ME782 Project

Mechanical Engineering

Manan Mehta(22b2129), Daksh Soni(22b2150)

Contents

Defining the Problem

1.1 The Problem Statement

This project will use topology optimisation to identify and remove redundant material from a 2D rectangular structure, significantly reducing its weight while maintaining structural integrity under various loads. The concept is analogous to designing a bicycle frame; instead of using a solid block of metal, material is placed only where it is needed to effectively distribute loads and enhance strength. Our approach will determine which regions within the rectangle can be eliminated without compromising its ability to withstand applied forces, leading to a highly efficient and lightweight design.

1.2 Background

In modern engineering, the drive to create lightweight yet durable structures is essential, especially in fields such as aerospace, automotive, and cycling. Reducing material usage without sacrificing strength leads to better performance, lower costs, and improved energy efficiency. Traditional design approaches often depend on experience or trial and error, which can leave unnecessary material in the structure. Topology optimisation, however, provides a scientific and computational method to achieve material efficiency. It systematically determines where material should be retained and where it can be removed, based on how loads and stresses flow through the structure.

A practical example of this concept can be seen in bicycle frame design. A solid metal block could easily withstand loads but would be unnecessarily heavy. Instead, engineers design frames with hollow tubes, placing material only along paths where forces travel between the pedals, seat, and wheels. These tubes form an efficient skeleton that provides stiffness and strength while keeping the frame lightweight. Similarly, in this project, topology optimisation will be applied to a 2D rectangular structure to identify regions that do not significantly contribute to load-bearing. By removing redundant material and maintaining essential load paths, the resulting structure will exhibit high strength-to-weight efficiency — much like the optimized form of a modern bicycle frame. To solve this problem, we'll look into Solid Isotropic Material with Penalisation techniques for removal of material.

The Solid Isotropic Material with Penalisation (SIMP) method is a widely used approach in topology optimisation to determine the optimal material distribution within a given design domain [?].

The core idea of SIMP is to represent the material properties of each finite element in the structure using a density variable, which can vary continuously between 0 (void) and 1 (solid material). This allows for a smooth transition between material and void, enabling the optimisation algorithm to explore a wide range of design possibilities.

1.3 Governing Equations

To solve the topology optimisation problem, we will use the Solid Isotropic Material with Penalisation (SIMP) method. The structure will be discretised into N finite elements. Each element will have a density variable x_e that can take values between 0 and 1, where 0 represents void (no material) and 1 represents solid material.

The domain consists of a rectangular grid of $nelx \times nely$ Q4 finite elements. Each element has an associated density variable $x_e \in [0, 1]$.

The structure is subjected to:

- Distributed supports (bottom edge fixed),
- Multiple external point loads at arbitrary nodes.

1.3.1 Design Variables

Each finite element has a design variable:

$$x_e \in [0,1], \qquad e = 1, \ldots, nelx \times nely$$

1.3.2 Objective Function

Compliance is computed using the FE displacement field u:

$$C = \sum_e x_e^p \, u_e^T K_0 u_e$$

where:

- p = penalization factor,
- K_0 = element stiffness matrix,
- u_e = displacement vector of element e.

1.3.3 Optimization Problem

$$\begin{aligned} &\min_x \quad C(x) \\ &\text{s.t.} \quad \frac{1}{N}\sum_e x_e = V^* \\ &0.001 \leq x_e \leq 1 \end{aligned}$$

1.3.4 Constraints

- Maximum allowed material volume V^*,
- Fixed displacement boundary conditions,
- Multiple external loads,
- Filtering radius r_{min} for density smoothing.

1.3.5 Assumptions

- Linear elasticity,
- Small displacements,
- Isotropic materials,
- Constant Young's modulus $E = 1$, Poisson ratio $\nu = 0.3$.

Methodology

2.1 Algorithm Overview

The implemented procedure consists of:

1. Initialization of density field,
2. Construction of Q4 FE stiffness matrix,
3. Assembly of global FE system,
4. Application of boundary conditions,
5. FE analysis using sparse solvers,
6. Sensitivity computation,
7. Density filtering,
8. OC density update,
9. Visualization of load/support locations and density evolution.

2.2 Filtering

A density filter is used to remove checkerboard patterns:

$$\tilde{d}_e = \frac{\sum_{i \in N_e} H_{ei} x_i d_i}{\sum_{i \in N_e} H_{ei}}$$

2.3 Optimality Criteria Update

The OC method updates densities with:

$$x_e^{new} = \max\left(0.001, \max(x_e - m, \min(1, \min(x_e + m, x_e\sqrt{-\frac{\partial C/\partial x_e}{\lambda}})))\right)$$

2.4 Detailed Algorithmic Implementation

The topology optimization code in our project directly follows the classical SIMP methodology but implements every component manually using NumPy and SciPy. In this section, we explain how each algorithmic step is achieved inside the Python implementation.

2.4.1 Initialization of the Density Field

The optimizer is initialized with a uniform material distribution:

$$x_e = \text{volfrac}, \qquad \forall e$$

In the code, this is:

```
self.x = np.ones(nely * nelx) * volfrac
self.xPhys = self.x.copy()
```

This sets every element in the domain to the same initial density.

2.4.2 Construction of the Q4 Element Stiffness Matrix

The Q4 isoparametric element stiffness matrix K_e is computed once in the constructor using analytical expressions:

```
k = np.array([1/2-nu/6, 1/8+nu/8, ... ])
self.KE = E/(1-nu**2) * np.array([...])
```

This 8×8 matrix corresponds to standard bilinear quadrilateral elasticity.

2.4.3 Assembly of the Global FE System

For each element the global degrees of freedom are computed using an `edofMat` array:

```
edofMat[el, :] = np.array([2*n1+2, 2*n1+3, ..., 2*n1+1])
```

The global stiffness matrix K is assembled using sparse COO storage:

```
iK = np.repeat(edofMat, 8).flatten()
jK = np.tile(edofMat, (1,8)).flatten()
sK = (self.KE.flatten() * xPhys**penal).flatten(order='F')
K = sp.coo_matrix((sK, (iK, jK)))
```

This implements:

$$K = \sum_e x_e^p K_e$$

2.4.4 Application of Boundary Conditions

The user can add supports anywhere via:

```
self.fixed_dofs.append(2*node_idx)
self.fixed_dofs.append(2*node_idx+1)
```

All fixed DOFs are removed from the global solve:

```
free = np.setdiff1d(np.arange(self.ndof), fixed)
u[free] = spsolve(K[free,:][:,free], F[free])
```

2.4.5 FE Analysis Using Sparse Solvers

The FE problem solves:

$$Ku = F$$

with the sparse solver:

```
u[free] = scipy.sparse.linalg.spsolve(...)
```

This ensures memory efficiency for the 20000×20000 stiffness matrix.

2.4.6 Sensitivity Analysis

The compliance contribution from each element is:

$$c_e = u_e^T K_e u_e$$

Implemented as:

```
ue = u[edofMat]
ce = np.sum(np.dot(ue, self.KE) * ue, axis=1)
```

The sensitivity is computed as:

$$\frac{\partial C}{\partial x_e} = -p x_e^{p-1} c_e$$

implemented via:

```
dc = -self.penal * (self.xPhys**(self.penal - 1)) * ce
```

2.4.7 Density Filtering

The filter matrix H is assembled once in `_prepare_filter()`. Filtering applies:

$$\tilde{d}_e = \frac{\sum_i H_{ei}\, x_i d_i}{\sum_i H_{ei}}$$

In code:

```
dc[:] = (H * (self.x * dc)) / Hs / np.maximum(0.001, self.x)
```

This smoothing ensures well-behaved sensitivities and removes checkerboard patterns.

2.4.8 Optimality Criteria (OC) Density Update

The OC method solves the constrained optimization via a binary search on λ:

$$x_e^{new} = \max(0.001, \max(x_e - m, \min(1, \min(x_e + m, x_e\sqrt{-dc_e/\lambda}))))$$

In code:

```
xnew = max(0.001,
           max(x - move,
               min(1, min(x + move, x * sqrt(-dc/lmid)))))
```

The search loop continues until the volume constraint is satisfied:

```
if sum(xnew) > volfrac * nelx * nely:
    l1 = lmid
else:
    l2 = lmid
```

2.4.9 Visualization and Plot Updates

The density field is reshaped and plotted each iteration:

```
im.set_data(-self.xPhys.reshape((nelx, nely)).T)
```

Loads and supports are drawn via scatter plots and arrows:

```
ax.scatter(..., marker='^')      # supports
ax.arrow(..., fx*scale, fy*scale) # loads
```

Plots update dynamically using:

```
fig.canvas.draw()
plt.pause(0.01)
```

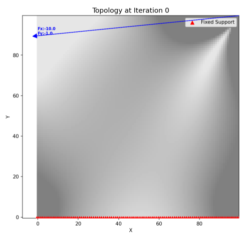
This produces real-time visualization of the evolving topology.

Results

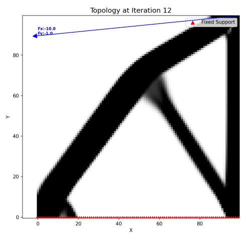
The optimizer was run for 50 iterations with the following conditions:

3.1 Topology Evolution

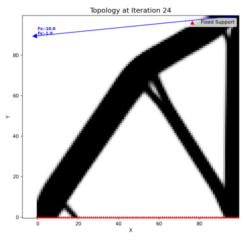
The figure below shows the evolution of the optimized topology over selected iterations:



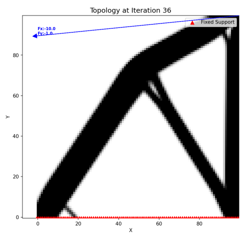
(a) Iteration 0



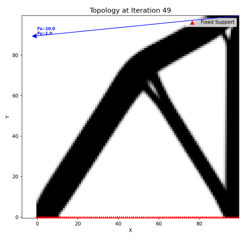
(b) Iteration 12



(c) Iteration 25



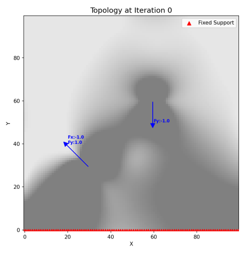
(d) Iteration 37



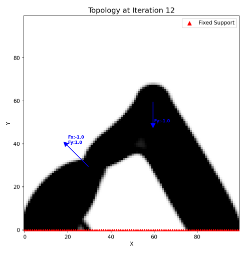
(e) Iteration 50

Figure 3.1: Topology optimization evolution showing material distribution across iterations. The dark regions represent retained material while light regions indicate removed material. The blue lines represent forces applied and red lines represent fixed loading for the structure

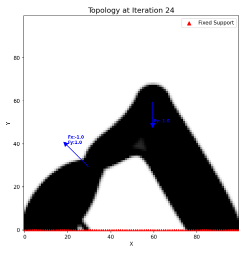
For another configuration, the topology evolution is shown below:



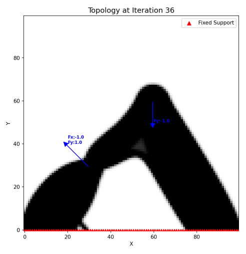
(a) Iteration 0



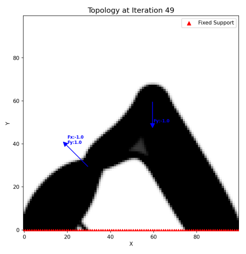
(b) Iteration 12



(c) Iteration 25



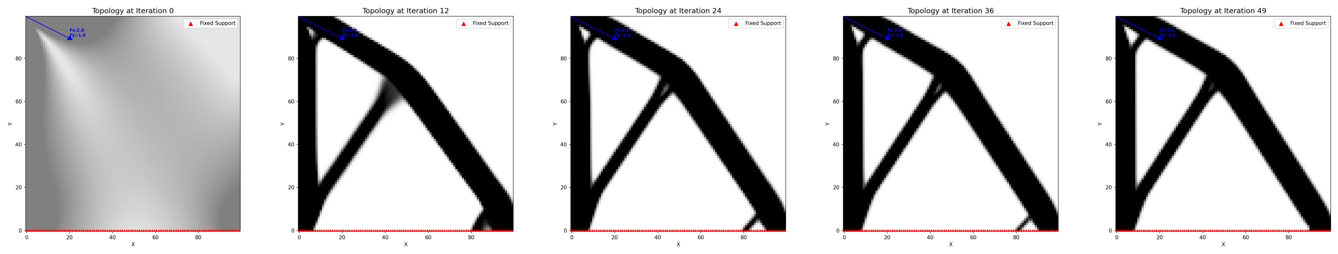
(d) Iteration 37



(e) Iteration 50

Figure 3.2: Topology optimization evolution showing material distribution across iterations. The dark regions represent retained material while light regions indicate removed material. The blue lines represent forces applied and red lines represent fixed loading for the structure

A different type of loading yields:



(a) Iteration 0 (b) Iteration 12 (c) Iteration 25 (d) Iteration 37 (e) Iteration 50

Figure 3.3: Topology optimization evolution showing material distribution across iterations. The dark regions represent retained material while light regions indicate removed material. The blue lines represent forces applied and red lines represent fixed loading for the structure

The figures illustrate the progression of topology optimization under different loading conditions across multiple iterations, beginning with a fully solid design domain and gradually evolving toward an optimized structural layout. As the algorithm proceeds, material is systematically removed from regions with low structural contribution, leaving dark areas that represent retained load-bearing material and lighter regions that indicate eliminated material; applied forces are shown by blue arrows and fixed boundaries by red lines.

In each configuration, the design converges by approximately iteration 37–50 to an efficient truss-like structure aligned with the primary stress paths, effectively transferring loads between supports while minimizing compliance and material usage. Although the evolutionary trends are similar among scenarios, differences in loading and boundary conditions lead to distinct final geometries, demonstrating the adaptability of topology optimization to varying structural requirements.

Overall, the results confirm the capability of topology optimization to generate lightweight and mechanically efficient structures by shaping material distribution according to stress flow. This highlights the strong relationship between boundary conditions, load paths, and optimal structural form.

Conclusions and References

Our project successfully demonstrates that topology optimization is a powerful approach for designing lightweight and mechanically efficient structures. By iteratively removing non-contributing material and aligning remaining regions with principal stress paths, the optimizer consistently converged to high-efficiency structural forms tailored to the imposed loading scenarios. The results validate the effectiveness of the SIMP method and optimality criteria update strategy in solving compliance-minimization problems, and highlight the strong dependency of optimal topology on boundary constraints and load locations. These findings reinforce the practicality of computational optimization tools in engineering design, offering systematic improvement over traditional intuition-based methods. Future work could expand to 3D domains, nonlinear materials, and multi-objective problems to further enhance applicability in real-world design applications. The 3d work could be used everywhere from designing drones to designing lightweight automotive components.

Contributions

- Manan Mehta (22b2129): SIMP formulation, FE assembly, equilibrium solver, sensitivity analysis, LaTeX documentation.
- Daksh Soni (22b2150): Optimization algorithm, density filtering, visualization, boundary conditions interface, project integration.

References

[1] Q. Wang, H. Han, C. Wang, et al. Topological control for 2d minimum compliance topology optimization using simp method. Structural and Multidisciplinary Optimization, 65(38):38, 2022.