

To obtain the weak formulation of Equation 2.6, we consider the subspace:

$$V = \{\boldsymbol{v} \in [H^1(\Omega)]^d \mid \boldsymbol{v} = \boldsymbol{0} \text{ on } \Gamma_D\} \tag{2.7}$$

The weak form of the problem is obtained after multiplying the equation for a test function $v \in V$ and integrating over the whole domain. After using the above constitutive equations and assuming zero body forces, it is possible to express the weak form of the problem as: find $\boldsymbol{u} \in V$ such that

$$\int_\Omega (\lambda(\nabla \cdot \boldsymbol{u})(\nabla \cdot \boldsymbol{v}) + 2\mu\boldsymbol{\varepsilon}(\boldsymbol{v}) : \boldsymbol{\varepsilon}(\boldsymbol{u}))d\boldsymbol{x} = \int_{\Gamma_N} \boldsymbol{g} \cdot \boldsymbol{v}\ ds, \tag{2.8}$$

or alternatively

$$\int_\Omega \boldsymbol{C}\boldsymbol{\varepsilon}(\boldsymbol{u}) : \boldsymbol{\varepsilon}(\boldsymbol{u}))d\boldsymbol{x} = \int_{\Gamma_N} \boldsymbol{g} \cdot \boldsymbol{v}\ ds. \tag{2.9}$$

The problem is solved numerically using the continuous Galerkin method.

2.2 Minimization of the compliace under a volume constrain

In this section, the problem of the minimization of the compliance under a volume constraint is studied. Considering only a surface load, we define an energy functional as

$$\mathcal{J}(\Omega, \boldsymbol{u}) = \int_{\Gamma_N} \boldsymbol{g} \cdot \boldsymbol{u}\ ds, \tag{2.10}$$

where $\boldsymbol{u}$ is the solution of Equation 2.8. The compliace minimization problem is formulated as:

$$\min_{\Omega \in \mathcal{U}_{ad}} \mathcal{J}(\Omega, \boldsymbol{u}) = \int_{\Gamma_N} \boldsymbol{g} \cdot \boldsymbol{u}\ ds \ \text{ s.t. } \begin{cases} -\nabla \cdot \boldsymbol{\sigma} = \boldsymbol{0} \text{ in } \Omega \\ \boldsymbol{u} = \boldsymbol{0} \text{ on } \Gamma_D \\ \boldsymbol{\sigma}(\boldsymbol{u})\boldsymbol{n} = \boldsymbol{g} \text{ on } \Gamma_N \\ \boldsymbol{\sigma}(\boldsymbol{u})\boldsymbol{n} = \boldsymbol{0} \text{ on } \Gamma_F \\ \displaystyle\int_\Omega 1\ d\boldsymbol{x} \leq V^\star \end{cases} \tag{2.11}$$

where $\mathcal{U}_{ad}$ and $V^\star$ represent the set of admissible domains in $\mathbb{R}^d$ and the maximum allowed volume.
The solid contained into the domain $\mathcal{D}$ will be described through a phase field $\phi(\boldsymbol{x})$, which will assume density values between 0 and 1, namely,

$$\begin{cases} \phi(\boldsymbol{x}) = 1 & \forall \boldsymbol{x} \in \Omega_1 \\ 0 < \phi(\boldsymbol{x}) < 1 & \forall \boldsymbol{x} \in \Omega_{int} \\ \phi(\boldsymbol{x}) = 0 & \forall \boldsymbol{x} \in \Omega_0 \end{cases} \tag{2.12}$$

where $\Omega_1 \cup \Omega_{int} \cup \Omega_0 = \mathcal{D}$, as depicted in Figure 2.1.

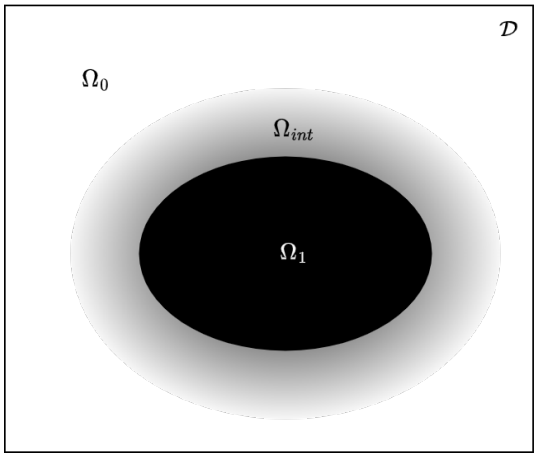


Figure 2.1: Visual representation of the solid via a Phase field

It is now possible to define the material properties through said phase field, using the SIMP method as

$$E(\phi) = E_{min} + (E_0 - E_{min})\phi^p, \tag{2.13}$$

where E_0 is the material Young's modulus and E_{min} is set to avoid singularities. The penalty parameter p is

set to 3 [3]. Likewise is possible to define the Lamé constants and the elastic tensor by means of ϕ:

$$\lambda(\phi) = \frac{\nu}{(1+\nu)(1-2\nu)}E(\phi), \quad \mu(\phi) = \frac{1}{2(1+\nu)}E(\phi), \tag{2.14}$$

$$\boldsymbol{C}(\phi) = \boldsymbol{C}_{min} + (\boldsymbol{C}_0 - \boldsymbol{C}_{min})\phi^p. \tag{2.15}$$

It is possible now to re-state the minimization problem as a function of ϕ, that is,

$$\min_{\phi} \ \mathcal{J}(\phi, \boldsymbol{u}_\phi) = \int_{\Gamma_N} \boldsymbol{g} \cdot \boldsymbol{u}_\phi \ ds \ \text{ s.t. } \begin{cases} -\nabla \cdot (\boldsymbol{C}(\phi)\boldsymbol{\varepsilon}(\boldsymbol{u}_\phi)) = \boldsymbol{0} \text{ in } \mathcal{D} \\ \boldsymbol{u}_\phi = \boldsymbol{0} \text{ on } \Gamma_D \\ (\boldsymbol{C}(\phi)\boldsymbol{\varepsilon}(\boldsymbol{u}_\phi))\boldsymbol{n} = \boldsymbol{g} \text{ on } \Gamma_N \\ (\boldsymbol{C}(\phi)\boldsymbol{\varepsilon}(\boldsymbol{u}_\phi))\boldsymbol{n} = \boldsymbol{0} \text{ on } \Gamma_F \\ G(\phi) = \frac{\int_{\mathcal{D}} \phi \ d\boldsymbol{x}}{|\mathcal{D}|} - V_{req} \leq 0 \end{cases} \tag{2.16}$$

In the above formulation $|\mathcal{D}|$ is the area of the total domain and $V_{req} = V^\star/|\mathcal{D}|$ is the volume fraction required.

2.3 The optimization strategy

The solution of the minimization problem requires the sensitivity, namely the Fréchet derivative of the objective function with respect to ϕ.[39]. The Lagrangian $\mathcal{L}$ accounting for the compliance and the state equation of the linear elasticity problem is given by

$$\mathcal{L}(\phi, \widetilde{\boldsymbol{u}}, \widetilde{\boldsymbol{p}}) = \mathcal{J}(\phi, \widetilde{\boldsymbol{u}}) + \int_{\mathcal{D}} (\boldsymbol{C}(\phi)\boldsymbol{\varepsilon}(\widetilde{\boldsymbol{u}})) : \boldsymbol{\varepsilon}(\widetilde{\boldsymbol{p}}) \ d\boldsymbol{x} - \int_{\Gamma_N} \boldsymbol{g} \cdot \widetilde{\boldsymbol{p}} \ ds \tag{2.17}$$

where $\widetilde{\boldsymbol{u}} \in V$ is associated with the solution of the elastic problem and $\widetilde{\boldsymbol{p}}$ is a Lagrange multiplier. The elastic problem is notoriously self-adjoint [38],allowing to directly compute the sensitivity as

$$\mathcal{J}'(\phi, \boldsymbol{u}) = -(\lambda'(\phi)(\nabla \cdot \boldsymbol{u})(\nabla \cdot \boldsymbol{u}) + 2\mu'(\phi)\boldsymbol{\varepsilon}(\boldsymbol{u}) : \boldsymbol{\varepsilon}(\boldsymbol{u})). \tag{2.18}$$

Once obtained the sensitivity, we can add the inequality constraint related to the volume restriction using the augmented Lagrangian [4]

$$\widetilde{\mathcal{L}}(\phi, q, r) = \eta\mathcal{J}(\phi, \boldsymbol{u}) + qG^+(\phi, q, r) + \frac{1}{2}r\left|G^+(\phi, q, r)\right|^2, \tag{2.19}$$

Where q is the Lagrange multiplier of the inequality constraint and r is the penalty parameter. Moreover

$$G^+(\phi, q, r) = \max\left(G(\phi), -\frac{q}{r}\right), \quad \eta = \frac{|\mathcal{D}|}{\|\mathcal{J}'(\phi, \boldsymbol{u})\|_{\mathcal{L}^2(\mathcal{D})}} \tag{2.20}$$

In the above equation, the term

$$G(\phi) = \frac{\int_{\mathcal{D}} \phi d\boldsymbol{x}}{|\mathcal{D}|} - V_{req}, \tag{2.21}$$

represents the difference between the volume represented by ϕ and the required volume V_{req}.

To solve the above problem, a gradient-based method is implemented exploiting the derived sensitivity. In particular an iterative strategy is devised to optimize the augmented Lagrangian function $\widetilde{\mathcal{L}}$ with respect to the variable ϕ, fixing q_k and r_k

$$\phi_{k+1} = \arg\min_{\phi} \widetilde{\mathcal{L}}(\phi, q_k, r_k) \ \text{ s.t. } \ \phi \in [0, 1] \tag{2.22}$$

Then the Lagrange multiplier q and the penalty coefficient r used in the augmented Lagrangian will be updated [4] as follows

$$\begin{cases} q_{k+1} = q_k + r_k \max\left(G(\phi), -\frac{q_k}{r_k}\right) \\ r_{k+1} = \gamma r_k, \quad \gamma > 1 \end{cases} \tag{2.23}$$

More precisely, to solve Equation 2.22 a combination of a Reaction Diffusion Equation [8] and a Double Well Potential [39] is employed.

We introduce a fictitious time t to describe the evolution of ϕ. The RDE used for the topology optimiza-

tion is defined as

$$\frac{\partial \phi}{\partial t} = \kappa \nabla^2 \phi - \left. \frac{\partial \widetilde{\mathcal{L}}}{\partial \phi} \right|_{t=t_1} \quad \text{in} \quad \mathcal{D} \times (t_1 \leq t \leq t_2), \tag{2.24}$$

with boundary conditions

$$\frac{\partial \phi}{\partial \boldsymbol{n}} = 0 \quad \text{on} \quad \partial \mathcal{D}. \tag{2.25}$$

The right hand side of Equation 2.24 is composed of a diffusion and a reaction term. The diffusion coefficient κ controls the complexity of the optimal shape and stabilizes the optimization, while the reaction is the derivative of the augmented Lagrangian with respect to ϕ and encapsulates the sensitivity with respect to the design variable ϕ, namely

$$\frac{\partial \widetilde{\mathcal{L}}}{\partial \phi}(\phi, q, r) = \eta \mathcal{J}'(\phi, \boldsymbol{u}) + \begin{cases} G'(\phi)(q + rG(\phi)) & \text{if} \quad G(\phi) \geq -q/r \\ 0 & \text{if} \quad G(\phi) < -q/r \end{cases} \tag{2.26}$$

where $G'(\phi) = |\mathcal{D}|$.

Given this formulation, is possible to derive the weak form of Equation 2.22. Using a Backward Euler method for the time derivative, and continuous FEM in space, the weak formulation results [2] is: for $n = 1, 2, ..$ find $\phi \in \mathcal{H}^1(\mathcal{D})$ such that

$$\int_{\mathcal{D}} \left[\frac{\phi_{n+1} - \phi_n}{\Delta t} \tilde{\phi} + \kappa \nabla \phi_{n+1} \cdot \nabla \tilde{\phi} + \frac{\partial \widetilde{\mathcal{L}}}{\partial \phi}(\phi_n, q_n, r_n) \tilde{\phi} \right] d\boldsymbol{x} = 0 \quad \forall \tilde{\phi} \in \mathcal{H}^1(\mathcal{D}) \tag{2.27}$$

Where Δt is the time step. It is worth noticing that the reaction term is relaxed and evaluated at the previous time step.

2.4 Post processing

This RDE approach will yield an optimal configuration which is not printable, since it will present a wide area where $0 < \phi < 1$. This area does not represent neither solid nor empty space. In order to obtain a printable configuration, a second optimization is done replacing the DWP as a reaction term in the RDE. The DWP-RDE equation is defined as follows [39]

$$\frac{\partial \phi}{\partial t} = \kappa \nabla^2 \phi - \frac{\partial \mathcal{F}}{\partial \phi} \quad \text{in} \quad \mathcal{D} \times (t_1 \leq t \leq t_2), \tag{2.28}$$

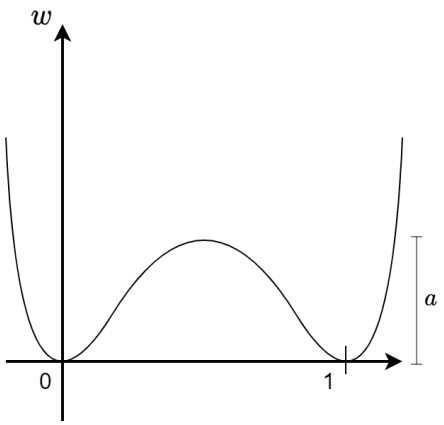
with

$$\mathcal{F}(\phi) = aw(\phi) + g(\phi) \left. \frac{\partial \widetilde{\mathcal{L}}}{\partial \phi} \right|_{t=t_1}, \tag{2.29}$$

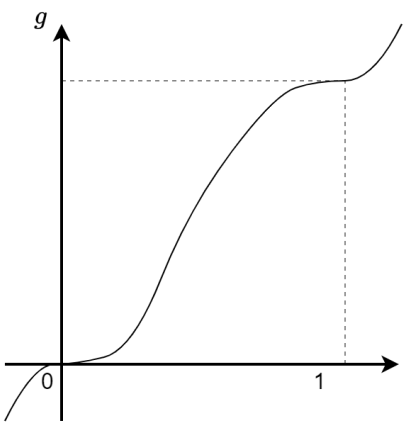
where the two functions w and g are set as

$$\begin{cases} w(\phi) = \phi^2(1-\phi)^2, \\ g(\phi) = \phi^3(6\phi^2 - 15\phi + 10). \end{cases} \tag{2.30}$$

When restricted to the interval $[0, 1]$, these functions become effectively a smooth Dirac's delta and a smooth Heaviside function, as graphically shown in Figure 2.2



(a) function w



(b) function g

As before, we need the weak formulation of Equation 2.28 to numerically solve the problem. Computing all the terms and using yet again a Backward Euler time scheme and continuous FEM is space, the weak formulation is: For $n = 1, 2, ..$, find $\phi_{n+1} \in \mathcal{H}^1(\mathcal{D})$ such that

$$\int_{\mathcal{D}} \left[\frac{\phi_{n+1} - \phi_n}{\Delta t} \tilde{\phi} + \kappa \nabla \phi_{n+1} \cdot \nabla \tilde{\phi} + \left(a w'(\phi_{n+1}) + 30 w(\phi_{n+1}) \frac{\partial \widetilde{\mathcal{L}}}{\partial \phi}(\phi_n, q_n, r_n) \right) \tilde{\phi} \right] d\boldsymbol{x} = 0 \quad \forall \tilde{\phi} \in \mathcal{H}^1(\mathcal{D}) \quad (2.31)$$

This second optimization procedure, does not aim to obtain a new optimal configuration, but rather to to diminish the area in which ϕ is transitioning from 0 to 1.

The two optimization cycles are done sequentially. The first cycle will stop when the difference in volume between two consecutive iterations will be below a certain threshold, while the second one will stop when the transitional area of ϕ is reduced to less than 1% of the domain area.

Algorithm 1 Topology Optimization algorithm

```
Construct computational mesh T_h
Initialize φ_0^h
while ‖φ_{k+1}^h − φ_k^h‖_{L^∞(Ω)} > tol do
    Solve elastic problem → u_{k+1}^h
    Compute sensitivity → J'(φ_k^h, u_{k+1}^h)
    Solve RDE → φ_{k+1}^h
    Update parameters → {k + 1, q_{k+1}, r_{k+1}}
end while
while M_nd > 1% do
    Solve elastic problem → u_{k+1}^h
    Compute sensitivity → J'(φ_k^h, u_{k+1}^h)
    Solve RDE-DWP → φ_{k+1}^h
    Update parameters → {k + 1, q_{k+1}, r_{k+1}, κ_{k+1}}
end while
```

In the above algorithm the term M_{nd} represents the measure of the transitorial interface of ϕ, formally:

$$M_{nd} = \frac{\int_{\mathcal{D}} 4\phi(1-\phi)}{|\mathcal{D}|}. \quad (2.32)$$

2.5 Anisotropic mesh adaptation

This section will cover the explanation of the anisotropic mesh adaptation procedure used inside the topology optimization routine, starting from the concept of metric of a mesh.

It is known that the precision of any numerical scheme is directly linked to the size of the mesh over which the solution is computed. It is clear how an indiscriminate refining of the domain represents a sub optimal

approach. An adapted mesh, which is refined only where needed is a much more ideal concept, which can allow to reduce the computational burden of the optimization routine. The mesh adaptation procedure contained in the following sections allows to obtain meshes with highly stretched elements near the interface of ϕ through a recovery based error indicator.

2.5.1 Metric of a FEM mesh

We define any triangulation of the domain $\mathcal{D}$ as $\mathcal{T}_h$. Each element K of the mesh can be characterized geometrically through the spectral properties of an affine invertible map $T_K : \hat{K} \to K$. This mapping transforms a reference element $\hat{K}$, by definition an equilateral triangle inscribed in the unit circle centered at the origin, into a generic triangle K inscribed in an ellipse, as Figure 2.2 shows.

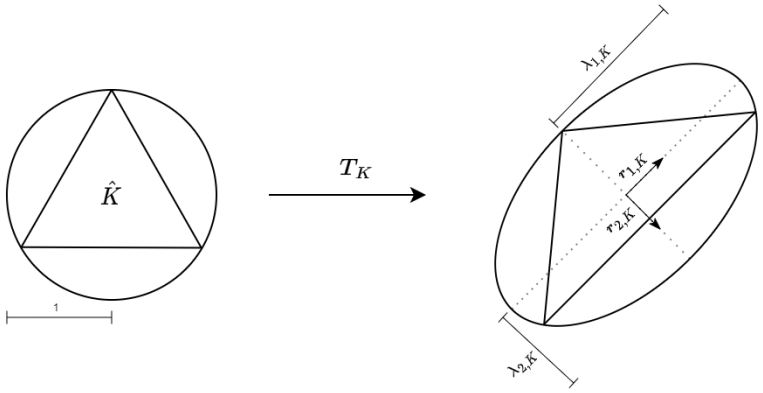


Figure 2.2: Mapping from reference element to actual element

The position of each vertex of the triangle in the reference element $\hat{K}$ is known in the 2D case, and this allows to compute the Jacobian of the transformation T_K, namely, $\boldsymbol{M}_K \in \mathbb{R}^{2\times2}$. This matrix can be factored via the polar decomposition as

$$\boldsymbol{M}_K = \boldsymbol{B}_K \boldsymbol{Z}_K, \tag{2.33}$$

where $\boldsymbol{B}_K \in \mathbb{R}^{2\times2}$ is a symmetric positive definite matrix describing the deformation of the triangle K, while $\boldsymbol{Z}_K \in \mathbb{R}^{2\times2}$ characterizes the rotation of the element.

Additionally, it is possible to spectrally decompose the matrix $\boldsymbol{B}_K$ as

$$\boldsymbol{B}_K = \boldsymbol{R}_K^T \boldsymbol{\Lambda}_K \boldsymbol{R}_K, \tag{2.34}$$

where $\boldsymbol{R}_K^T = [\boldsymbol{r}_{1,K}, \boldsymbol{r}_{2,K}]$ is the right eigenvector of $\boldsymbol{B}_K$, and $\boldsymbol{\Lambda}_K \in \mathbb{R}^{2\times2}$ is the diagonal matrix storing the corresponding eigenvalues $\lambda_{1,K} \geq \lambda_{2,K} > 0$. Figure 2.2 gives a graphical representation of the eigenvectors describing the direction of the two semi-axis of the ellipse, while the eigenvalues represent their length.

We now introduce the symmetric positive definite matrix $\mathcal{M} \in \mathbb{R}^{2\times2}$, known as metric. It will contain all the information of the triangulation relative to it. Practically, the metric will be approximated with its piecewise constant version $\mathcal{M}|_K$ defined on the mesh $\mathcal{T}_h$.

The mesh metric is defined as follows

$$\mathcal{M}|_K = \boldsymbol{R}_K^T \boldsymbol{\Lambda}_K^{-2} \boldsymbol{R}_K \quad \forall K \in \mathcal{T}_h \tag{2.35}$$

Being $\mathcal{M}|_K$ symmetric positive definite, is possible to re-organize all the information contained in it in a vector using the Voigt notation, as follows

$$\begin{bmatrix} M_{11} & M_{12} \\ M_{12} & M_{22} \end{bmatrix}_K = \begin{bmatrix} M_{11} & M_{12} & M_{22} \end{bmatrix}^T \tag{2.36}$$

This method allows us to not store M_{12} twice, while still having all the needed information about the metric. It is important to note, that by definition of symmetric positive definite matrix, the two components M_{11} and M_{22} will always be positive, while M_{12} could be both positive or negative.

An additional local quantitative indicator for the mesh elements will be the stretching factor s_K, defined as $s_K = \lambda_{1,K}/\lambda_{2,K} \geq 1$. This parameter quantifies how deformed an element K actually is, since it is the ratio between the two semi-axis of the ellipse in which the triangle is inscribed.

2.5.2 Mesh adaptation

The mesh adaptation routine is based on the a posteriori error estimation technique proposed by Zienkiewicz and Zhu [44, 45]. Considering a numerical solution ϕ^h and its continuous counter part ϕ, the gradient $\nabla\phi^h$ will always be less accurate than the solution. A recovery procedure is performed to obtain an improved approximation $\boldsymbol{P}(\nabla\phi^h)$ of $\nabla\phi^h$. The difference $\|\boldsymbol{P}(\nabla\phi^h) - \nabla\phi^h\|_{\mathcal{L}^2(\Omega)}$ will provide an a posteriori error estimation of the discretization error $\phi - \phi^h$ in the $\mathcal{H}^1$ seminorm.

The explicit formula for the estimation $\boldsymbol{P}(\nabla\phi^h)$ [31] used in this work is:

$$\boldsymbol{P}_{\Delta_K}(\nabla\phi^h) = \frac{1}{|\Delta_K|}\sum_{T\in\Delta_K} |T|\ \nabla\phi^h\Big|_T \tag{2.37}$$

In the above formulation, Δ_K represents an elemental patch $\Delta_K = \{T \in \mathcal{T}_h | T \cup K \neq \emptyset\}$ and $|\cdot|$ stands for area. A visual depiction of the reconstructed gradient is given by Figure 2.3.

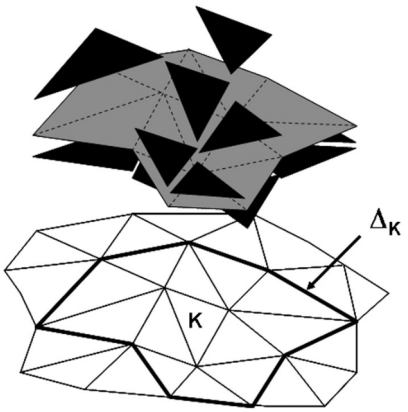


Figure 2.3: Gradient of the discrete solution (black) and the recovered gradient (gray). Image from [35]

The formulation of the anisotropic recovery-based estimator will be based on the definition of Equation 2.37. We define a symmetric positive definite matrix $\boldsymbol{G}_{\Delta_K} \in \mathbb{R}^{2\times 2}$ such that the entries are defined as

$$[\boldsymbol{G}_{\Delta_K}]_{ij} = \sum_{T\in\Delta_K}\int_T \left[\boldsymbol{P}_{\Delta_K}(\nabla\phi^h) - \nabla\phi^h\Big|_{\Delta_K}\right]_i \left[\boldsymbol{P}_{\Delta_K}(\nabla\phi^h) - \nabla\phi^h\Big|_{\Delta_K}\right]_j d\boldsymbol{x} \tag{2.38}$$

We now define the elemental anisotropic estimator [2]

$$\eta_K^2 = \lambda_{1,K}\lambda_{2,K}|\widehat{\Delta}_K|\left[s_K \boldsymbol{r}_{1,K}^T \widehat{\boldsymbol{G}}_{\Delta_K}\boldsymbol{r}_{1,K} + s_K^{-1}\boldsymbol{r}_{2,K}^T\widehat{\boldsymbol{G}}_{\Delta_K}\boldsymbol{r}_{2,K}\right] \tag{2.39}$$

where $\widehat{\boldsymbol{G}}_{\Delta_K} = \boldsymbol{G}_{\Delta_K}/|\Delta_K|$ and $|\widehat{\Delta}_K| = \mathcal{T}_K^{-1}(\Delta_K)$. The minimization of the number of element while using the anisotropic estimator leads to the problem

$$\min \mathcal{J}_K(s_K, \boldsymbol{r}_{1,K}, \boldsymbol{r}_{2,K}) = s_K\boldsymbol{r}_{1,K}^T\widehat{\boldsymbol{G}}_{\Delta_K}\boldsymbol{r}_{1,K} + s_K^{-1}\boldsymbol{r}_{2,K}^T\widehat{\boldsymbol{G}}_{\Delta_K}\boldsymbol{r}_{2,K} \quad \text{s.t.} \quad \begin{cases} s_K \geq 1 \\ \boldsymbol{r}_{i,K}\cdot\boldsymbol{r}_{j,K} = \delta_{ij} \end{cases} \tag{2.40}$$

where δ_{ij} is the Kronecker delta. The solution to the latter problem (see [30]) is obtained when $s_K = s_K^\star$ and $\boldsymbol{r}_{i,K} = \boldsymbol{r}_{i,K}^\star$, such that

$$\begin{gathered} s_K^\star = \sqrt{\frac{\theta_1}{\theta_2}} \\ \boldsymbol{r}_{1,K}^\star = \boldsymbol{t}_2, \quad \boldsymbol{r}_{2,K}^\star = \boldsymbol{t}_1 \end{gathered} \tag{2.41}$$

where $\{\theta_i, \boldsymbol{t}_i\}_{i=1,2}$ are the eigenpairs associated with $\widehat{\boldsymbol{G}}_{\Delta_K}$, with $\theta_1 \geq \theta_2 \geq 0$ and $\{\boldsymbol{t}_i\}_{i=1,2}$ orthonormal. The only missing component before being able to define an optimal metric $\mathcal{M}^\star$ unambiguously, are the optimal eigenvalues $\lambda^\star_{1,K}$ and $\lambda^\star_{2,K}$, which are embedded in the optimal stretch factor $s^\star_K$. Combining the definition of optimal stretch factor with the estimator, the optimal eigenvalues results to be:

$$\lambda^\star_{1,K} = \theta_2^{-1/2} \left(\frac{\tau^2 \|\nabla \phi^h\|^2_{\Delta_K}}{2|\widehat{K}|} \right)^{1/2}, \tag{2.42a}$$

$$\lambda^\star_{2,K} = \theta_1^{-1/2} \left(\frac{\tau^2 \|\nabla \phi^h\|^2_{\Delta_K}}{2|\widehat{K}|} \right)^{1/2}, \tag{2.42b}$$

where τ is a user defined parameter used to impose a desired tolerance, and

$$\|\nabla \phi^h\|^2_{\Delta_K} = \frac{1}{|\Delta_K|} \sum_{T \in \Delta_K} |T| \left\| \nabla \phi^h \Big|_T \right\|^2. \tag{2.43}$$

Using the pairs $\{\lambda^\star_{i,K}, \boldsymbol{r}^\star_{i,K}\}_{i=1,2}$ is possible to compute the optimal metric $\mathcal{M}^\star$ for every element K. Such metric is turned into a new mesh by means of the function `adaptmesh` provided by FreeFem++ [17]. This mesh adaptation procedure is implemented during the RDE-DWP potential optimization cycle, before every DWP computation, effectively allowing the routine to always have the optimal mesh supporting the computation, thanks to the Adaptive Mesh Refinement (AMR) procedure.

Algorithm 2 Topology Optimization algorithm with AMR

Construct computational mesh $\mathcal{T}_h$
Initialize ϕ^h_0
while $\|\phi^h_{k+1} - \phi^h_k\|_{\mathcal{L}^\infty(\Omega)} >$tol **do**
  Solve elastic problem $\rightarrow \boldsymbol{u}^h_{k+1}$
  Compute sensitivity $\rightarrow \mathcal{J}'(\phi^h_k, \boldsymbol{u}^h_{k+1})$
  Solve RDE $\rightarrow \phi^h_{k+1}$
  Update parameters $\rightarrow \{k+1, q_{k+1}, r_{k+1}\}$
end while
while $M_{nd} > 1\%$ **do**
  Construct gradient reconstruction $\rightarrow \boldsymbol{P}_{\Delta_K}(\nabla \phi^h_k)$
  Estimate optimal eigenpairs $\rightarrow \{\lambda^\star_{i,K}, \boldsymbol{r}^\star_{i,K}\}_{i=1,2}$
  Compute optimal metric $\rightarrow \mathcal{M}^\star$
  Compute optimal mesh via `adaptmesh` $\rightarrow \mathcal{T}^{k+1}_h$
  Solve elastic problem $\rightarrow \boldsymbol{u}^h_{k+1}$
  Compute sensitivity $\rightarrow \mathcal{J}'(\phi^h_k, \boldsymbol{u}^h_{k+1})$
  Solve RDE-DWP $\rightarrow \phi^h_{k+1}$
  Update parameters $\rightarrow \{k+1, q_{k+1}, r_{k+1}, \kappa_{k+1}\}$
end while

Chapter 3

The Graph Neural Network for quasi-optimal meshes

3.1 The idea

The main objective of the Graph Neural Network (GNN) will be to build a complete FEM mesh with anisotropically stretched elements, without resorting to the above error estimator and adaptation procedures. To achieve this goal, a Graph Auto Encoder (GAE) architecture was devised. A GAE consists in a GNN that reduces the dimension of its input data via an encoding procedure, and then reconstructs the input as output, through a deconding procedure that starts from the so-called latent space.

The information relative to the original mesh chosen as input is the metric, since it allow to uniquely and fully describe a given mesh. Figure 3.1 gives the general scheme of how the GNN will operate. A mesh arising from a topology optimization case will produce the input metric and the output will be a reconstructed version of the it, that will allow us to build a new mesh from it. The architecture will learn to reconstruct the metric as close to the original as possible.

$$\text{GNN Input: } \begin{bmatrix} M_{11}^1 & M_{12}^1 & M_{22}^1 \\ M_{11}^2 & M_{12}^2 & M_{22}^2 \\ \vdots & \vdots & \vdots \\ M_{11}^N & M_{12}^N & M_{22}^N \end{bmatrix}, \quad \text{GNN Output: } \begin{bmatrix} m_{11}^1 & m_{12}^1 & m_{22}^1 \\ m_{11}^2 & m_{12}^2 & m_{22}^2 \\ \vdots & \vdots & \vdots \\ m_{11}^N & m_{12}^N & m_{22}^N \end{bmatrix}$$

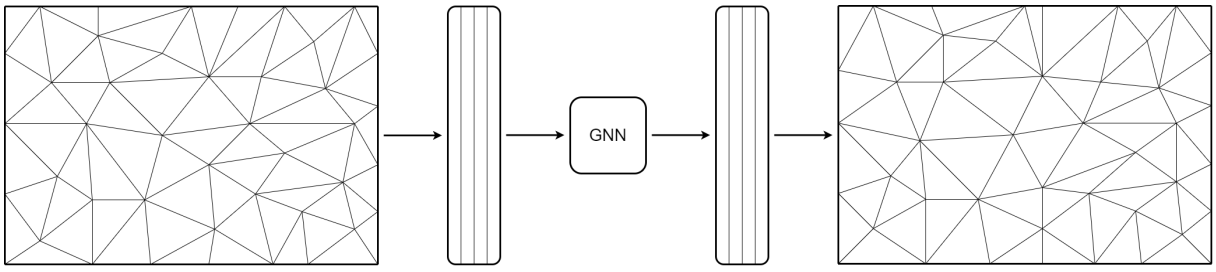


Figure 3.1: General idea

The GNN will essentially be divided in two sections: encoding and decoding. The encoding part will take care of reducing the cardinality of the graph associated with the input and compressing the information into a lower dimension latent space. The decoding part will do exactly the opposite, starting from the compressed information of the latent space, it will increase the number of nodes of the graph and will decompress the information over the new nodes.

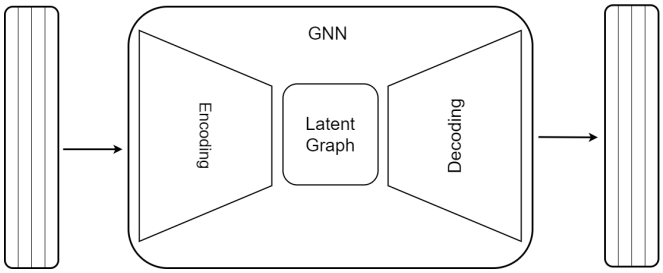


Figure 3.2: Internal structure of the GNN

To assest how similar the input and output metrics in the N graph nodes really are, an L1 loss function has been implemented, as well as a L2 Regularization [27] for the M parameters $\theta_j,\ j = 1..M$ that each respective model will have.

$$\mathcal{L}(\boldsymbol{M}, \boldsymbol{m}) = \frac{1}{N}\sum_{N}^{i=1} |\boldsymbol{M}_i - \boldsymbol{m}_i| + \frac{\lambda}{M}\sum_{M}^{j=1} \theta_j^2 \tag{3.1}$$

3.1.1 From metric to graph

As seen in Section 2.5.1, it is possible to relate a mesh with a vectorial function called metric, which is defined on each node of the discretization. It is possible to define the metric at each point of the domain through linear interpolation between nodes. This interpolation allows to project the metric function on a different mesh and more specifically, a structured reference mesh, as Figure 3.3 grafically shows.

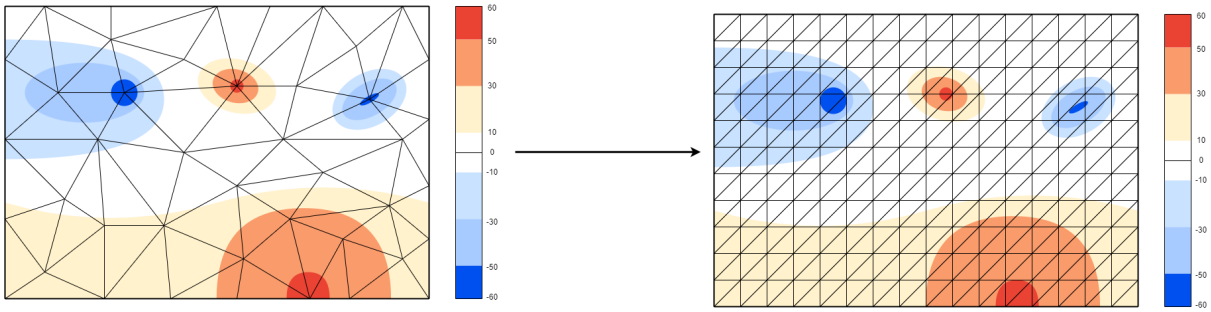


Figure 3.3: Projection of the metric on the structured reference mesh

The reference structured mesh will act as the base structure of the graph used in the GNN. To relate this triangulation to a graph, each node of the reference mesh will be considered as a node of the graph. The edges of the discretization will be used as edges of the graph as well, effectively allowing us to represent the original mesh through a graph structured as the reference mesh.
Each node of the graph will contain as information the values of the metric in that point of the domain, which will be exact up to the linear interpolation.

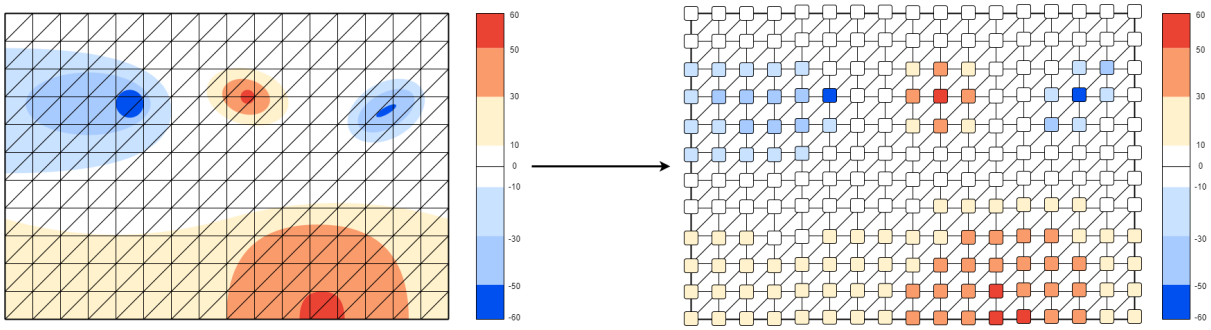


Figure 3.4: Creation of the graph associated with the metric

It is important to remark the role of the reference mesh for the whole architecture and for the accuracy of final results. The structured reference mesh will also be used for the outputs of the GNN, allowing straight forward comparison thanks to the node enumeration and position.

Remark. It is clear that if the reference mesh is insufficiently refined, part of information originally computed on the original triangulation could potentially be lost during the projection, effectively preventing the reconstruction of an accurate mesh, even with the exact output metric.

3.2 Structure of the GNN

The Encoding-Decoding procedure was designed to be as modular as possible, allowing to tackle different problems that would require different model structures, to be treated with the same code. For this reason, both Encoder and Decoder have been built in blocks, which can be sequentially stacked to achieve a more complex network, capable of reducing the dimension of the latent space at will.

To successfully reduce the inputs, three operations are necessary:

- Message Passing
- Graph Reduction
- Graph Augmentation

The first operation will ensure that each node shares information with its neighbors, and it will modulate how many information are stored in each graph node. The second operation will reduce the cardinality of the graph by eliminating nodes and all the edges connected to it. The last operation will improve the connectivity of a given graph, allowing for a more even spread of the information.

It has to be noted that only the first two operations are learnable layers of the GNN, while the third is a deterministic mathematical operation used to improve the performance of the overall structure.

3.2.1 Encoding block

Using all the three operations previously cited, is possible to create a modular encoding block, which will produce as output a graph smaller than the input one. It is clear how multiple blocks could be sequentially stacked in order to obtain very diminished and dense graph that encodes as much information as possible.
The first operation of a block will be the Graph Augmentation. The augmented connectivity will be used for the second operation, the Message Passing. After these two operations, the only task left to do would be to select the most important nodes, and discard the less valuable by using the Graph Reduction.

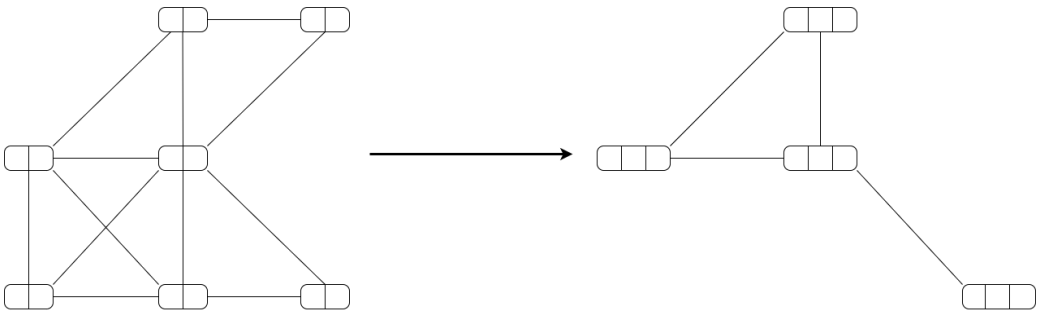


Figure 3.5: Combination of Message Passing and Graph Reduction layers in an encoding block

Figure 3.5 gives a visual representation of a possible pair of input and output graph of an encoding block, where the amount of nodes features stored in each node increases, while the cardinality of the graph decreases.

3.2.2 Decoding block

The decoder of the GNN will function similarly to the encoder, by using stackable blocks that will take a graph as input, and return a graph as output.

The objective of the decoding is exactly the opposite as the encoder, meaning that the number of features of each node will decrease between input and output, but the number of nodes of the overall graph cardinality will increase, as Figure 3.6 visually describes.

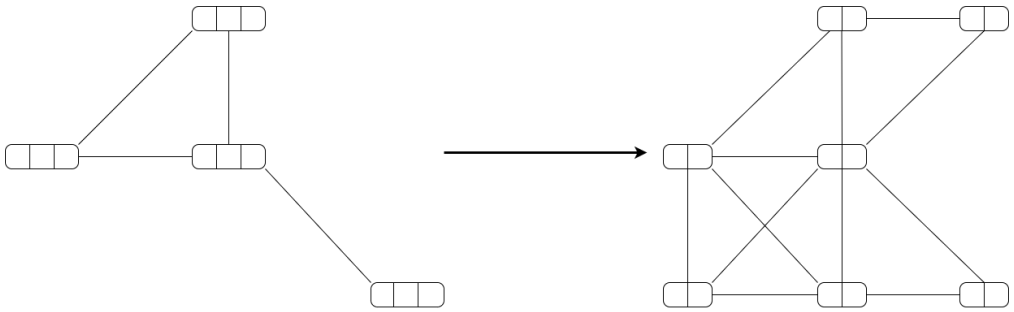


Figure 3.6: Combination of unpooling operation and Convolution layer in an decoding block

The first operation of the decoding block would be the unpooling, which is the opposite operation to the Graph Reduction. New nodes are created, with empty features, and are connected to previously existing nodes using minimal information coming from the encoding procedure. The empty nodes are then filled with information using a Message Passing layer.

3.2.3 Technical details of the operations

Message Passing: Convolutional layer

The chosen Massage Passing operation for the GNN will be the GCN Convolution[23]. This layer allows each node to receive information about the nodes directly connected to them and to share their own, allowing to compute the updated nodal information $\boldsymbol{X}'$ through the following operation:

$$\boldsymbol{X}' = \widehat{\boldsymbol{D}}^{-\frac{1}{2}}\widehat{\boldsymbol{A}}\widehat{\boldsymbol{D}}^{-\frac{1}{2}}\boldsymbol{X}\boldsymbol{\Theta} \tag{3.2}$$

where $\widehat{\boldsymbol{A}} = \boldsymbol{A} + \boldsymbol{I}$ denotes the adjacency matrix with inserted self-loops, $\widehat{\boldsymbol{D}}$ the diagonal matrix of $\widehat{\boldsymbol{A}}$, $\boldsymbol{X}$ the node feature prior to the operation and $\boldsymbol{\Theta}$ denotes the learnable parameters.

From the above formulation is clear how the dimension of the new nodes feature $\boldsymbol{X}'$ could be different from the original ones, meaning that is possible to increase or decrease the number of features contained in each node. Figure 3.7 visually explains how the information stored in each node can spread in the graph thanks to the connections between them.

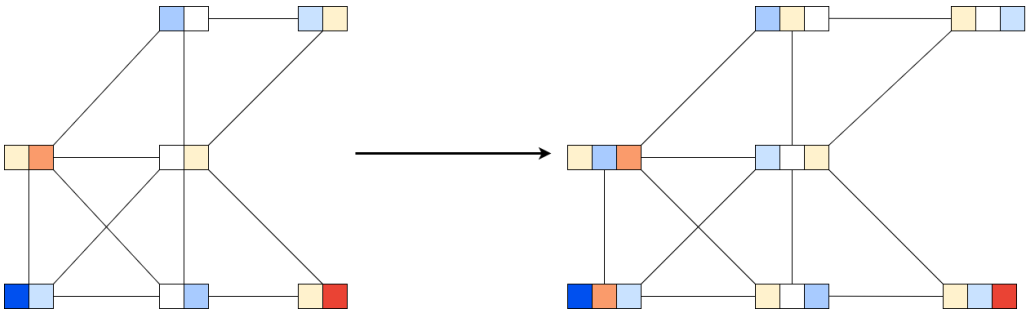


Figure 3.7: Visualization of the GCN Convolution acting on a graph

Graph Reduction: Pooling layer

To reduce the cardinality of the graph, the TopK Pooling layer[11] was used.
This layer assigns a score $\boldsymbol{y}$ to each node, based on its feature and the learnable parameters, to then discard all the nodes that do not have a sufficiently high score. The node features are then filtered via an activation function and the adjacency matrix is updated by retaining only the pre-existing edges between the non-discarded nodes.

$$\begin{cases} \boldsymbol{y} = \sigma\left(\frac{\boldsymbol{X}\boldsymbol{\Theta}}{\|\boldsymbol{\Theta}\|}\right) \\ i = top_k(\boldsymbol{y}) \\ \boldsymbol{X}' = (\boldsymbol{X} \odot \tanh \boldsymbol{y})_i \\ \boldsymbol{A}' = \boldsymbol{A}_{i,i} \end{cases} \tag{3.3}$$

The amount of eliminated nodes can be modulated by specifying the percentage of nodes to discard, effectively leaving on the graph the top percentage of them, based on their score.

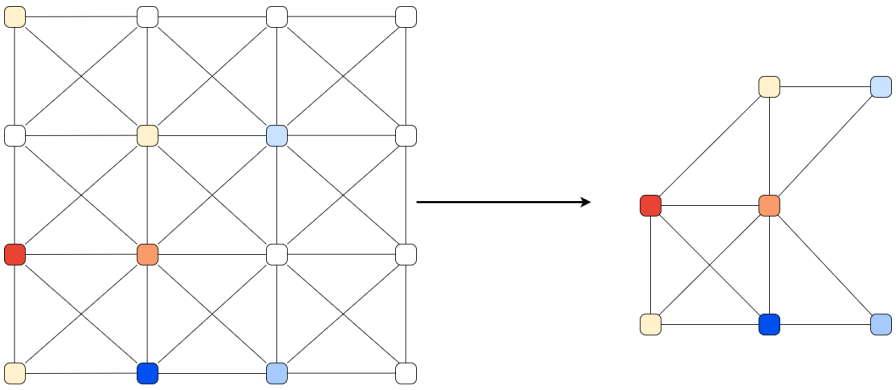


Figure 3.8: Visualization of the TopK Pooling acting on a graph

Graph augmentation

It is important to improve the connectivity of the base graph to allow the information stored in each node, to be shared not only with their first neighbors, but with as much nodes as possible. Furthermore, the pooling operation could leave some poorly connected nodes isolated during the training process, effectively preventing them from sharing or receiving any information about their features.

The chosen graph augmentation technique will be the Graph Power[7].
By raising the adjacency matrix relative the a graph to the k-th power, it is possible to build connections for nodes that are at most k-th neighbors. This work will use $k = 2$ for the graph augmentation operation, increasing the connectivity of each node to its second neighbors every time it is used, that is,

$$\boldsymbol{A}' = \boldsymbol{A}^2 \tag{3.4}$$

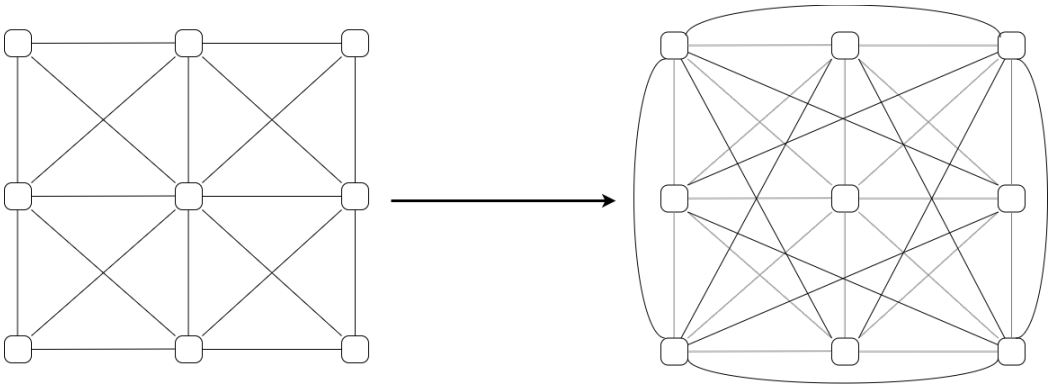


Figure 3.9: Graph augmentation with $k = 2$ applied to a sample graph

3.2.4 Overall GNN architecture

Figure 3.10 visually summarizes the overall structure of the Graph Neural Network. It is important to note that the encoder does not include only encoding blocks, but it also features a single Message Passing layer after the last encoding block. This layer does not aim to increase or decrease the amount of feature in each node like the ones in the encoding and decoding blocks.
This layer is in place to better share the information contained in the most compressed graph, allowing a more even spread of the information between features among the latent space graph.

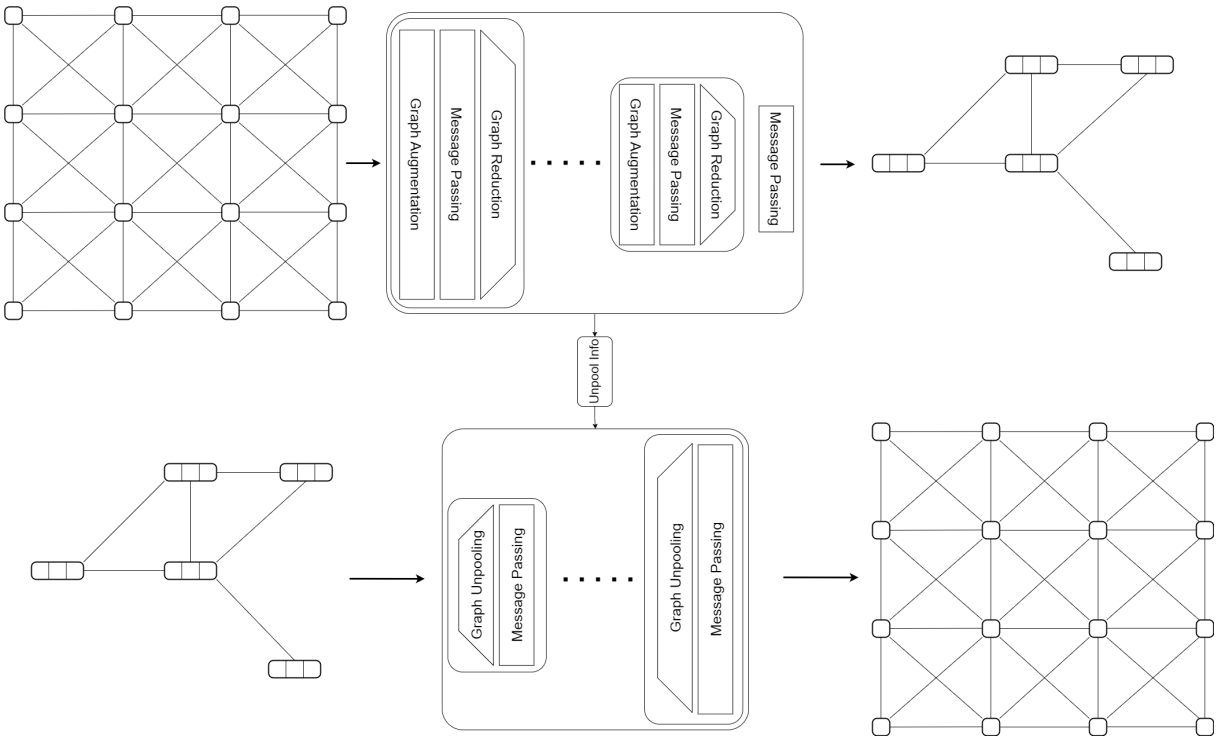


Figure 3.10: The complete structure of the GNN, divided into Encoding and Decoding

3.3 Metric transformations and normalization

As seen in Section 2.5.1, the metric has three components, but by definition, only the second one allows negative values. This disparity of range between the various components of the metric places some heavy constraints on the problem as it is.

The choice of an activation function that does not filter out negative values, could potentially impact in a negative way the training of the strictly positive components [14]. Another potential issue could be the presence of sudden changes in the metric, and being able to capture and reproduce them while they range in $\mathbb{R}^N$, could be much more challenging than doing so in a strictly positive valued environment such as $[\mathbb{R}^+]^N$.

3.3.1 Transformation applied to the metric

In order to avoid having one component having values in $\mathbb{R}^N$ and the two of them in $[\mathbb{R}^+]^N$, a transformation for the $\boldsymbol{M}_{12}$ component of the metric has been developed. The latter is divided into two new ones, as visually depicted in Figure 3.11

$$\begin{aligned} \mathcal{F} &: \mathbb{R}^N \to [\mathbb{R}^+]^N \times [\mathbb{R}^+]^N \\ \mathcal{F}(\boldsymbol{M_{12}}) &\to \{\boldsymbol{M_{12}}^{mod}, \boldsymbol{M_{12}}^{sgn}\} = \{|\boldsymbol{M_{12}}|, |\min(\boldsymbol{M_{12}}, \boldsymbol{0})|\} \end{aligned} \tag{3.5}$$

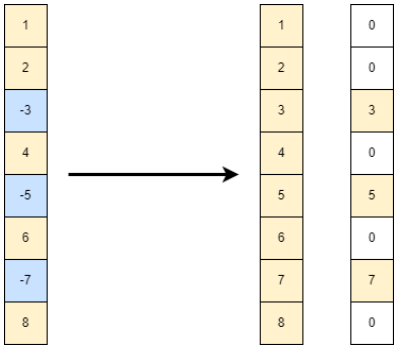


Figure 3.11: Transformation applied to the $\boldsymbol{M}_{12}$ component of the metric

The first new component of the i-th node, namely $M^{mod}_{12,i}$ is the absolute valued version of $M_{12,i}$. The second component of the i-th node, $M^{sgn}_{12,i}$, is the absolute value of $\min(M_{12,i}, 0)$.
Essentially, $\boldsymbol{M}^{sgn}_{12}$ is meant to represent the nodes in which the sign of $\boldsymbol{M}_{12}$ is negative. This tensor has been developed this way to impose a bias on the important indices to estimate during the training process. In particular, it is important to estimate correctly the position of the nodes having negative sign, but more importance must be given to the nodes that have a greater absolute value, since they will influence the meshing process the most. This transformation is applied to the data in the pre-process phase.

3.3.2 Normalization applied to the metric

considering the transformation applied to the metric in the pre-process phase, each component of the metric could potentially range in the whole $\mathbb{R}^+$ domain. To normalize the input data of the GNN, a Min-Max Scaling[32] approach has been used, and each component is normalized individually. The Min-Max Scaling consists of dividing every value of the tensor considered by the same value, effectively modifying the range in which tensor values lies. The chosen normalization value has been the maximum of each component, scaling all the features to the interval $[0, 1]$.
The normalization is applied right after the transformation in the pre-process phase.

3.3.3 Inverse transformation

The meshing algorithm of FreeFem++ will produce a mesh only if provided with a metric composed of three features, which raises the need for an inverse transformation to apply to the output of the GNN in order to have a compliant metric. The chosen method, has been to multiply the features of $\boldsymbol{M}^{mod}_{12}$ by -1 only if the corresponding feature of $\boldsymbol{M}^{sgn}_{12}$ is a number greater or equal than 1.

$$\mathcal{F}^{-1} : [\mathbb{R}^+]^N \times [\mathbb{R}^+]^N \to \mathbb{R}^N$$

$$\mathcal{F}^{-1}(\boldsymbol{M^{mod}_{12}}, \boldsymbol{M^{sgn}_{12}}) \to \begin{cases} M^{mod}_{12,i} \quad \text{if} \quad M^{sgn}_{12,i} < 1 \\ \\ -M^{mod}_{12,i} \quad \text{if} \quad M^{sgn}_{12,i} \geq 1 \end{cases} \qquad \forall i = 1..N \tag{3.6}$$

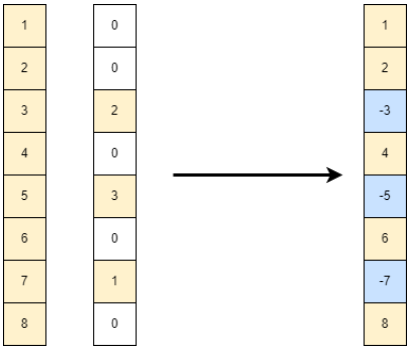


Figure 3.12: Inverse transformation applied to the $\boldsymbol{M}^{mod}_{12}$ and $\boldsymbol{M}^{sgn}_{12}$ component of the metric

Contrarily to the previous transformation, when reconstructing $\boldsymbol{M}_{12}$, it is not important if the estimated feature of the i-th node has a great absolute value $M^{sgn}_{12,i}$, but it is only important that the value has been estimated as a non-zero. The threshold of 1 was imposed due to the possibility of the network of estimating very low numbers instead of zeros.
The four scaling values are saved and passed to the post-process to allow a backward scaling to the original range before being passed to the `adaptmesh` function.

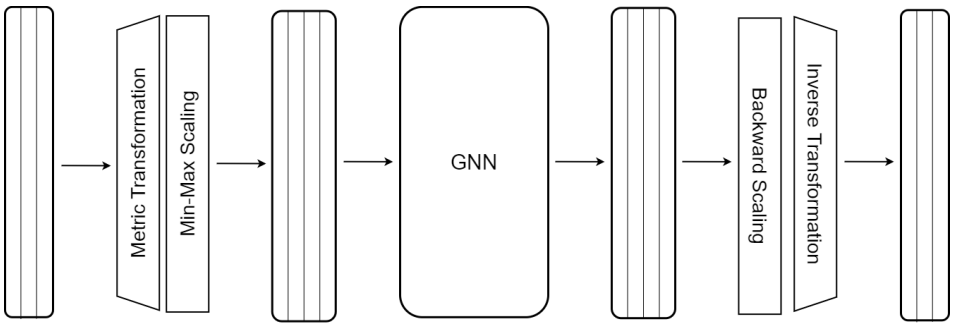


Figure 3.13: Workflow of the metric in pre and post process

Chapter 4

Numerical validation of the GNN

In this chapter the GNN is tested over a simplified problem. All the key features of an anisotropically adapted mesh are included, making possible to understand the capabilities of the GNN architecture.

4.1 Problem statement and Finite Elements approximation

Consider the following Poisson problem in Ω:

$$\begin{cases} -\Delta u = f \text{ in } \Omega, \\ u = u_D \text{ on } \partial\Omega, \end{cases} \tag{4.1}$$

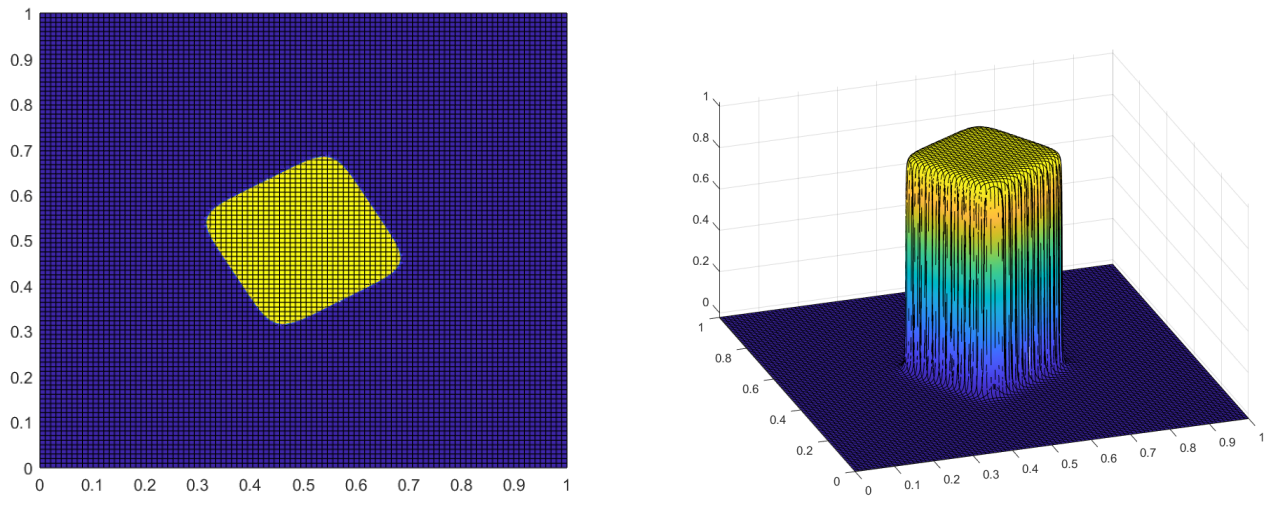
Where $\Omega = (0,1)^2$ is a unitary square domain, $\partial\Omega$ is its boundary and u_D is the analytical solution evaluated on said boundary. The forcing factor imposed ensures that the analytical solution u coincides with the following function

$$u(x,y) = \frac{1}{2} - \frac{1}{3}\arctan\Big(a\Big(\big((p_1(x,y) - p_1(b,c)\big)^m + \big((p_2(x,y) - p_2(b,c)\big)^m - d^m\Big)\Big). \tag{4.2}$$

Where p_1 and p_2 are the components of a vector function $\boldsymbol{p}$, which is a rotation of the Cartesian coordinates

$$\boldsymbol{p}(x,y) = \begin{pmatrix} p_1(x,y) \\ p_2(x,y) \end{pmatrix} = \begin{bmatrix} \cos\theta & -\sin\theta \\ \sin\theta & \cos\theta \end{bmatrix} \begin{pmatrix} x \\ y \end{pmatrix} \tag{4.3}$$

Furthermore, between u and $\boldsymbol{p}$, six parameters appears. These parameters modify how the solution looks like and where its maximum peak is located.
The values of the parameters b and c allows to move the peak of the solution over the domain, while maintaining its shape. Figure 4.1 shows how the analytical solution looks like with the parameters reported below.



(a) View from above

(b) View with an angle

Figure 4.1: Analytical solution for $b = c = 0.5$

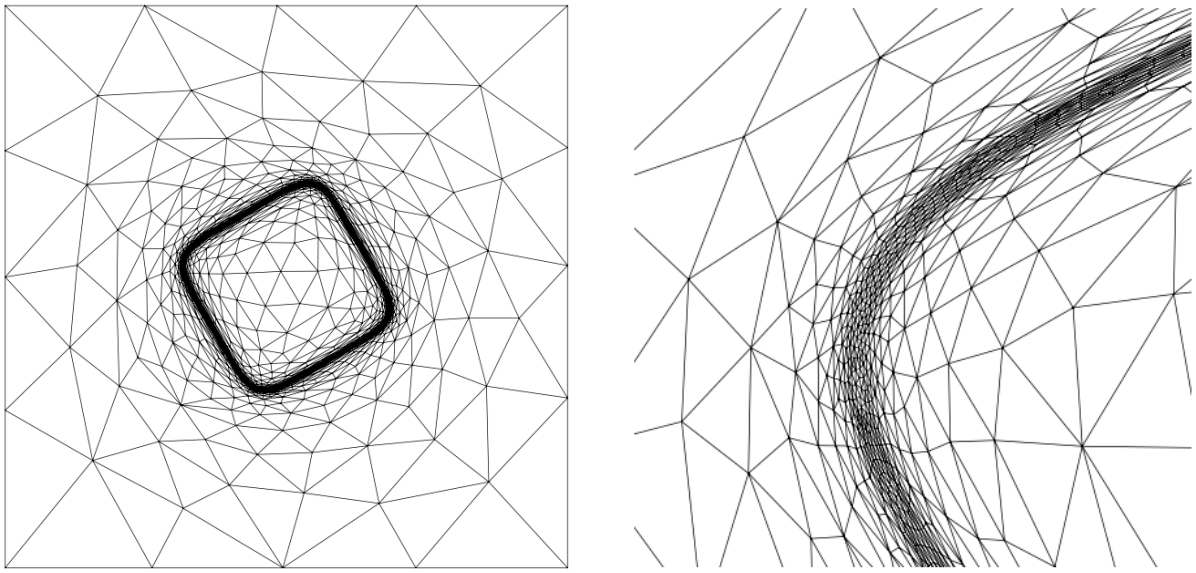
For the validation problem four of the six parameters will be fixed, while the remaining two will change in a closed interval, in order to construct a dataset of solutions.

$$\begin{cases} a = 5 \times 10^6 \\ b \in [0.05, 0.95] \\ c \in [0.05, 0.95] \\ d = 0.15 \\ m = 6 \\ \theta = -\frac{\pi}{6} \end{cases} \tag{4.4}$$

The validation problem is solved by means of Continuous Galerkin, using the following weak formulation for Equation 4.1: find $u \in \mathcal{H}^1(\Omega)$ such that

$$\int_\Omega \nabla u \nabla v \; d\boldsymbol{x} = \int_\Omega f v d\boldsymbol{x} \quad \forall v \in \mathcal{H}_0^1(\Omega). \tag{4.5}$$

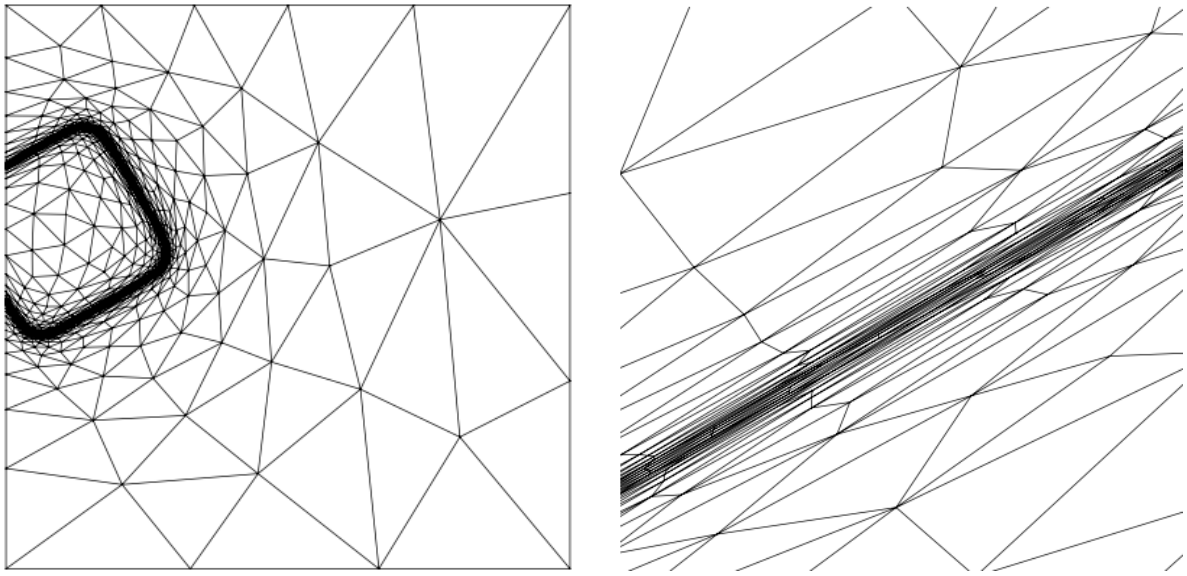
Given the analytical solution and the differential problem, is possible to produce anisotropically adapted meshes for the numerical solution, which will be the target to reconstruct for the GNN later on.
Figure 4.2a and 4.3a show two example of the resulting meshes. Is possible to note in Figure 4.2b and 4.2b how the elements near the vertical interface of the numerical solution are stretched in the direction perpendicular to the gradient of it.



(a) 2799 Vertices - 5580 elements

(b) Zoom on stretched elements

Figure 4.2: Resulting mesh when $b = c = 0.5$, with detailed view



(a) 2012 Vertices - 3952 elements (b) Zoom on stretched elements

Figure 4.3: Resulting mesh when $b = 0.1$ and $c = 0.6$, with detailed view

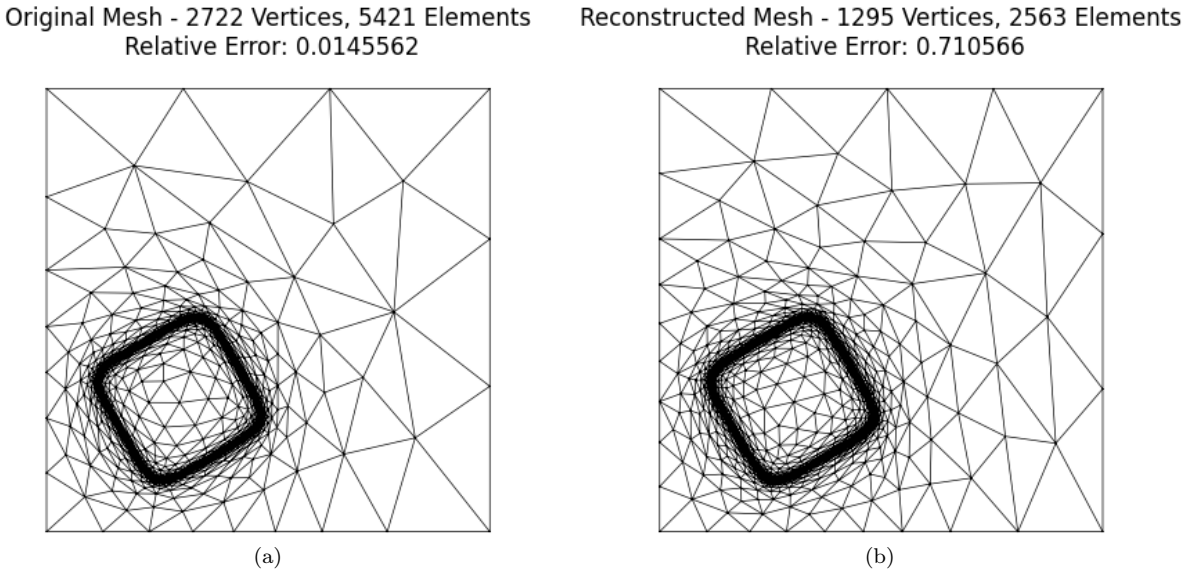
4.2 Mesh reconstruction from the metric

The GNN will work only with the normalized metrics, hence it must ensured that the information it works with are sufficient to correctly reconstruct the original mesh from which the metric is arising from. In order to check if a perfect metric reconstruction would recreate the original mesh, we provide the meshing algorithm with the exact metric, and we compare the input triangulation with the reconstructed one given as output by `adaptmesh`.

Additionally, we want to test how effective the two meshes are for the numerical validation problem, therefore we define a global relative error to evaluate the numerical solutions:

$$E_{rel} = \frac{\|u_{\mathcal{T}} - u_h\|_2}{\|u_{\mathcal{T}}\|_2} \tag{4.6}$$

where $u_{\mathcal{T}}$ represent the discretised version of the analytical solution u. As Figure 3.4 suggests, it is important to choose a reference mesh fine enough to capture as much information as the metric has to offer. An example of how much the reference mesh could alter the reconstruction of the original mesh, can be seen in Figure 4.4



(a) (b)

Figure 4.4: Metric sampling with a 100×100 reference mesh

The reconstructed mesh has less than half the vertices and elements, and the relative error of the numerical solution computed over it is more than one order of magnitude greater than the original one. By zooming on the element-dense area is possible to visualize how poorly the mesh was reconstructed.

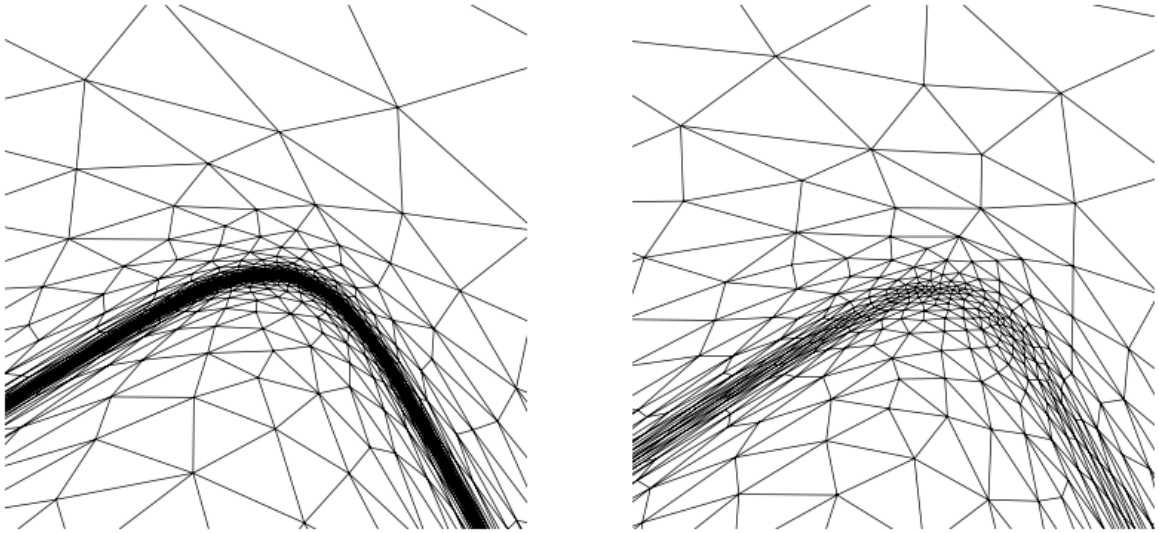
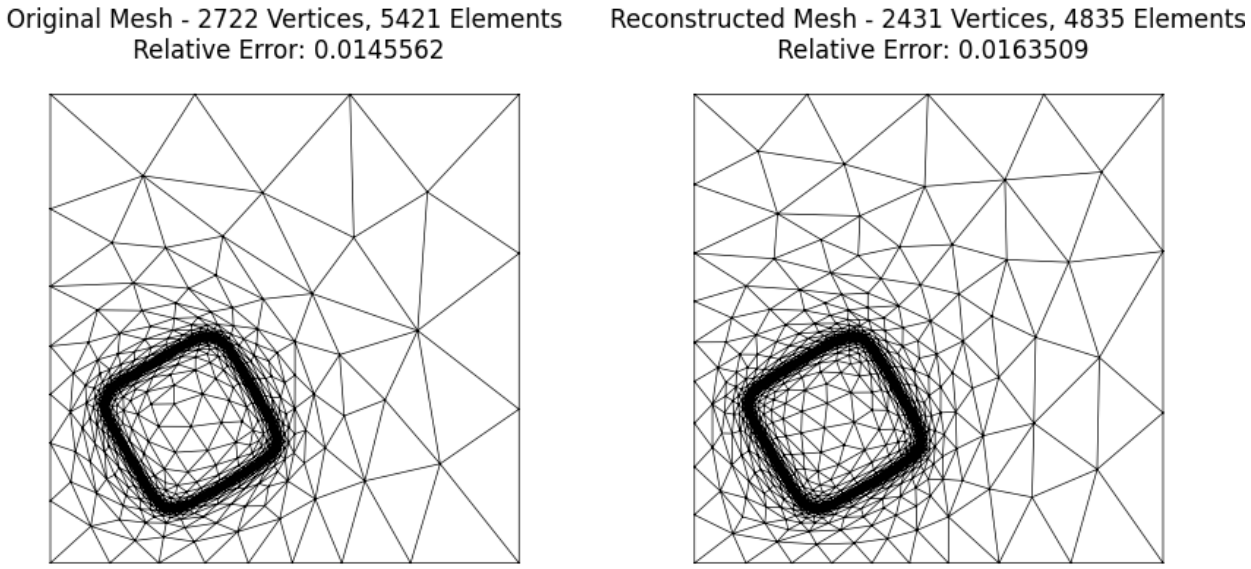


Figure 4.5: Detailed view of the element dense area of the mesh

It is therefore evident how the limiting factor in reconstructing the original mesh lies in the chosen reference mesh. Ideally an infinitely refined mesh would perfectly describe all the information contained within the metric, but this would be impossible on a practical level.

However, we recall that the metric is originally defined node-wise, and by linear interpolation, it is extended to the whole domain. This means that a reference mesh with characteristic length $h \simeq h_{min}$, where h_{min} is the smallest edge length of the original mesh, will capture the metric up to the interpolation error. Any finer reference mesh, will add little to no information to the reconstruction, but increasing the computational requirements needed to handle the operations.

Applying these consideration to the reconstruction procedure, we are able to almost perfectly reconstruct the original meshes, as seen in Figure 4.6



(a) (b)

Figure 4.6: Metric sampling with a 350×350 reference mesh

Even when comparing the relative error computed on both meshes, they differ from one another by less than $2 \cdot 10^{-3}$, hence proving that the reconstructed mesh is just as good as the original one to be the support of the

numerical problem. Given the similarities of the two meshes, and the impossibility of achieving a better result without compromising the hardware requirements to process the resulting graph, the reconstructed meshes will be considered the Ground Truth for the GNN, and any result computed will be compared to them.

4.3 GNN configurations

As seen in Section 3.3.1, the input graph of the GNN will have four features assigned to each node, one for every pre-processes component of the metric. Our objective is to reconstruct the metric as accurately as possible, as to have a resulting mesh similar to the input one. To achieve this goal, two main strategies have been tested during the training process: a monolithic and a multi network approach.

4.3.1 Monolithic approach

The monolithic approach consists of training one single Neural Network, which will have as input all the features of the pre-processed metric. This approach is intended to test the flexibility of the GNN, to understand if one single set of trainable parameters could be capable of reconstructing all the features. This method is undoubtedly the fastest, since it requires a single training to produce result, but it is potentially very inaccurate, since the features could have significantly different structure from one another in each single graph, making it difficult to reconstruct while using a single set of learnable parameters.

4.3.2 Multi Network approach

The Multi Network approach uses the opposite idea of the monolithic one. Each graph will be divided into four separate sub graphs, which will have the same connectivity as the original one, but only one feature in each node.

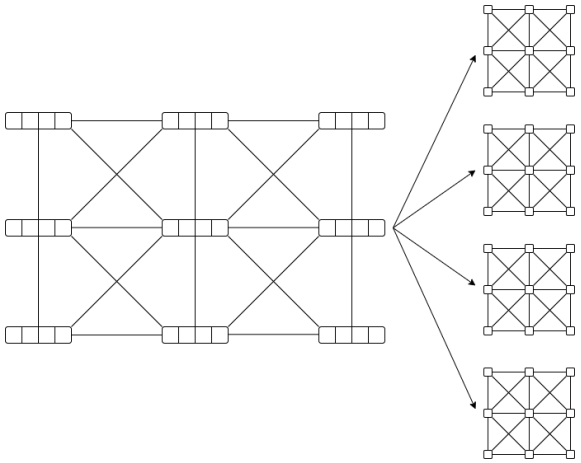


Figure 4.7: Multi Network splitting

This method is the slowest in terms of time and computations, since it requires four independent training to take place before allowing the computation of any result. On the other hand, this method could potentially be much more accurate, since it will have a set of trainable parameters for each component, allowing for a much more flexible reconstruction of the metric. Furthermore, this approach potentially allows the four networks to be trained in parallel, since they are independent from one another.

4.3.3 Technical details of the architecture

All the results obtained in the following section, and in Chapter 5 as well, use similar hyper-parameters. Only two activation function were used: the *tanh* [28] inside the pooling layer, and the *ReLu* [33]in the convolution layer.
The optimization algorithm implemented during training as been the Adam [21], with a MultiStep linear decay scheduler for the learning rate [36]. The latter has been used in a range from 5×10^{-3} to 5×10^{-5}. Additionally, a fixed batch size has been used during all training, varying in size from 2% to 5% of the size of the training database. The same batch size has been used for both training and validation, although the model allows for different sizes during the training process. To avoid problems related to vanishing gradients during training, all parameters are initialized using a He normal initialization [16].

4.4 Numerical results of the validation

To understand which configuration is capable of producing the best results, two equivalent architectures comprised of a single encoding block have been built.

4.4.1 The database

The database used for all the results in this section, is composed of 1600 metrics, arising from an equivalent number of meshes generated following the procedure detailed in Section 4.3.1. The parameters used for the mesh generation are shown in Equation 4.4, and each mesh differs from all the others in the choice of the parameters b and c respectively.
The database has been divided in 80% for training, 10% for validation and 10% for test.

4.4.2 Monolithic approach results

The chosen configuration for the monolithic approach is defined as follows in its single encoding block:

	Node Features	Cardinality Reduction
Block 1	$4 \rightarrow 16$	80%

Table 4.1: Monolithic single block architecture

The Message Passing layer will increase the number of features in each node from the original 4 to 16, and then the Graph Reduction layer will discard 80% of the graph nodes, based on the score it assigned. The training of the network went on until the average loss over the validation set stagnated for more than 10 epochs. Figure 4.8 is a visual representation of the metric as a vector. Each plot has the nodes' indices as horizontal axis, and the value of the metric on the vertical one. Comparing these plots for each component can give a qualitative idea of the results produced by the GNN.

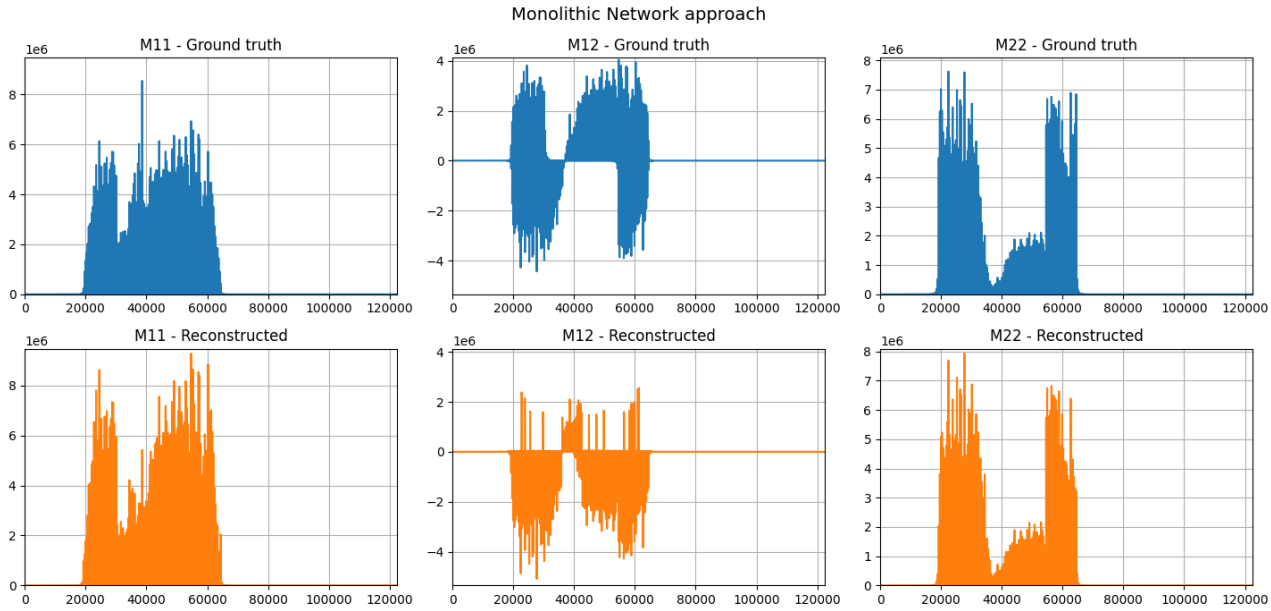


Figure 4.8: Metrics comparison

The comparison between the input and reconstructed metric using a monolithic approach suggests that the quality of the corresponding mesh will be low. It is particularly clear the struggle of in the reconstruction of the M_{12} component. By plotting the reconstructions of M_{12}^{mod} and M_{12}^{sgn}, appears that the source of the error lies in the estimation of the sign tensor, which leads to the inversion of sign of otherwise nicely estimated features.

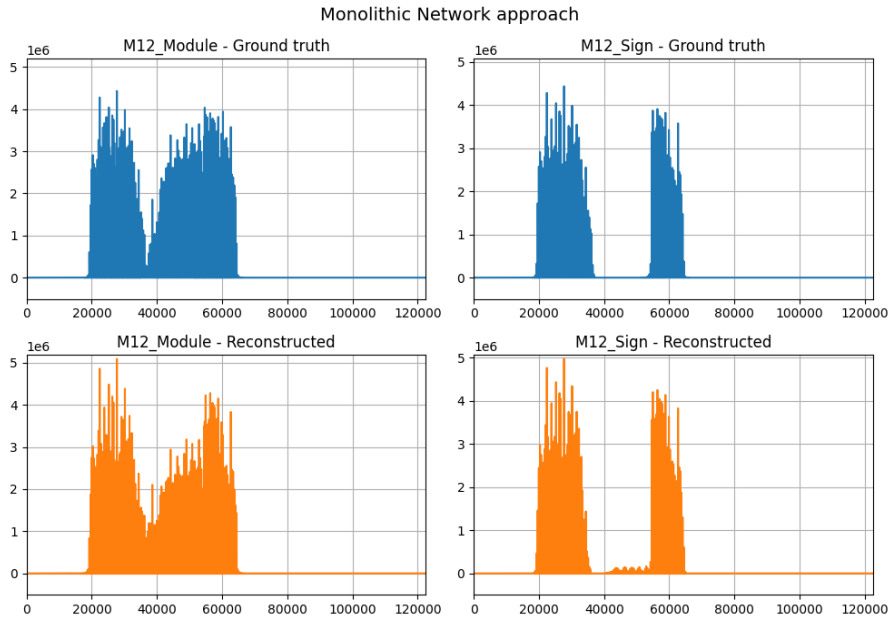


Figure 4.9: M_{12} transformation comparison

M_{11}, M_{22} and M_{12}^{mod} all have similar norms when compared, presenting a similar number of indices with relevant values, while M_{12}^{sgn} tends to have a norm which is approximately half when compared with the others. The difficulty in reconstruction could be credited to this imbalance, which is difficult to fully capture with a single set of trainable parameters.

After providing the output metric to the meshing algorithm of FreeFem++, the reconstructed mesh is returned, allowing from a direct comparison.

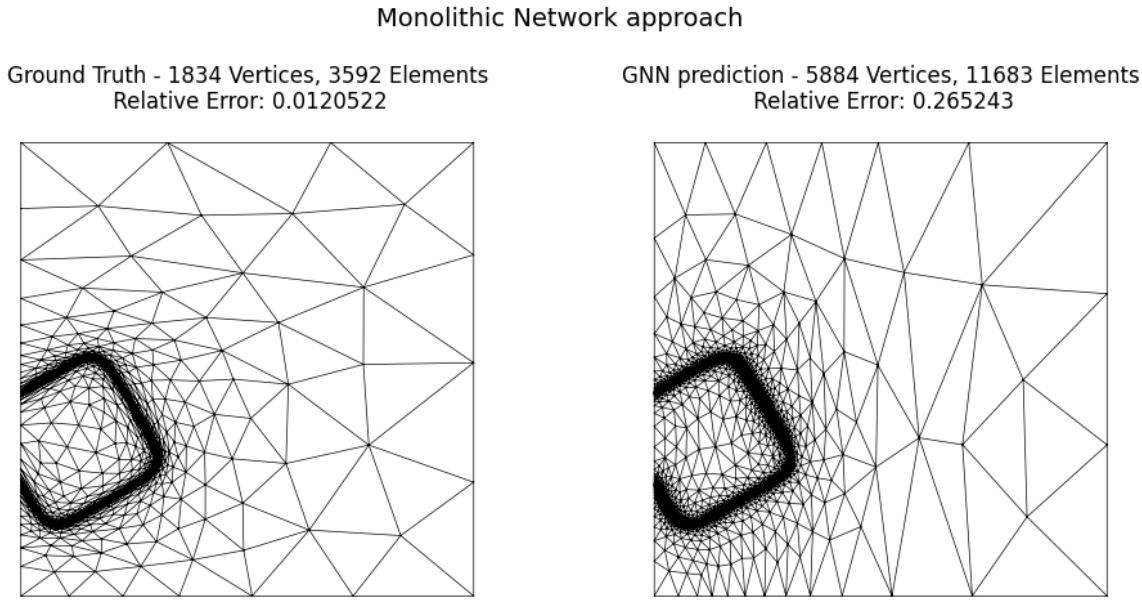
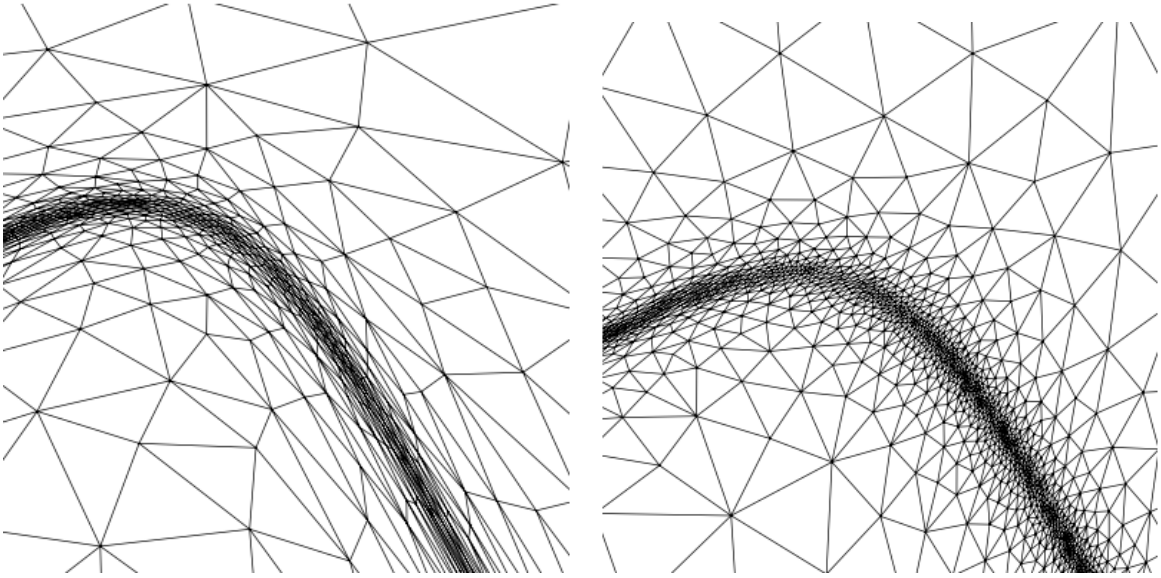


Figure 4.10: Meshes comparison

The reconstructed mesh clearly catches the shape of its original counterpart, but completely overshoots on the amount of vertices and elements. Furthermore, the relative error on the reconstructed mesh is more than one order of magnitude larger than the original one. Figure 4.11 zooms into the element-dense area, showing the inability of the monolithic approach to produce stretched elements, effectively leading to a worst numerical solution when the numerical problem is solved over the new mesh.



(a) Original (b) Reconstructed

Figure 4.11: Detailed view of the mesh comparison

4.4.3 Multi Network approach results

The Multi Network approach for the validation problem used four identical one block structures to attempt the reconstruction of the data. In order to pose a fair comparison between this methodology and the monolithic one, the data compression reached by the sum of the four independent networks is equivalent to the one achieved by the previous model.

	Node Features	Cardinality Reduction
Block 1	$1 \rightarrow 4$	80%

Table 4.2: Multi network single block architecture

Each of the four networks will increase the number of features from 1 to 4 via the Massage passing layer, and then the Graph Reduction layer will maintain only the top 20% of the nodes, based on their individual scores. As before, every network has been individually trained until a prolonged stagnation of the average loss over the validation set arose.

From the comparison of the original and reconstructed metrics, it is immediately clear how the reconstruction yielded better results than before. The plots suggest that the resulting mesh will be very accurate, but possibly not perfect due to a visible over-shoot in the estimation of the M_{22} component.

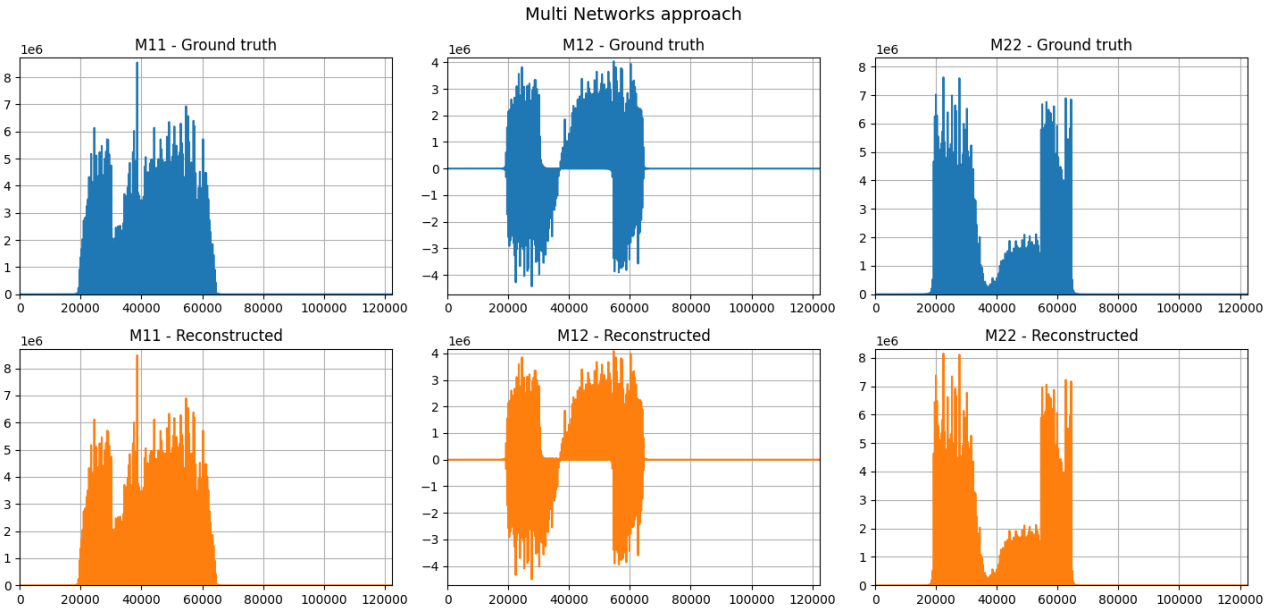


Figure 4.12: Metrics comparison

The partition of the training for M_{12}^{mod} and M_{12}^{sgn} solved the problem encountered in the monolithic approach, suggesting that the issue in the estimation of the previous model was indeed the inability of a single set of parameters to interpret fully the differences between the two transformation components.

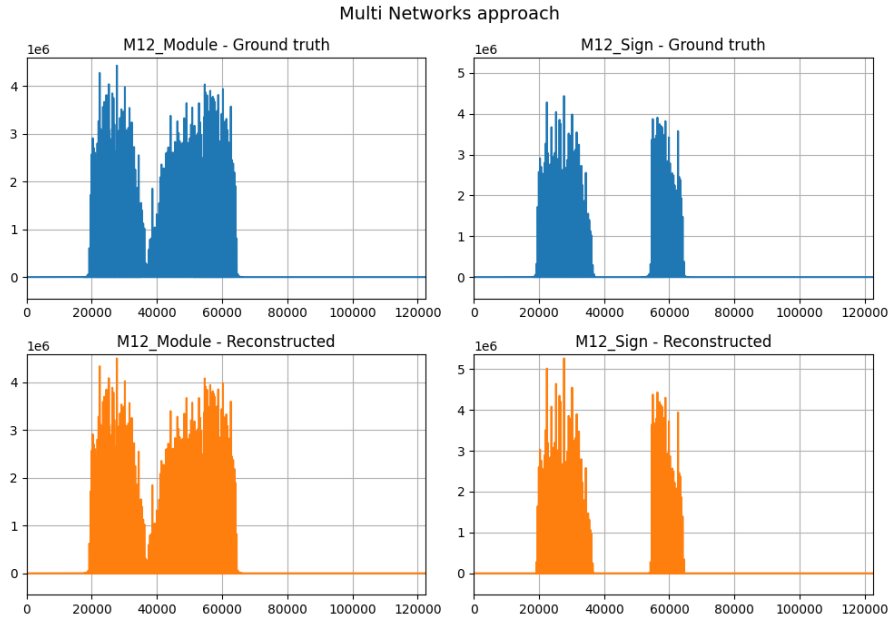


Figure 4.13: M_{12} transformation comparison

The mesh computed from the reconstructed metric still uses more vertices and elements when compared to the original one, but the numerical solution obtained by solving the validation problem over it produces a relative error, when computing the FEM problem over it, which is much closer to the original than the previous approach.

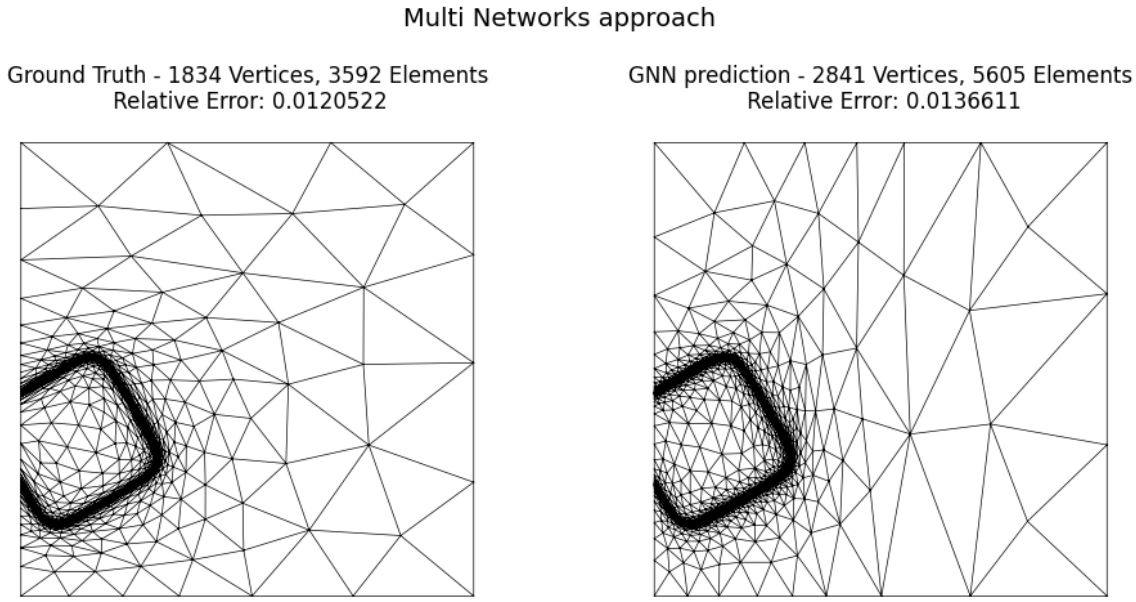
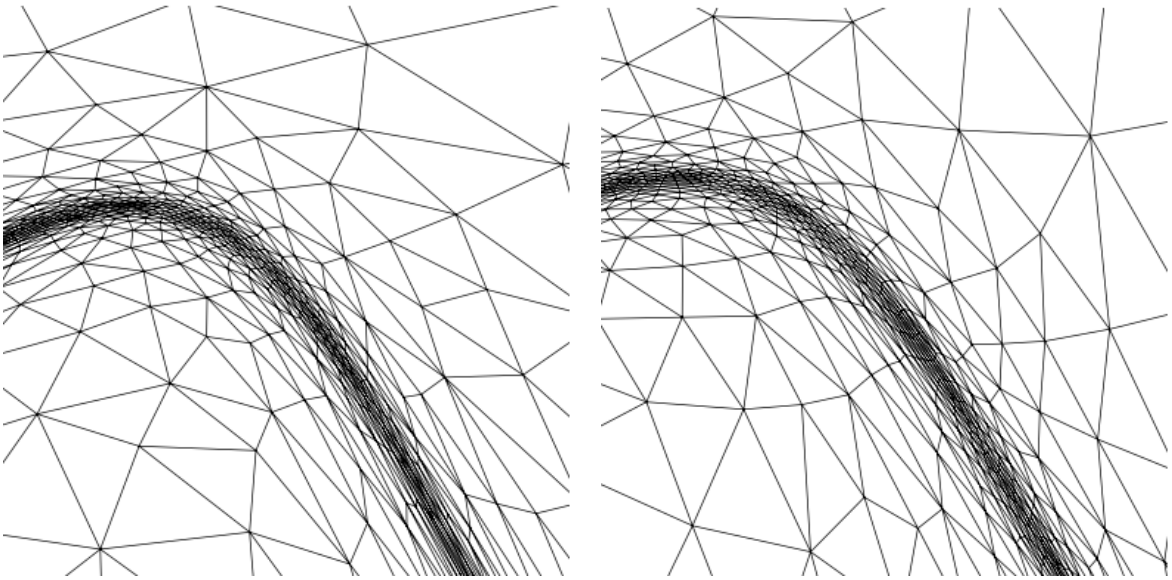


Figure 4.14: Meshes comparison

When zooming on the element-dense area of the meshes, it appears that also the reconstructed one has highly stretched triangles. Elements which are far from an isotropic configuration, requires the metric to be accurate in all three of the components, and this approach seems to be flexible enough to allow such accuracy during the reconstruction.



(a) (b)

Figure 4.15: Detailed view of the mesh comparison

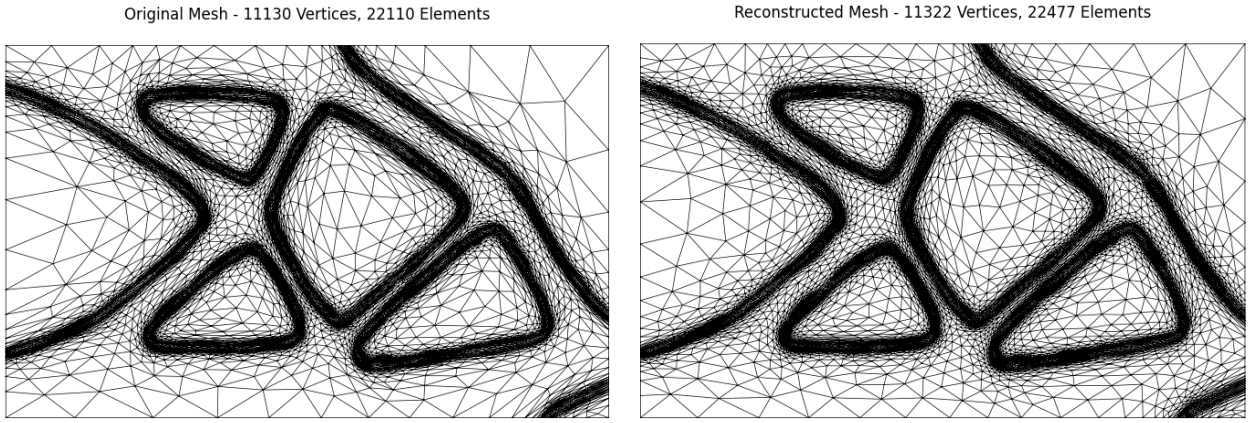
Chapter 5

Numerical Results for GNN-enhanced Topology Optimization

This chapter contains the results of the numerical experiments conducted while using a database of meshes arising from the optimization problem described in Chapter 2. Various set-ups are described, with different number of encoding-deconding blocks and different models that achieve various features dimensions and graph cardinality in the latent graph.

5.1 Mesh reconstruction from the metric

As detailed in Section 4.2, the reference mesh plays a central role in the accuracy that the GNN will be able to provide, even with perfectly reconstructed results.

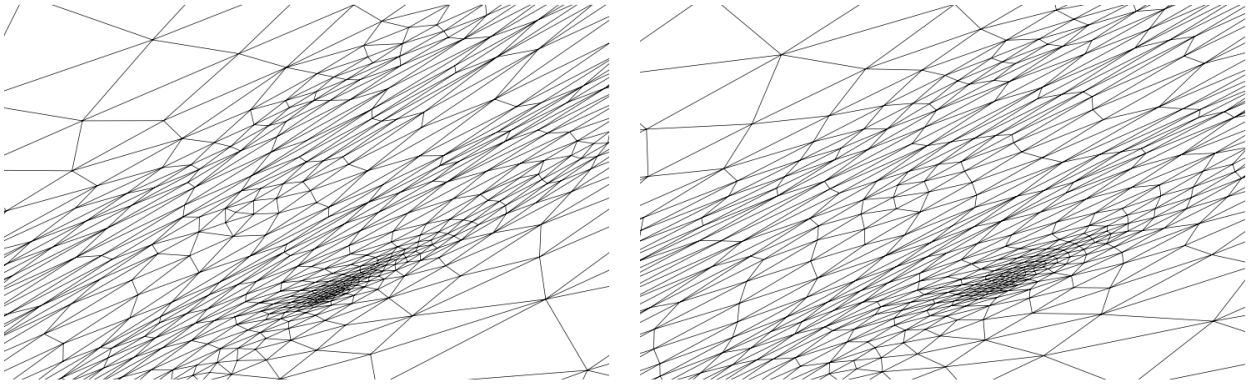


(a) (b)

Figure 5.1: Metric sampling with a 350×699 reference mesh

To tackle the problem of meshes arising from the topology optimization routine, a structured 350×699 mesh was chosen. This choice was done to maintain the similarity relation between the minimum edge length h_{min} of the original mesh, and the characteristic edge length h of the structured mesh.

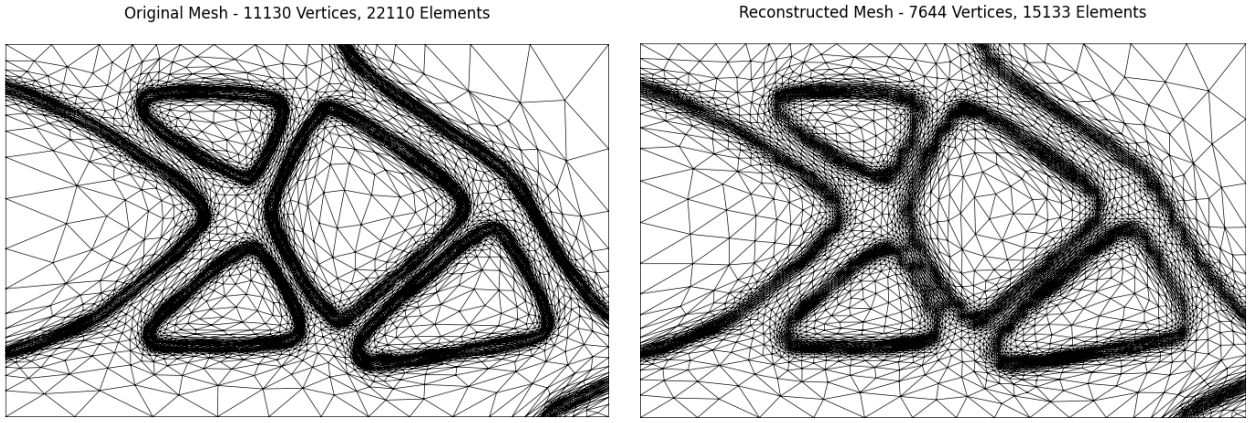
This sampling allow us to be able to capture even minute areas where the adaptive refinement produced element-dense configuration, as pictured in Figure 5.2.



(a) (b)

Figure 5.2: Detailed view of the 350×699 sampling

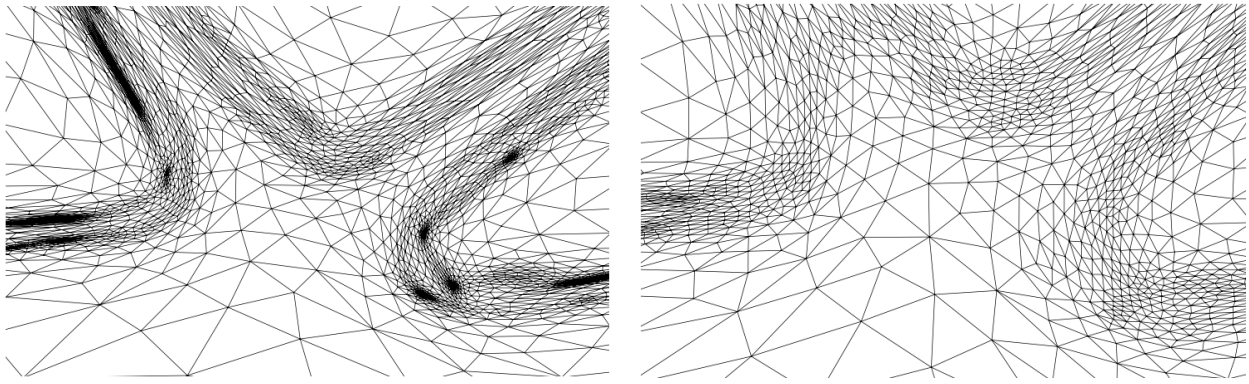
This choice comes with a noticeable computational cost, since the starting graph associated with the metric will have as many nodes as the reference mesh, that is, 244650 graph nodes. Unfortunately, lowering the refinement of the reference mesh will produce reconstruction of the mesh that looses many of the important features of the original one. An example case is brought in Figure 5.3.



(a) (b)

Figure 5.3: Metric sampling with a 100×199 reference mesh

The reconstructed mesh is visibly different from its original counterpart, presenting almost half of the vertices and a reduced number of elements as well. Key areas that had a very high level or refinement are lost. Clearly, the reference mesh used in this exemplification is not suitable to be the one used to store the information for the metric's database.



(a) (b)

Figure 5.4: Detailed view of the 100×199 sampling

5.2 The database

The database has been generated by computing 1674 different meshes, arising from an equivalent number of topology optimization cases. The domain considered consists in a 2×1 rectangle, constrained over its left side. The forcing factors have been imposed only on the right portion of the domain's boundary, which includes two horizontal and one vertical sections.

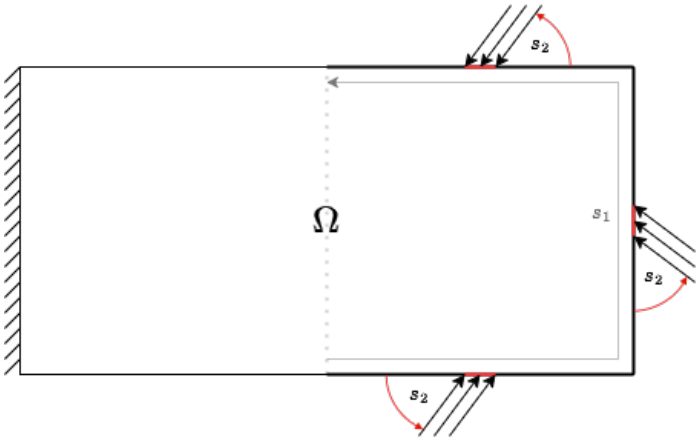


Figure 5.5: Graphic representation of the dataset cases

Each computed case had only a singular force acting with an angle on the boundary, over a small portion of one of the three available sections. Every interval over which the force is applied has width 1/16, dividing the boundary in 48 equal parts, numbered in counterclockwise order starting from the left end of the bottom section of the available boundary. All the computed cases can be uniquely identified by a combination of two parameters $\{p_1, p_2\}$. The first parameter p_1 determines where the acting force is applied, based on the parametric function s_1.

$$s_1 : [1, 48] \to \mathbb{R}^2 \quad \text{s.t.} \quad s_1(p_1) = \mathbb{I}_i \tag{5.1}$$

Where $\mathbb{I}_i$ represent the i-th subdivision of the boundary, according to the enumeration given before.
The second parameter $p_2 \in [0, 35]$ determines the angle used, with respect to the subdivision considered, to apply the force, following the function s_2.

$$s_2 : [0, 35] \to \mathbb{R} \quad \text{s.t.} \quad s_2(p_2) = 2 + p_2 \cdot 5 \tag{5.2}$$

s_2 returns the angle in degrees that is used to apply the forcing factor. The angle is considered in a counter-clockwise manner starting from the boundary's wall of each section, as depicted in Figure 5.5

The resulting database of solutions has been divided into training, validation and test cases accordingly a 80-10-10 scheme, precisely as the validation problem in Section 4.4.1

5.3 Results

Stemming from the results of Chapter 4 only a Multi Network approach is used to produce results. A random case was selected from the test partition of the database, and used to showcase the capabilities of each attempted set-up.

5.3.1 One encoding block

The first set-up tested, consisted of a single encoding-decoding block, in an attempt to obtain the same precision achieved in the equivalent case of the validation problem.

	Node Features	Cardinality Reduction
Block 1	$1 \to 4$	80%

Table 5.1: Single block architecture

The direct comparison of the input and output metrics of the GNN suggest that the corresponding mesh will be accurate, since all three of the components are visually similar in Figure 5.6.

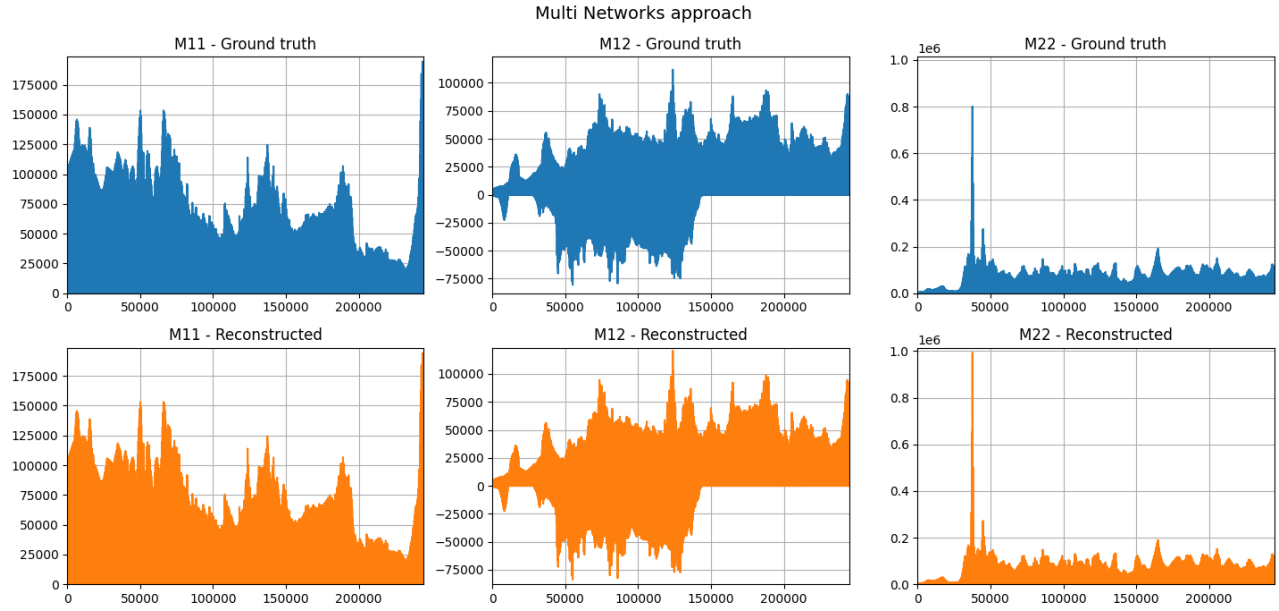


Figure 5.6: Metrics comparison

As seen for the validation problem, the presence of anisotropically stretched elements depends mostly on the accuracy of the M_{12} component. Comparing the transformation components M_{12}^{mod} and M_{12}^{sgn}, there are no visually relevant mistakes in Figure 5.7, hence implying the possibility of a high fidelity mesh as output.

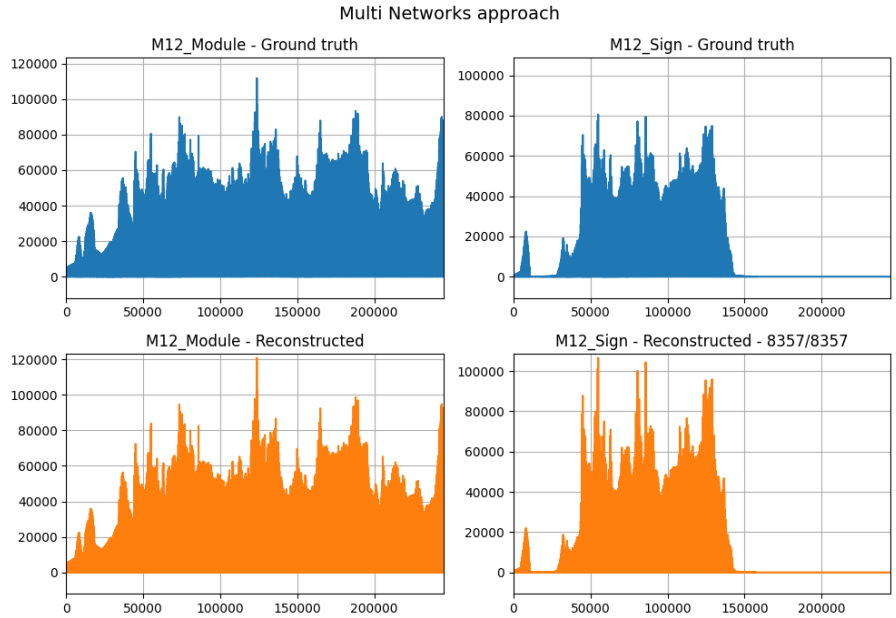


Figure 5.7: M_{12} transformation comparison

As expected, the mesh resulting from the reconstructed metric is remarkably similar to the original one, having only a slightly lower number of vertices and elements.

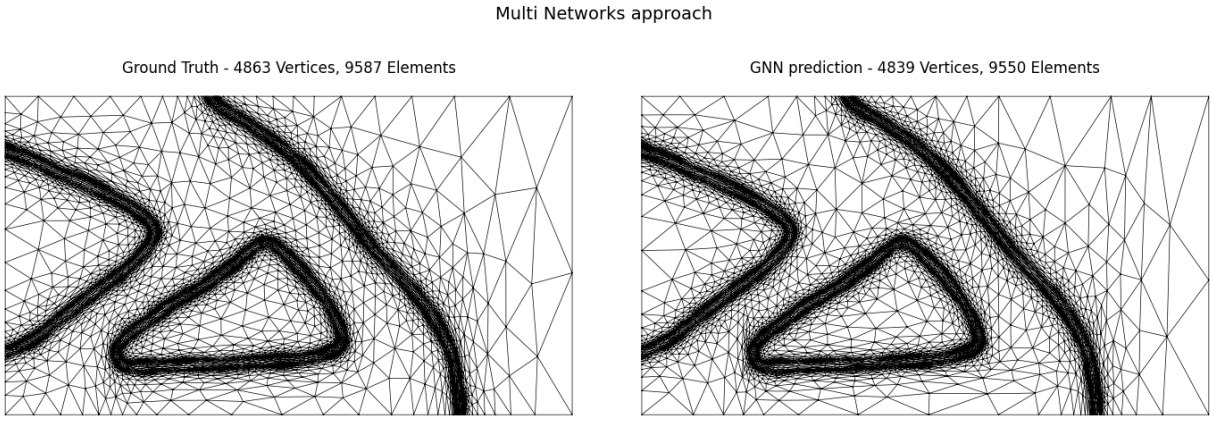
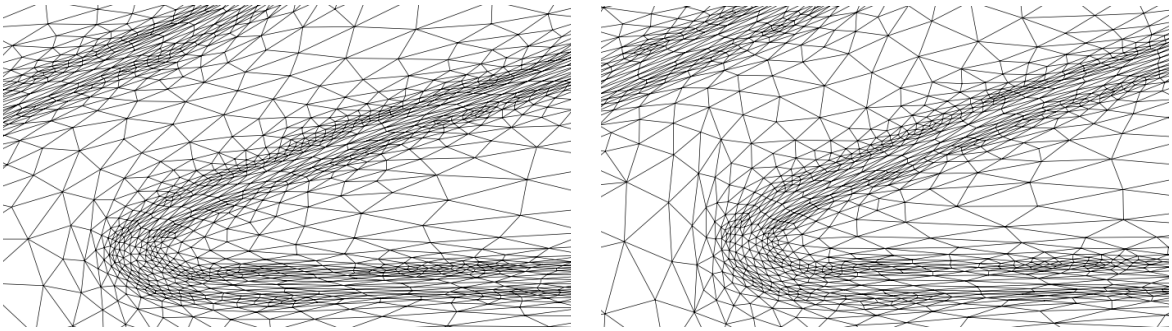


Figure 5.8: Meshes comparison

Zooming in, is possible to observe the similarities between the two meshes. Both of them have anisotropically stretched elements, and the reconstructed one is just as stretched just as the original one.



(a) (b)

Figure 5.9: Detailed view of the mesh comparison

The precision obtained in this example is noteworthy, however it is mandatory to recall the simplicity of the architecture used. In particular, this set-up reaches a mere 20% of information compression in the latent graph, which is still comparable with the amount of information given as input, hence compromising the usefulness of a single block set-up.

5.3.2 Three encoding blocks

When using multiple encoding-decoding blocks, several ways of equally compressing the input data arises. In this work, two opposite alternatives were tested, whilst keeping the final compression of the data equivalent, to prompt a fair comparison between them.

Model 1

The first of the two methodologies consists of removing a large percentage of nodes at every block, while simultaneously increasing greatly the amount of features in each graph node. The architecture considered can be described as follows:

	Node Features	Cardinality Reduction
Block 1	$1 \rightarrow 4$	80%
Block 2	$4 \rightarrow 9$	60%
Block 3	$9 \rightarrow 16$	50%

Table 5.2: Multiple blocks architecture n°1

The reconstructed metric presents some issues from the first comparison. In particular it seems that the visible lower values of the metrics, were not successfully estimated, or estimated as a constant. This behaviour can be observed in the initial indices of both M_{12} and M_{22}, as well as in the last indices of M_{11}.

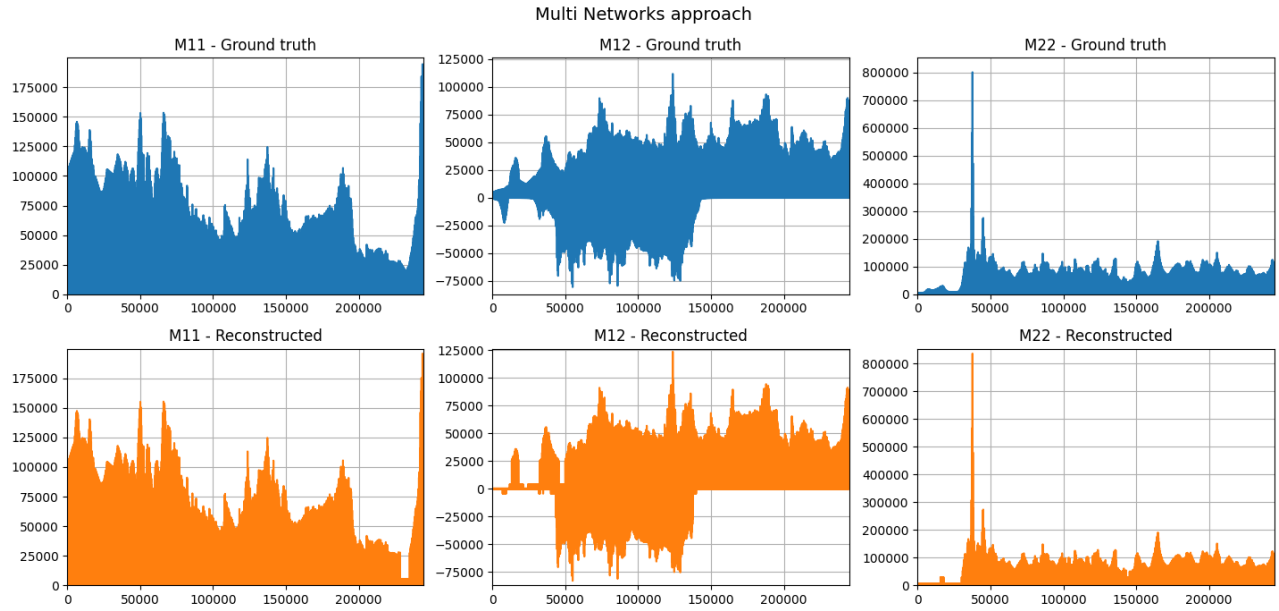


Figure 5.10: Metrics comparison

The transformation components shows that the inaccuracies in the M_{12} component are to be attributed to the estimation of M_{12}^{mod} rather than M_{12}^{sgn}.

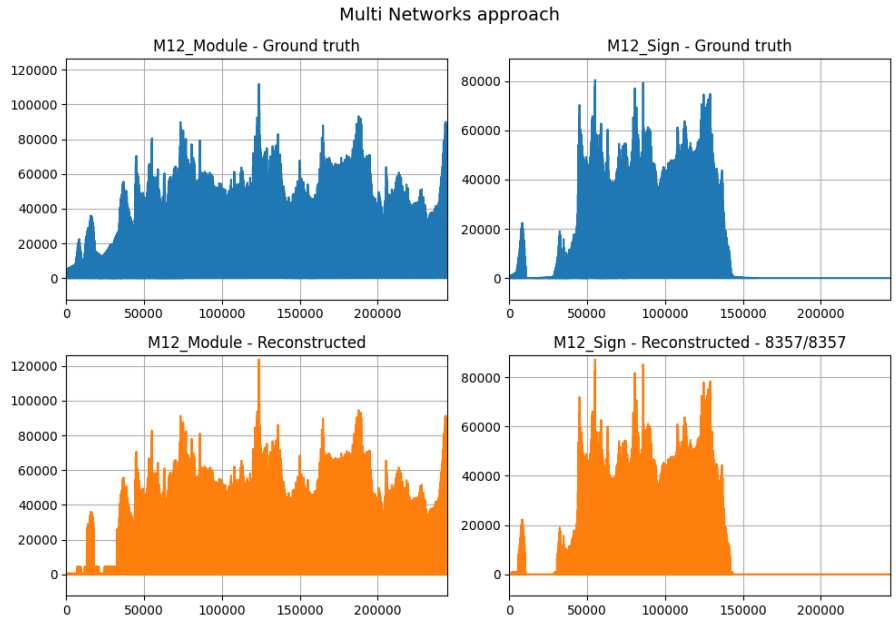


Figure 5.11: M_{12} transformation comparison

The reconstructed mesh captures the locations of the elements-dense ares on the domain, but overestimates the number of vertices and elements of the triangulation.

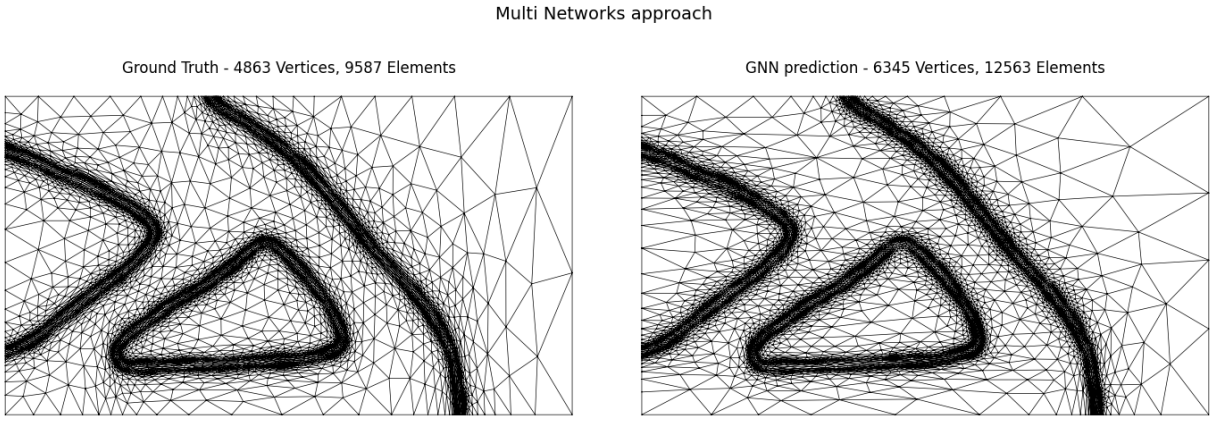
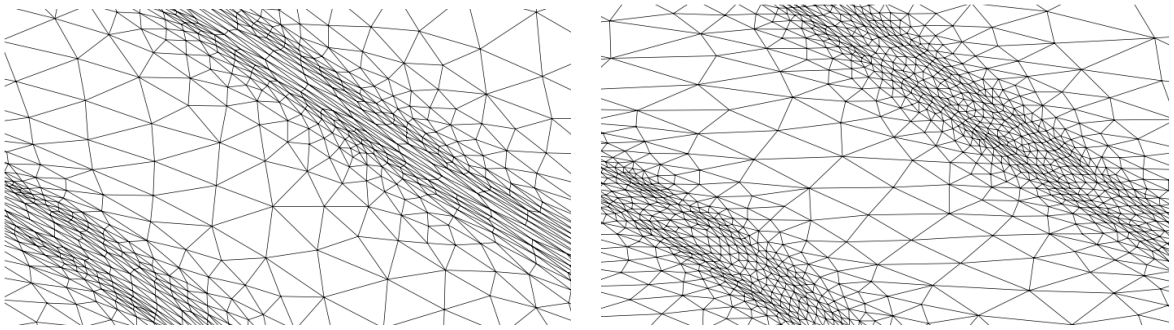


Figure 5.12: Meshes comparison

A detailed view shows the effects of the reconstructed metric's problems when building the mesh. The inaccuracies bring the meshing algorithm to produce more isotropic elements.



(a) (b)

Figure 5.13: Detailed view of the mesh comparison

Despite the inability of the architecture to reconstruct anisotropic elements, the mesh could be suitable to be the support of a Topology Optimization routine, given the capacity to correctly estimated the element-dense areas of the domain.

Model 2

The second viable option comprises of reducing the graph's cardinality slowly, as well as increasing the node features gradually. The achieved architecture is described in the table below.

	Node Features	Cardinality Reduction
Block 1	$1 \rightarrow 2$	50%
Block 2	$2 \rightarrow 4$	60%
Block 3	$4 \rightarrow 8$	60%

Table 5.3: Multiple blocks architecture n°2

The comparison between the original and reconstructed metrics in Figure 5.14 arises the same concerns of the previous set-up. It appears that the lower valued indices of the metric are the one most affected, being estimated at zero or at a constant value. When comparing with the previous case it seems like the problem is more contained, and possibly less prominent.

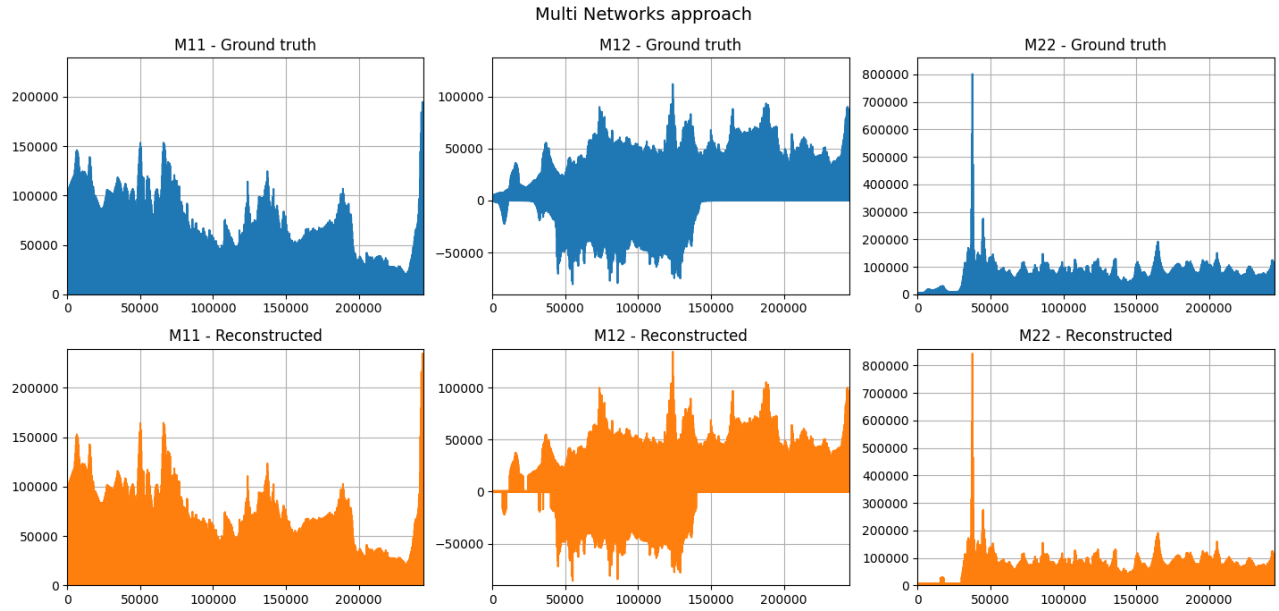


Figure 5.14: Metrics comparison

Once again it would seems that the most affected transformation component was M_{12}^{mod}, while M_{12}^{sgn} was able to estimate all the relevant indices required for the inverse transformation.

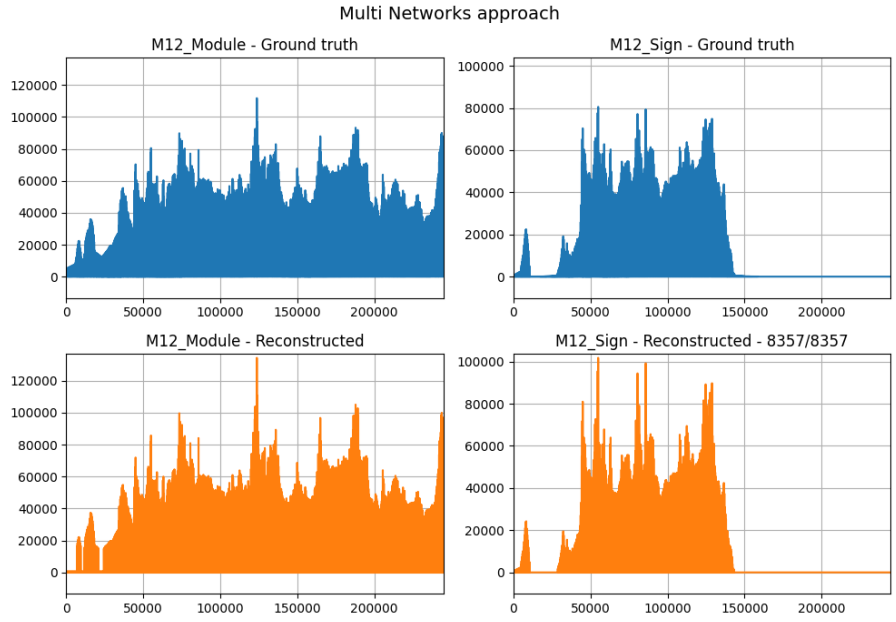


Figure 5.15: M_{12} transformation comparison

The mesh reconstructed from the output metric overestimates the number of vertices and elements. The location of the elements-dense areas is very accurate and there seems to be some areas where the elements are more stretched with respect to the previous architecture.

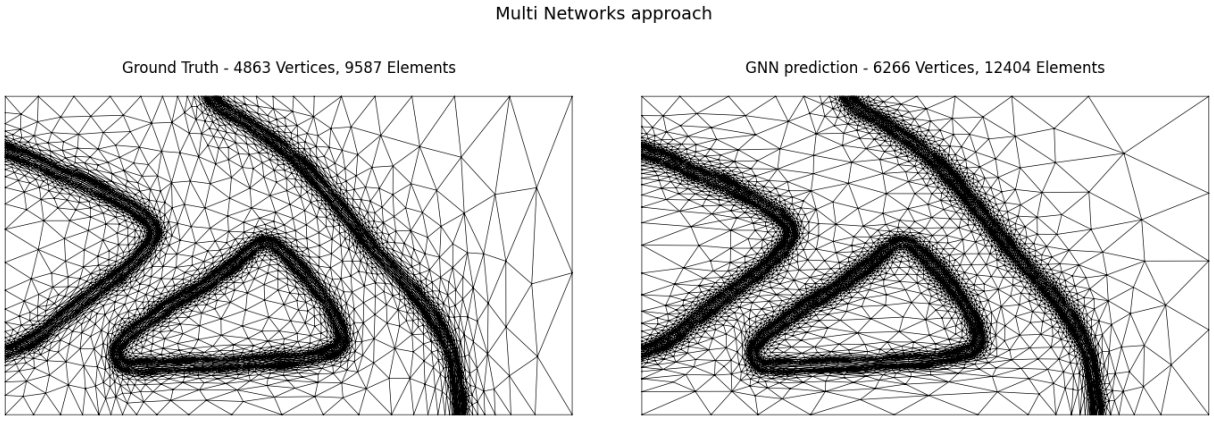
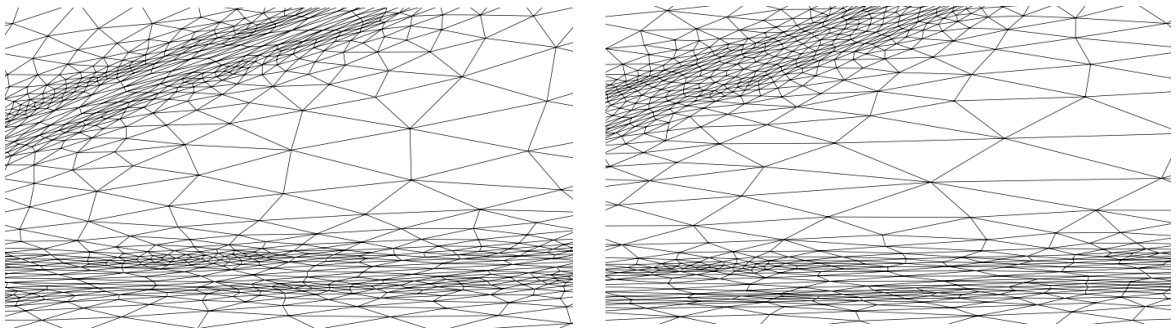


Figure 5.16: Meshes comparison

The reconstructed mesh presents stretched elements in the horizontal element-dense area, while having isotropic elements on the inclined ones. This result is due to the fact that horizontal and/or vertical stretching requires precision in less components than an oblique stretching. In fact to correctly estimate an horizontally stretched element, only M_{11} and M_{22} will have noticeable values in that point, while M_{12} will present a lower figure.



(a) (b)

Figure 5.17: Detailed view of the mesh comparison

All the presented case do not provide a quantitative comparison up to now, but only a qualitative comparison. In the following Section the resulting meshes will be integrated into a modified version of the Topology Optimization pipeline, allowing to understand in a quantitative manner the quality of the resulting meshes as support for optimization routines.

A problematic case

To prompt a more accurate comparison between the two three-blocks set-ups, a second case has been selected. A fraction of the test partition database returns less accurate results when tested. An example of the possible problematic results that the GNN could produce is presented below.

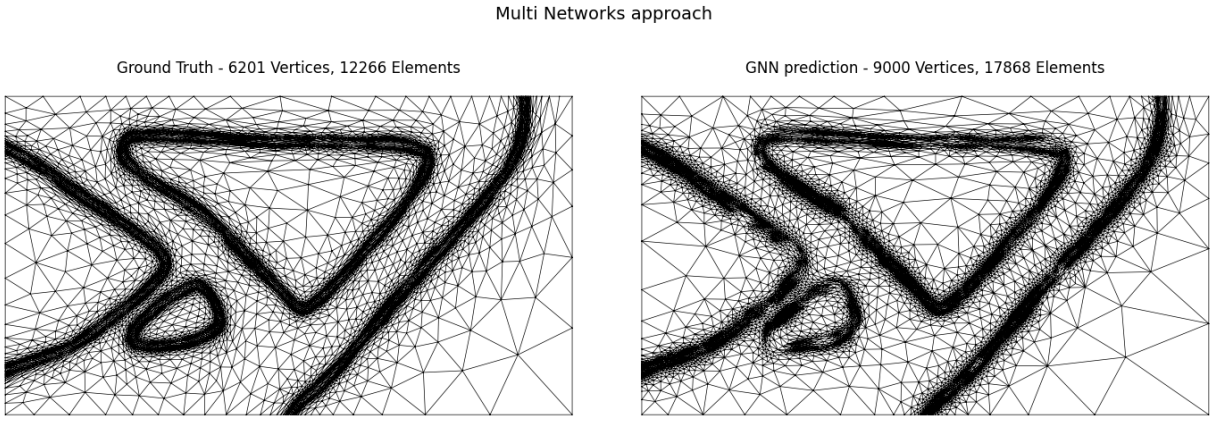


Figure 5.18: Meshes comparison - Model 1

The reconstructed mesh lacks some element-dense areas, is completely isotropic and has a number of vertices and elements that exceed the original one. When comparing the metrics, is not clear where the error lies, given the visual similarity between them. An accurate scanning of the database revealed that a limited group of cases presented very small areas of the domain, with elements stretched far more than the majority of the other cases computed. These areas appear in the metrics as very thin an high peaks, present in all the components, like the case under investigation shows in Figure 5.19.

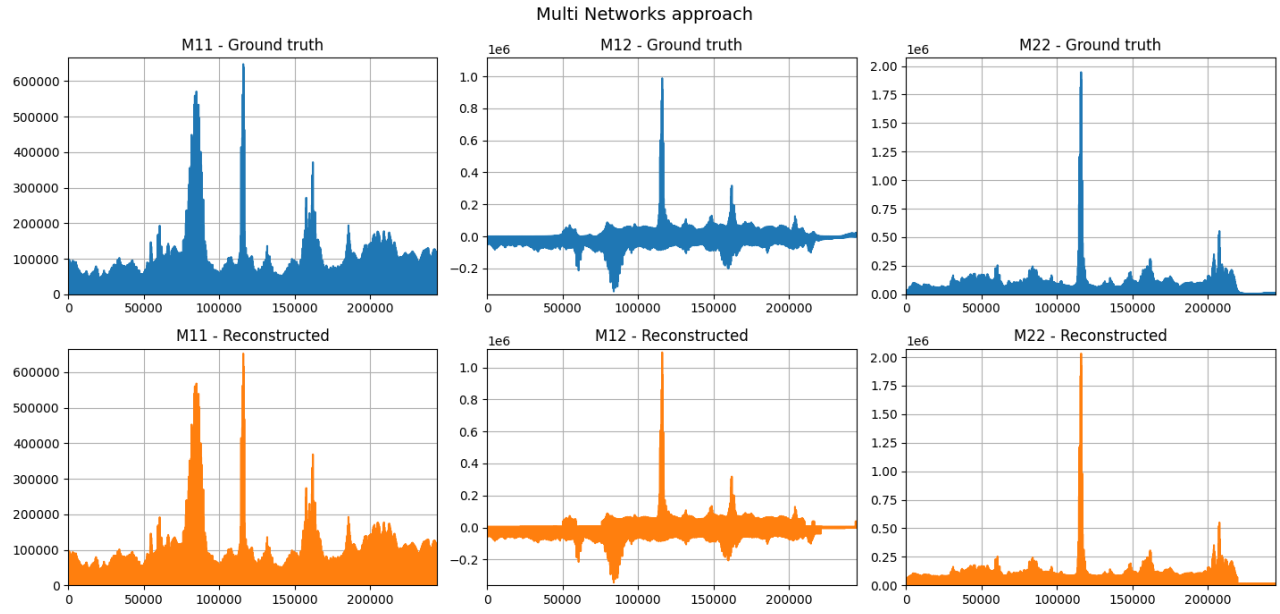


Figure 5.19: Metrics comparison

This problem could be solved by enlarging the available number of cases in the database. Clearly a finer selection of the nodes to discard in each block could mitigate the problem, and in fact the second multi-block methodology is found to produce more accurate reconstructions of the original meshes when this problem arises, as depicted in the below Figure.

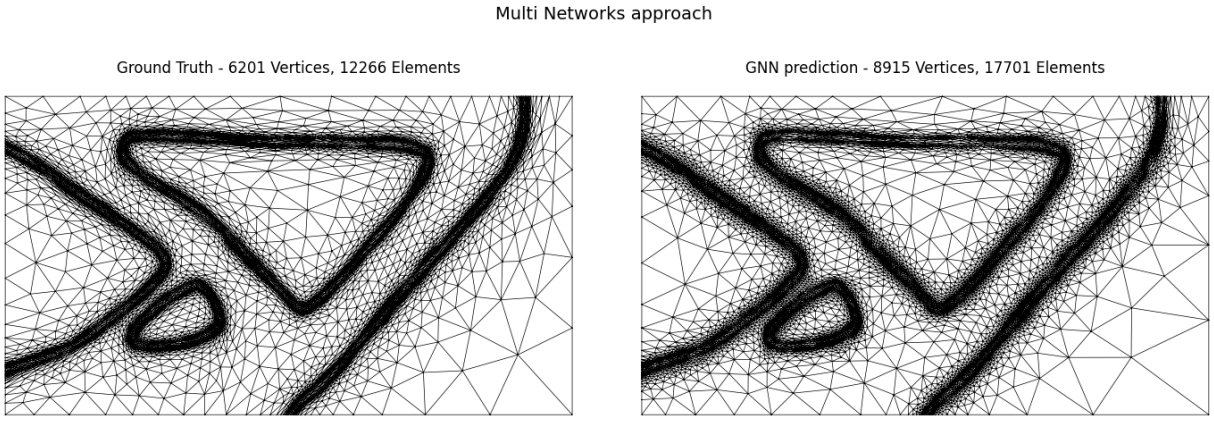


Figure 5.20: Meshes comparison - Model 2

It has to be noted that there is a small percentage of cases that work better while using the first model, but overall the last set-up has a higher number of cases that do not show problems. The fraction of non-satisfy cases amount to 15% of the test partition of the database with Model 1 and about 10% when using Model 2

5.3.3 Computational speed up

The reconstructed meshes presented in the previous Section, were used inside a modified version of Algorithm 2, which is presented below:

Algorithm 3 Modified Topology Optimization algorithm

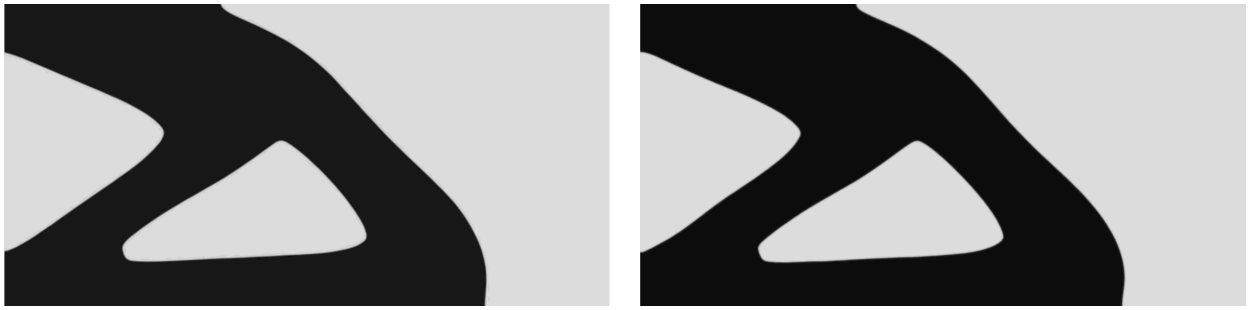
Construct structured computational mesh $\mathcal{T}_h$
Initialize ϕ_0^h
while $\|\phi_{k+1}^h - \phi_k^h\|_{\mathcal{L}^\infty(\Omega)} >$tol **do**
  Solve elastic problem $\rightarrow \boldsymbol{u}_{k+1}^h$
  Compute sensitivity $\rightarrow \mathcal{J}'(\phi_k^h, \boldsymbol{u}_{k+1}^h)$
  Solve RDE $\rightarrow \phi_{k+1}^h$
  Update parameters $\rightarrow \{k+1, q_{k+1}, r_{k+1}\}$
end while
Construct GNN mesh $\mathcal{T}_{nn}$
while $\nabla M_{nd} < 0$ and $M_{nd} > 1\%$ **do**
  Solve elastic problem $\rightarrow \boldsymbol{u}_{k+1}^h$
  Compute sensitivity $\rightarrow \mathcal{J}'(\phi_k^h, \boldsymbol{u}_{k+1}^h)$
  Solve RDE-DWP $\rightarrow \phi_{k+1}^h$
  Update parameters $\rightarrow \{k+1, q_{k+1}, r_{k+1}, \kappa_{k+1}\}$
end while
while $M_{nd} > 1\%$ **do**
  Construct gradient reconstruction $\rightarrow \boldsymbol{P}_{\Delta_K}(\nabla\phi_k^h)$
  Estimate optimal eigenpairs $\rightarrow \{\lambda_{i,K}^\star, \boldsymbol{r}_{i,K}^\star\}_{i=1,2}$
  Compute optimal metric $\rightarrow \mathcal{M}^\star$
  Compute optimal mesh via `adaptmesh` $\rightarrow \mathcal{T}_{nn}^{k+1}$
  Solve elastic problem $\rightarrow \boldsymbol{u}_{k+1}^h$
  Compute sensitivity $\rightarrow \mathcal{J}'(\phi_k^h, \boldsymbol{u}_{k+1}^h)$
  Solve RDE-DWP $\rightarrow \phi_{k+1}^h$
  Update parameters $\rightarrow \{k+1, q_{k+1}, r_{k+1}, \kappa_{k+1}\}$
end while

The objective is to show that the RDE-DWP loop can be performed on the meshes received as output of the GNN, allowing the procedure to reduce the transitorial interface without adapting the triangulation. In the case in which the discretization will not allow a smooth reduction of M_{nd}, a third loop will start, where the mesh adaptation is present. The reduction of M_{nd} is considered not smooth when more than 3 consecutive iterations fail to reduce it beyond its lowest value achieved up to that point.

Figure 5.21: Final optimal configuration using Algorithm 2

The first loop inside both algorithms is identical, allowing to start the second loop with the same distribution of ϕ.

The meshes reconstructed from the GNN are used as support of the computation during the second loop. The quality of these meshes is tested by measuring how many iterations of the RDE-DWP it takes to reduce the transitional part of ϕ to less than 1% of the domain area, or up until Adaptive Mesh Refinement is required.



(a) Model 1 (b) Model 2

Figure 5.22: Final configuration obtained while using the GNN meshes during the optimization

	Algorithm 2	Alg. 3 + Model 1	Alg. 3 + Model 2
RDE-DWP	38	38	37
AMR	38	0	0

Table 5.4: Topology Optimization iterations with different algorithms

For the first case analyzed, not only both meshes where able to support the optimization procedure without visibly altering the final configuration, but the result was achieved without any mesh refinement iteration. The most computationally demanding part of the loop for Algorithm 1, is clearly the mesh adaptation, since the RDE-DWP requires just to solve a linear system and update the parameters. The computational advantage in starting with a high quality mesh is quite apparent.

The same process is repeated for the problematic case, where the mesh is not completely reconstructed in its element-dense areas. The corresponding case computed with Algorithm 2 is shown below.

Figure 5.23: Final optimal configuration using Algorithm 2

The two final configurations obtained with Algorithm 3, are still compliant with the reference result, but in the case of Model 1, the solution interface in the lower part of the smaller hole, slightly creased.

(a) Model 1 (b) Model 2

Figure 5.24: Final configuration obtained while using the GNN meshes during the optimization

	Algorithm 2	Alg. 3 + Model 1	Alg. 3 + Model 2
RDE-DWP	45	41+4	43
AMR	45	4	0

Table 5.5: Topology Optimization iterations with different algorithms

Once again, the meshes produced by the GNN were able to speed up noticeably the computations of the optimization routine. In particular, the mesh produced by Model 2 allowed to reduce the gray area without any AMR iteration. The mesh produced by Model 1 had to enter the third loop and use adaptation to converge, but it still reduced the computational cost of the whole process noticeably.

5.4 Generalization over different problems

The last result being drawn is the capability of these architecture to generalize over data very different from the one included in the original database. To test this, a three-blocks architecture identical to the Model 1 of Section 5.3.2 was trained uniquely over the validation problem database, and later tested over the Topology Optimization dataset.

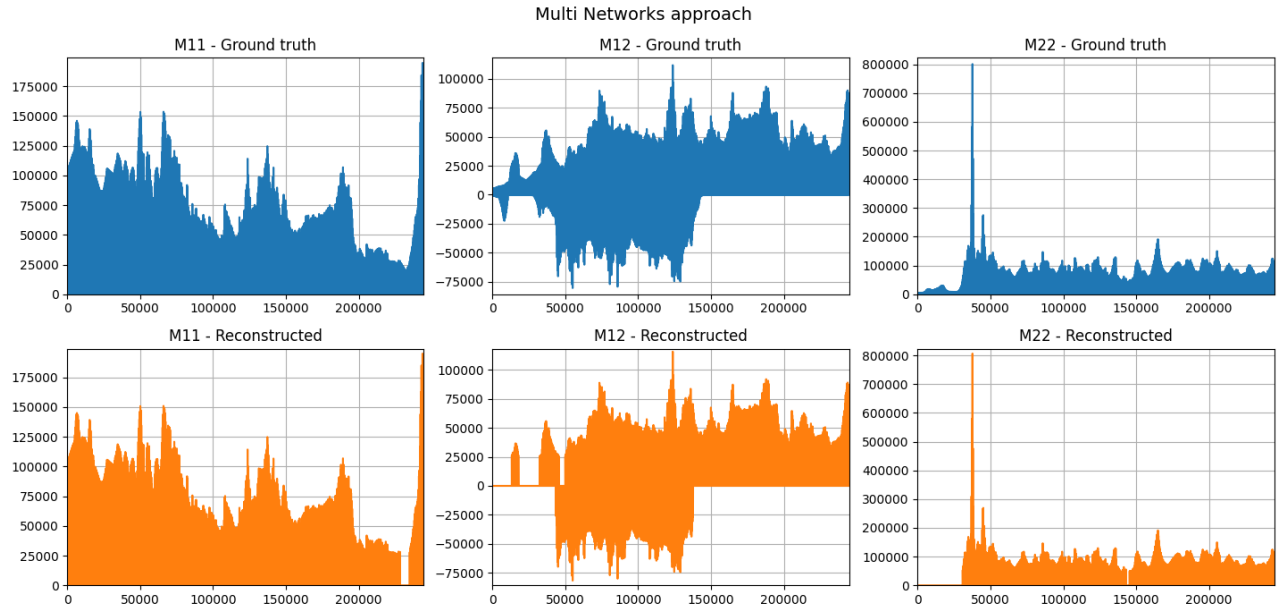


Figure 5.25: Metrics comparison

The comparison of the input and output metric is surprisingly adequate, with sections where the values where estimated at zero, but most of the metric structure seems to be present. The mesh reconstructed from the output of the GNN is also remarkably similar to the original one, capturing most of the locations of the element-dense areas.

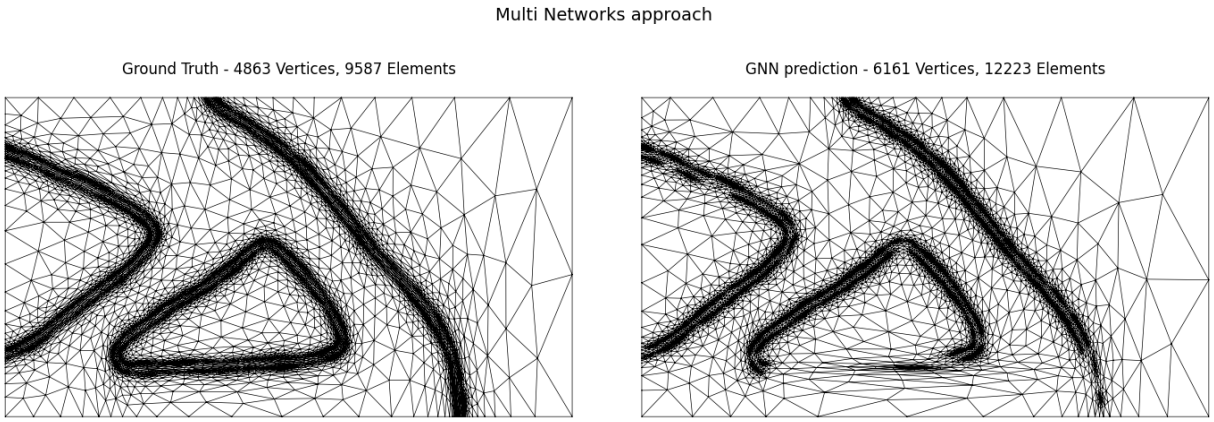
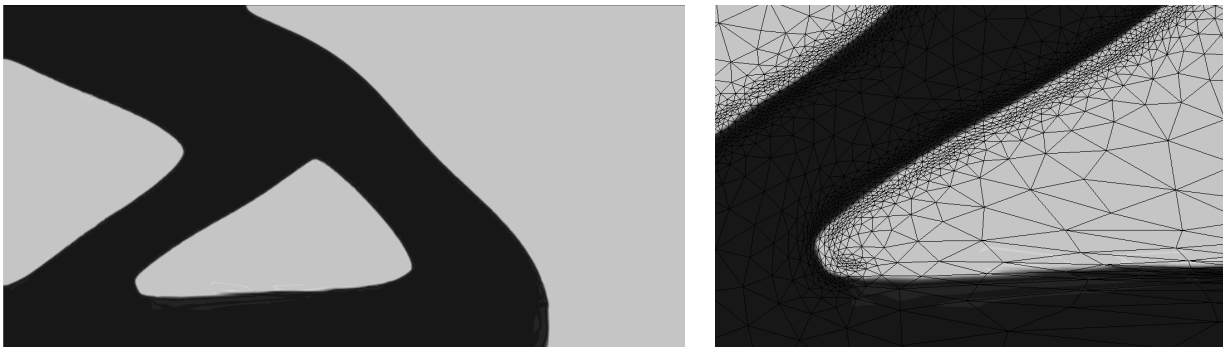


Figure 5.26: Meshes comparison

As for the previous set-ups, anisotropically stretched elements appears only in horizontal and vertical areas, while the oblique areas are mainly composed of isotropic elements.

The obtained mesh is inserted inside Algorithm 3, in order to quantify its quality as a support of a optimization routine.



(a) Optimal configuration computed

(b) Detailed view of defective area

Figure 5.27: Final configuration obtained while using the GNN mesh during the optimization

The routine converged to a grey area less than 1% without any need for mesh adaptation, using just 39 iterations of the RDE-DWP loop. This result was possible only thanks to the excessive over-meshing in the oblique interfaces, which allowed more gray area to be present in the poorly meshed part of the domain, as seen in Figure 5.27b

Chapter 6

Conclusions

In this work a new possible correlation between Topology Optimization and Machine Learning has been explored. Using Neural Network to speed up computational demanding tasks is not a new approach in the engineering world, but this work deep dived into the possibility of bringing this relation a step closer, by using Graph Neural Network to achieve its results. The correlation between Finite Elements Methods and Graph is inherent, but little to no application has been done so far to pursue industrial results, allowing this work to lay down the foundation to a new type of strategy to tackle computational demands.

Starting from an optimization pipeline, which was enriched using a state of the art Adaptive Mesh Refinement technique, a database of mathematically accurate results was built. This work revolved around the possibility of reconstructing meshes which were already adapted to the final solution, hence an efficient way to uniquely describe a triangulation has been used, the metric.

A novel Graph Neural Network architecture was built, inspired from recent works in the state of the art in Machine Learning, aiming to show the potential applications of graph based structures to treat mesh data. By associating the anisotropically stretched meshes resulting from the optimization routine to a graph through the metric, we were able to compress the starting graph representation into a latent one, whose dimension was considerably lower, then reconstruct the original data.

Multiple set-ups for this flexible architecture were developed and tested on a validation problem, which confirmed the capability of the GNN to accurately reproduce the objective meshes. When trained to the Topology Optimization database each model was able to obtain results accurate enough to allow the optimization procedure to successfully converge on them. It was shown how meshes created from a Neural Network were able to cut computational costs in a remarkable way when implemented inside the existing routines.

Lastly, the architecture showed a considerable capacity to generalize over new data never seen during the training, empirically proving the close correlation between the two fields. Additionally, these result suggest that future development of this work could be able to tackle a broad variety of problems, even with limited access to results arising from complex and expensive simulations.

6.1 Further developments

As of now, the developed Graph Neural Network focuses on compressing the information of the input metric into a smaller latent graph, to then reconstruct them. However, this GNN could be the starting point for a Generative Graph Neural Network (GGNN), which could predict quasi-optimal metrics by starting from a limited amount of problem parameters, e.g. the location and orientation of the applied forces.

Moreover, the focus of this work has been centered around the reconstruction of the mesh as a support for the optimal configuration, but not the configuration itself. It could be possible to improve even further the speed-up that the GGNN could guarantee, by starting the optimization from an accurate guess of the optimal configuration, allowing to reduce the computational costs in every part of the Topology Optimization pipeline described by Algorithm 2

Bibliography

[1] Grégoire Allaire, François Jouve, and Anca-Maria Toader. Structural optimization using sensitivity analysis and a level-set method. *Journal of computational physics*, 194(1):363–393, 2004.

[2] Daniel Rolando Araya Matilla. Enhancing topology optimisation of elastic structures via mesh adaptation. Master's thesis, Universitat Politècnica de Catalunya, 2023.

[3] Martin P Bendsøe and Ole Sigmund. Material interpolation schemes in topology optimization. *Archive of applied mechanics*, 69:635–654, 1999.

[4] Dimitri P Bertsekas. *Constrained optimization and Lagrange multiplier methods*. Academic press, 2014.

[5] Thomas Bolton and Laure Zanna. Applications of deep learning to ocean data inference and subgrid parameterization. *Journal of Advances in Modeling Earth Systems*, 11(1):376–399, 2019.

[6] Joan Bruna, Wojciech Zaremba, Arthur Szlam, and Yann LeCun. Spectral networks and locally connected networks on graphs. *arXiv preprint arXiv:1312.6203*, 2013.

[7] Sundeep Prabhakar Chepuri and Geert Leus. Subsampling for graph power spectrum estimation, 2016.

[8] Jae Seok Choi, Takayuki Yamada, Kazuhiro Izui, Shinji Nishiwaki, and Jeonghoon Yoo. Topology optimization using a reaction–diffusion equation. *Computer Methods in Applied Mechanics and Engineering*, 200(29-32):2407–2420, 2011.

[9] Michaël Defferrard, Xavier Bresson, and Pierre Vandergheynst. Convolutional neural networks on graphs with fast localized spectral filtering. *Advances in neural information processing systems*, 29, 2016.

[10] Andre Esteva, Brett Kuprel, Roberto A Novoa, Justin Ko, Susan M Swetter, Helen M Blau, and Sebastian Thrun. Dermatologist-level classification of skin cancer with deep neural networks. *nature*, 542(7639):115–118, 2017.

[11] Hongyang Gao and Shuiwang Ji. Graph u-nets, 2019.

[12] Matteo Giacomini and Simona Perotto. Anisotropic mesh adaptation for region-based segmentation accounting for image spatial information. *Computers & Mathematics with Applications*, 121:1–17, 2022.

[13] Pedro Gomes and Rafael Palacios. Aerodynamic-driven topology optimization of compliant airfoils. *Structural and Multidisciplinary Optimization*, 62:2117–2130, 2020.

[14] Will Hamilton, Zhitao Ying, and Jure Leskovec. Inductive representation learning on large graphs. *Advances in neural information processing systems*, 30, 2017.

[15] William L Hamilton. *Graph representation learning*. Morgan & Claypool Publishers, 2020.

[16] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Delving deep into rectifiers: Surpassing human-level performance on imagenet classification. In *Proceedings of the IEEE international conference on computer vision*, pages 1026–1034, 2015.

[17] Frédéric Hecht, Olivier Pironneau, A Le Hyaric, and K Ohtsuka. Freefem++ manual. *Laboratoire Jacques Louis Lions*, 2005.

[18] Quercus Hernández, Alberto Badías, Francisco Chinesta, and Elías Cueto. Thermodynamics-informed graph neural networks. *arXiv preprint arXiv:2203.01874*, 2022.

[19] Richard B Hetnarski and Józef Ignaczak. *The mathematical theory of elasticity*. CRC Press, 2016.

[20] Andreas Kamilaris and Francesc X Prenafeta-Boldú. Deep learning in agriculture: A survey. *Computers and electronics in agriculture*, 147:70–90, 2018.

[21] Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

[22] Thomas N Kipf and Max Welling. Semi-supervised classification with graph convolutional networks. *arXiv preprint arXiv:1609.02907*, 2016.

[23] Thomas N. Kipf and Max Welling. Semi-supervised classification with graph convolutional networks. *CoRR*, abs/1609.02907, 2016.

[24] Thomas N Kipf and Max Welling. Variational graph auto-encoders. *arXiv preprint arXiv:1611.07308*, 2016.

[25] Naoki Kishimoto, Kazuhiro Izui, Shinji Nishiwaki, and Takayuki Yamada. Optimal design of electromagnetic cloaks with multiple dielectric materials by topology optimization. *Applied Physics Letters*, 110(20), 2017.

[26] Yann LeCun, Yoshua Bengio, and Geoffrey Hinton. Deep learning. *nature*, 521(7553):436–444, 2015.

[27] Yann LeCun, Léon Bottou, Yoshua Bengio, and Patrick Haffner. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324, 1998.

[28] Yann LeCun, Léon Bottou, Genevieve B Orr, and Klaus-Robert Müller. Efficient backprop. In *Neural networks: Tricks of the trade*, pages 9–50. Springer, 2002.

[29] Heeseung Lim, Jeonghoon Yoo, and Jae Seok Choi. Topological nano-aperture configuration by structural optimization based on the phase field method. *Structural and Multidisciplinary Optimization*, 49:209–224, 2014.

[30] Stefano Micheletti and Simona Perotto. Reliability and efficiency of an anisotropic zienkiewicz–zhu error estimator. *Computer methods in applied mechanics and engineering*, 195(9-12):799–835, 2006.

[31] Stefano Micheletti and Simona Perotto. Anisotropic adaptation via a zienkiewicz–zhu error estimator for 2d elliptic problems. In *Numerical Mathematics and Advanced Applications 2009: Proceedings of ENUMATH 2009, the 8th European Conference on Numerical Mathematics and Advanced Applications, Uppsala, July 2009*, pages 645–653. Springer, 2010.

[32] Andreas C. Müller and Sarah Guido. *Introduction to Machine Learning with Python: A Guide for Data Scientists*. O'Reilly Media, 2016.

[33] Vinod Nair and Geoffrey E Hinton. Rectified linear units improve restricted boltzmann machines. In *Proceedings of the 27th international conference on machine learning (ICML-10)*, pages 807–814, 2010.

[34] Jyoti Narwariya, Pankaj Malhotra, Vishnu TV, Lovekesh Vig, and Gautam Shroff. Graph neural networks for leveraging industrial equipment structure: An application to remaining useful life estimation. *arXiv preprint arXiv:2006.16556*, 2020.

[35] GIOVANNI MICHELE Porta, Simona Perotto, and Francesco Ballio. Anisotropic mesh adaptation driven by a recovery-based error estimator for shallow water flow modeling. *International journal for numerical methods in fluids*, 70(3):269–299, 2012.

[36] Sebastian Ruder. An overview of gradient descent optimization algorithms. *arXiv preprint arXiv:1609.04747*, 2016.

[37] Ole Sigmund and Joakim Petersson. Numerical instabilities in topology optimization: a survey on procedures dealing with checkerboards, mesh-dependencies and local minima. *Structural optimization*, 16:68–75, 1998.

[38] Theofanis Strouboulis, Kevin Copps, and Ivo Babuška. The generalized finite element method. *Computer methods in applied mechanics and engineering*, 190(32-33):4081–4193, 2001.

[39] Akihiro Takezawa, Shinji Nishiwaki, and Mitsuru Kitamura. Shape and topology optimization based on the phase field method and sensitivity analysis. *Journal of Computational Physics*, 229(7):2697–2718, 2010.

[40] Evangelos Tyflopoulos, Flem David Tollnes, Martin Steinert, Anna Olsen, et al. State of the art of generative design and topology optimization and potential research needs. *DS 91: Proceedings of NordDesign 2018, Linköping, Sweden, 14th-17th August 2018*, 2018.

[41] Yulei Wu, Hong-Ning Dai, and Haina Tang. Graph neural networks for anomaly detection in industrial internet of things. *IEEE Internet of Things Journal*, 9(12):9214–9231, 2021.

[42] Takayuki Yamada, Kazuhiro Izui, Shinji Nishiwaki, and Akihiro Takezawa. A topology optimization method based on the level set method incorporating a fictitious interface energy. *Computer Methods in Applied Mechanics and Engineering*, 199(45-48):2876–2891, 2010.

[43] Xiao Xiang Zhu, Devis Tuia, Lichao Mou, Gui-Song Xia, Liangpei Zhang, Feng Xu, and Friedrich Fraundorfer. Deep learning in remote sensing: A comprehensive review and list of resources. *IEEE geoscience and remote sensing magazine*, 5(4):8–36, 2017.

[44] Olgierd Cecil Zienkiewicz and Jian Zhong Zhu. The superconvergent patch recovery and a posteriori error estimates. part 1: The recovery technique. *International Journal for Numerical Methods in Engineering*, 33(7):1331–1364, 1992.

[45] Olgierd Cecil Zienkiewicz and Jian Zhong Zhu. The superconvergent patch recovery and a posteriori error estimates. part 2: Error estimates and adaptivity. *International Journal for Numerical Methods in Engineering*, 33(7):1365–1382, 1992.