Physics-Informed Neural Networks for Topology Optimization

JAKOB DRUSANY

CCS Concepts: • **Applied computing** → *Computer-aided design*; • **Mathematics of computing** → *Mathematical optimization.*

Additional Key Words and Phrases: Topology optimization, Physics informed neural network, Machine learning

ACM Reference Format:
Jakob Drusany. 2025. Physics-Informed Neural Networks for Topology Optimization. 1, 1 (May 2025), 4 pages.

1 Motivation

With a recent increase in the availability of additive manufacturing, topology optimization is no longer a method only for industrial use, but also for everyday users. Physics-informed neural networks (PINNs) have become a powerful tool for solving systems of differential equations and replacing finite element analysis. The goal of this project is to implement a PINN that can be used to solve topology optimization problems from the PINNTO paper by Jeong and Bai et al. [3]. For the topology optimisation problem, the Solid Isotropic Material with Penalization (SIMP) method is used, where we replace the FEA step with our PINN.

2 Related work

Using physics informed neural networks in topology optimization (TO) [2] is very attractive because it removes the need for ground truth data. Iterative methods are not strictly replaced by PINNs but sometimes combined for generating synthetic data or by first running a few iterations of an iterative solver and then have the neural network refine the result [6]. Most of the related work focuses on building a good framework for solving the differential equations or how to make use of measurements. The problem is formulated as determining for each point in space, if the material is there or not. The discrete nature of the problem is addressed using the Solid Isotropic Material with Penalization method (SIMP) [1]. This method is used in many finite element solvers, but can also be used in a PINN framework [3]. This PINN-based topology optimization (PINNTO) uses PINN to approximate the strain energy field while enforcing material distribution constraints. By combining it with SIMP, it can adds or remove materials systematically, so the residual of the structure evolves towards an optimum. Training time can be reduced by dinamically adjusting the trainable parameters based on the optimization state of the TO [8].

Author's Contact Information: Jakob Drusany, jd5454@student.uni-lj.si.

ACM XXXX-XXXX/2025/5-ART
https://doi.org/

3 Theoretical setup

3.1 Problem description

First we have to define the problem formulation. First we have to know the properties of the material we are optimising. For this we are given the Young's modulus E in Pascals and Poisson's ratio ν. Then we are given a design domain Ω and volume fraction v_f. Our target design has to be a subset of this domain whose volume must be equal to v_f times the volume of the domain, i.e. $|\Omega_{target}| = v_f|\Omega|$. Then a list of fixed points in either direction is given and a list of external forces. Under these conditions, we have to optimize the topology of the domain to support the loads with minimum compliance. In figure 1 we can see two such examples. In the first example, every point on the left side is fixed only in the x direction and the bottom right point is fixed in the y direction. The second problem has all points on the left and right side fixed in both directions.

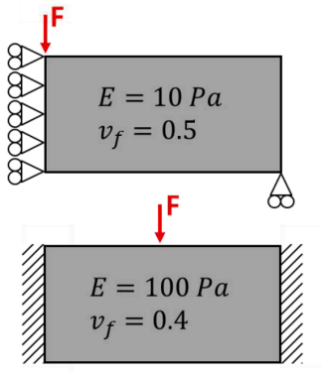

Fig. 1. Two example problems [3]

The solution of our program is a density distribution on the design domain representing the proportion of the material at the corresponding point in the domain. These proportions should be 0 and 1 representing where the material is present and where it is not.

3.2 Solid Isotropic Material with Penalization method

The problem is solved using the Solid Isotropic Material with Penalization method where it is assumed that the material is isotropic. Note that the problem is discretized into finitely many elements but a more formal discretization will be described later. We are minimizing the compliance C of the system under the conditions that the Hook's law between the displacements $\mathbf{U}$ and external forces $\mathbf{F}$ holds $\mathbf{KU} = \mathbf{F}$, where $\mathbf{K}$ is the global stiffness matrix and the volume fraction of the target density distribution ρ is v_f:

$$\begin{aligned} \min \quad & C(\rho) = \mathbf{U}^T\mathbf{KU}; \\ \text{s.t.} \quad & \mathbf{KU} = \mathbf{F}, \\ & V(\rho)/V_0 = v_f, \\ & 0 \le \rho_i \le 1, \quad i = 1...N, \end{aligned}$$

To solve this, a pipeline is used where we chose a density distribution that is homogeneous $\rho = V(\rho)/V_0$. From this density, we calculate the Young's modulus for each point p_i by the following penalization scheme

$$E_i^*(p_i) = E_{\min} + (E - E_{\min})\rho_i^p \quad (1)$$

This ensures that the density values gravitate towards 0, 1 because of the diminishing return in the material's toughness if an intermediate value is chosen. p is the penalization parameter which is generally $p = 3$. This modulus is then used in the stiffness matrix $\mathbf{K}$ and the system $\mathbf{KU} = \mathbf{F}$ is solved, to obtain $\mathbf{U}$. Now the density is updated to reduce the overall compliance of the system (the details of the update function are not relevant for this project). This process is then repeated until the density converges. The pipeline is depicted in figure 2.

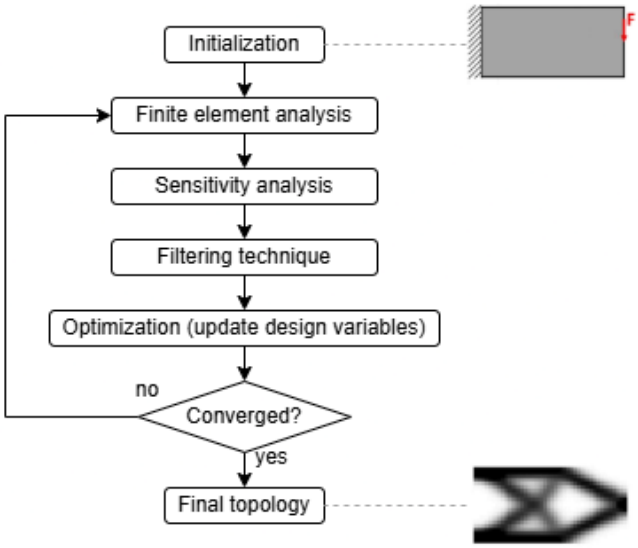

Fig. 2. The SIMP pipeline

3.3 Physics informed network overview

Our neural network does not replace the whole pipeline, only the Finite Element analysis part, depicted in figure 3. This is the part where the system $\mathbf{KU} = \mathbf{F}$ is solved, to obtain $\mathbf{U}$. Because of this, the matrix $\mathbf{K}$ does not need to be constructed.

For this subproblem, the network is tasked with creating a displacement solution u to our finite element problem. $u(x, y)$ is a function that assigns a displacement vector $[u_1, u_2]$ to each point (x, y) in the domain. The function $u(x, y)$ is approximated by a PINN depicted in figure 4.

There are 2 main approaches to calculate the physical loss function. The first is to solve the governing PDE, the second is to use the potential energy as a loss function according to the principle of minimum potential energy. We chose to use the energy-based approach, because it is computationally more efficient [5]. The subscripts α, β, and γ refer to the coordinate directions (these directions are not the actual directions but variables for directions). For a given output of the network u, the potential energy of the problem

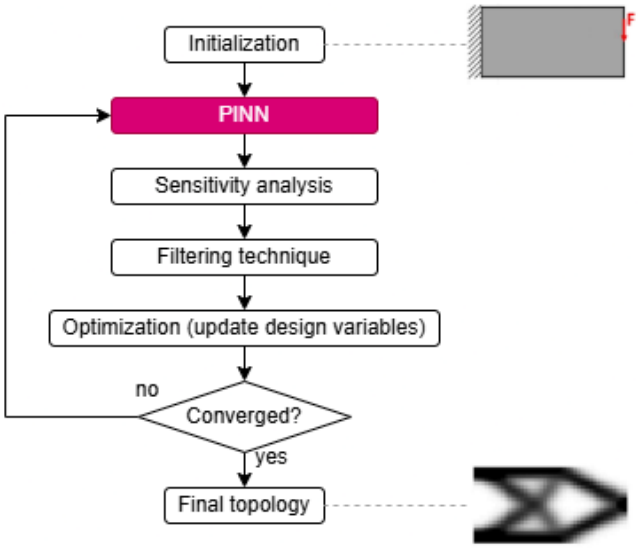

Fig. 3. The PINNTO pipeline

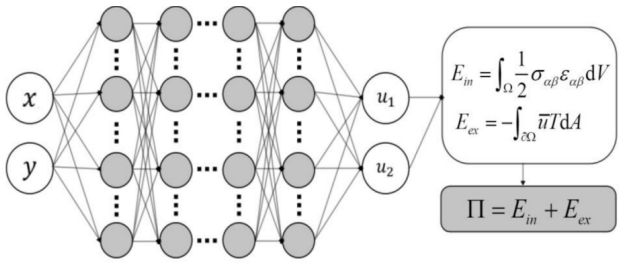

Fig. 4. Diagram of the PINN with coordinates x, y as inputs and the displacements u_1, u_2 as outputs with the potential energy as a loss function [3]

is $\Pi(u) = E_{in} + E_{ex}$. E_{in} is the internal strain energy and can be calculated as

$$E_{in} = \int_\Omega \frac{1}{2}\sigma_{\alpha\beta}\varepsilon_{\alpha\beta} dV$$

where σ and ε are the Cauchy stress and strain for the problem.

$$\varepsilon_{\alpha\beta} = \frac{1}{2}\left(\frac{\partial u_\alpha}{\partial x_\beta} + \frac{\partial u_\beta}{\partial x_\alpha}\right)$$

$$\sigma_{\alpha\beta} = \lambda\delta_{\alpha\beta}\varepsilon_{\gamma\gamma} + 2\mu\varepsilon_{\alpha\beta}$$

where $\delta_{\alpha\beta}$ is the Kronecker delta function. The Einstein summation has been applied here, in full these equations write out as:

$$E_{in} = \int_\Omega \frac{1}{2}\left(\sigma_{xx}\varepsilon_{xx} + \sigma_{yy}\varepsilon_{yy} + \sigma_{xy}\varepsilon_{xy} + \sigma_{yx}\varepsilon_{yx}\right) dV$$

Because for small strains and linear elasticity, the stress and strain tensors are symmetric ($\sigma_{xy} = \sigma_{yx}$, $\varepsilon_{xy} = \varepsilon_{yx}$). Because of this, we can simplify the equation

$$E_{in} = \int_\Omega \frac{1}{2}\left(\sigma_{xx}\varepsilon_{xx} + \sigma_{yy}\varepsilon_{yy} + 2\sigma_{xy}\varepsilon_{xy}\right) dV \quad (2)$$

The strain and stress equations are expanded in the following way:

$$\begin{aligned} \varepsilon_{xx} &= \frac{\partial u_x}{\partial x} \\ \varepsilon_{yy} &= \frac{\partial u_y}{\partial y} \\ \varepsilon_{xy} &= \frac{1}{2}\left(\frac{\partial u_x}{\partial y} + \frac{\partial u_y}{\partial x}\right) \end{aligned} \tag{3}$$

for the strain and

$$\begin{aligned} \sigma_{xx} &= \lambda(\varepsilon_{xx} + \varepsilon_{yy}) + 2\mu\varepsilon_{xx} \\ \sigma_{yy} &= \lambda(\varepsilon_{xx} + \varepsilon_{yy}) + 2\mu\varepsilon_{yy} \\ \sigma_{xy} &= 2\mu\varepsilon_{xy} \end{aligned} \tag{4}$$

for the stress. λ and μ are the Lamé parameters:

$$\begin{aligned} \lambda &= \frac{E^* \nu}{(1+\nu)(1-\nu)}, \\ \mu &= \frac{E^*}{2(1+\nu)} \end{aligned} \tag{5}$$

here E^* represents the Young's modulus that is calculated from the SIMP penalisation equation 1 and is different for each point in the computational domain.

To calculate the external energy E_{ex} (this is the potential energy corresponding to the external forces), we multiply the external force vector T with the corresponding displacement on the boundary $\overline{u}$ integrated over the boundary of the domain $\partial\Omega$

$$E_{ex} = -\int_{\partial\Omega} \overline{u} T dA \tag{6}$$

4 Implementation

For the implementation I used an already implemented SIMP pipeline and modified the code to include my PINN. The base repository is https://github.com/AJJLagerweij/topopt from the Master's thesis of Dr. van der Heijden [7]. I chose to use only the `src_Compliance` folder as it implements the compliance minimization.

4.1 Discretisation

The continuous domain Ω is discretized into a grid of finite elements of size (`nelx`, `nely`). Around the elements, a grid of nodes is formed. The elements are referred to as the design domain and the grid of nodes is referred to as the computational domain. The design domain holds the densities and the computational domain holds all other values of the problem (forces, displacements, boundary conditions).

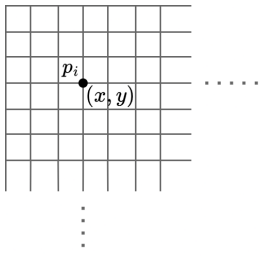

Fig. 5. Discretization into elements p_i and nodes (x, y)

Because the densities are in the design domain and are needed in the computational domain, to calculate the Lamé parameters in equation 5, a max pooling scheme is introduced depicted in Figure 6.

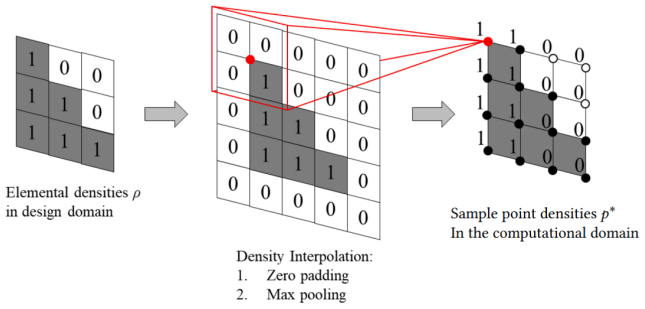

Fig. 6. Density interpolation scheme to calculate the densities in the computational domain [3]

4.2 Boundary conditions

The fixed points are represented as the Dirichlet boundary conditions on the displacement field. If a point is fixed in the x direction, the x displacement field u_1 must be 0 at the corresponding point; similarly, if it is fixed in the y direction, the y displacement field u_2 must also be zero at that point. The external forces are the Neumann boundary conditions on the displacement field , this is not enforced as it is already included in the E_{ex} equation.

4.3 The neural network

The neural network used to predict the displacements is a fully connected feed forward network with two inputs and two outputs. The hyperbolic tangent (Tanh) activation is applied on each hidden layer. The points of the domain are normalized into a $[0, 1] \times [0, 1]$ space. For the loss function the discrete version of the potential energy is used $\mathscr{L} = \Pi'(u)$. After the forward pass on a set of points, the u matrix is obtained. First we calculate the derivatives $\frac{\partial u_1}{\partial x}, \frac{\partial u_1}{\partial y}, \frac{\partial u_2}{\partial x}, \frac{\partial u_2}{\partial y}$ using torch autograd. Then all the ε and σ can be calculated using equations 3 and 4 respectively. This has to be done for every point, so the result is $\varepsilon_{i,xx}$, $\varepsilon_{i,xy}$, $\varepsilon_{i,yy}$ and $\sigma_{i,xx}$, $\sigma_{i,xy}$, $\sigma_{i,yy}$. We now approximate the integral E_{in} from equation 2 by calculating the sum over all points from the forward pass

$$E'_{in}(u) = \sum_i \frac{1}{2}\left(\sigma_{i,xx}\varepsilon_{i,xx} + \sigma_{i,yy}\varepsilon_{i,yy} + 2\sigma_{i,xy}\varepsilon_{i,xy}\right) \cdot S_i,$$

where S_i is the mean area of the corresponding collocation point. The potential energy of the external forces is calculated by multiplying the forces with the displacements:

$$E'_{ex}(u) = \sum_i (u_{1,i} \cdot T_{i,x} + u_{2,i} \cdot T_{i,y}) \cdot l_i,$$

where $T_{i,x}$ represents the external force on point i in the x direction, l_i is the mean interval of the sample points. This concludes the loss function $\Pi'(u) = E'_{in}(u) + E'_{ex}(u)$. The Dirichlet boundary conditions are imposed using the hard boundary condition scheme

proposed by A. Joglekar, et al. [4]. The output of the network is multiplied by a generalized logistics function such that the function is zero at $x = 0$ and goes to 1 rapidly

$$f(x) = 2 \cdot \left(\frac{1}{1 + e^{-mx}} - 0.5 \right)$$

the m parameter determines the slope, the proposed $m = 20$ is used. The actual multiplication is determined by the Dirichlet boundary; For example if the left boundary is fixed in the x and y direction and the right boundary is fixed only in the y direction, the following scheme is used:

$$u_1^*, u_2^* = NN(x, y)$$
$$u_1 = f(x)f(y) \cdot u_1^*$$
$$u_2 = f(1 - y) \cdot u_2^*$$

Note that there can be single points in the boundary scheme of the problem, these were ignored but should be implemented by a penalty factor in the loss function calculated at these isolated points.

5 Results

The results of our implementation show that there is an error somewhere in the code. Due to time constraints, the error was not eliminated and the results are skewed. The results are still interesting because they show the inherent unstable nature of the system. Figure 7 shows that network has learned the general shape of the expected output quite well, but the magnitude does not match, leading to a maximum error of approximately 20%. We assume that the area

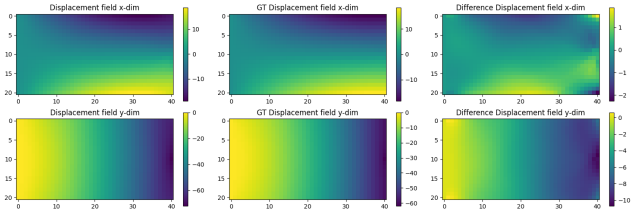

Fig. 7. Output of the network at pipeline iteration 0 compared the the FEA

$S_i = \frac{1}{(\texttt{nelx+1})(\texttt{nely+1})}$ and interval $l_i = \frac{2}{\texttt{nely+nelx+2}}$ were calculated inconsistently with the base implementation. This error 20% error resulted in a total failure downstream where the material was not even connected to the boundary forces as can be seen in Figure 8. The time performance of the framework is mostly unaffected by our error and shows that the framework is much slower than the base implementation using FEA. Our testing machine had a GTX1060 6gb GPU, i5-6600K 3.5GHz CPU and 16GB of RAM. The PINN was trained on the GPU while the FEA was performed on the CPU. Our network had 8 hidden layers size 80 and grid size 400×200. The PINN was trained for 1000 epochs. The base FEA implementation used 1.51s per iteration and our PINN used 162s per iteration. The total time can not be compared as the pipeline did not converge for the PINN version.

Fig. 8. The resulting design of our PINNTO implementation

6 Conclusion

We have shown that integrating Physics-Informed Neural Networks (PINNs) into the SIMP-based topology optimization pipeline is a highly delicate and challenging task, prone to instability and failure. While existing literature demonstrates the feasibility of this approach, it often comes at the cost of significant performance degradation. Therefore, although the incorporation of PINNs into topology optimization is technically possible, it should only be pursued when there is a compelling justification that clearly outweighs the associated complexities and computational drawbacks.

References

[1] M. P. Bendsøe. 1989. Optimal shape design as a material distribution problem. *Structural Optimization* 1, 4 (Dec. 1989), 193–202. doi:10.1007/bf01650949

[2] Martin P. Bendsøe and Ole Sigmund. 2004. *Topology Optimization.* doi:10.1007/978-3-662-05086-6

[3] Hyogu Jeong, Jinshuai Bai, C.P. Batuwatta-Gamage, Charith Rathnayaka, Ying Zhou, and YuanTong Gu. 2023. A Physics-Informed Neural Network-based Topology Optimization (PINNTO) framework for structural optimization. *Engineering Structures* 278 (2023), 115484. doi:10.1016/j.engstruct.2022.115484

[4] Aditya Joglekar, Hongrui Chen, and Levent Burak Kara. 2023. DMF-TONN: Direct Mesh-free Topology Optimization using Neural Networks. arXiv:2305.04107 [cs.CE] https://arxiv.org/abs/2305.04107

[5] Wei Li, Martin Z. Bazant, and Juner Zhu. 2021. A physics-guided neural network framework for elastic plates: Comparison of governing equations-based and energy-based approaches. *Computer Methods in Applied Mechanics and Engineering* 383 (2021), 113933. doi:10.1016/j.cma.2021.113933

[6] Ivan Sosnovik and Ivan Oseledets. 2019. Neural networks for topology optimization. *Russian Journal of Numerical Analysis and Mathematical Modelling* 34, 4 (2019), 215–223. doi:doi:10.1515/rnam-2019-0018

[7] Bram van der Heijden. 2019. *Topology Optimization for Damage Tolerance.* Ph. D. Dissertation.

[8] Jichao Yin, Ziming Wen, Shuhao Li, Yaya Zhang, and Hu Wang. 2024. Dynamically configured physics-informed neural network in topology optimization applications. *Computer Methods in Applied Mechanics and Engineering* 426 (2024), 117004. doi:10.1016/j.cma.2024.117004