МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

ЛЬВІВСЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ ІМЕНІ ІВАНА ФРАНКА

Факультет прикладної математики та інформатики
(повне найменування назва факультету)

Кафедра прикладної математики
(повна назва кафедри)

Магістерська робота

“РОЗВ’ЯЗУВАННЯ ЗАДАЧ ТОПОЛОГІЧНОЇ ОПТИМІЗАЦІЇ НА ОСНОВІ МЕТОДУ СКІНЧЕННИХ ЕЛЕМЕНТІВ”

Виконав: студент групи ПМпМ-22с
Дубей С.І.
спеціальності
113 Прикладна математика
(шифр і назва спеціальності)

Дубей С. І.
(підпис) (прізвище та ініціали)
Керівник доц. Ящук. О.Ю.

(підпис) (прізвище та ініціали)
Рецензент Шиманський В. М.

(підпис) (прізвище та ініціали)

Львів – 2022

ЛЬВІВСЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ ІМЕНІ ІВАНА ФРАНКА

Факультет прикладної математики та інформатики

Кафедра прикладної математики

Спеціальність 113 Прикладна математика
(шифр і назва)

«ЗАТВЕРДЖУЮ»
Завідувач кафедри Ящук Ю. О.
_____

" " 2021 року

ЗАВДАННЯ

НА МАГІСТЕРСЬКУ РОБОТУ СТУДЕНТОВІ

Дубею Святославу Ігоровичу
(прізвище, ім'я, по батькові)
(прізвище, ім'я, по батькові)

1.**Тема роботи** Розв'язування задач топологічної оптимізації на основі методу скінченних елементів

керівник роботи Ящук Юрій Олександрович, доцент кафедри прикладної математики, кандидат фізико-математичних наук
(прізвище, ім'я, по батькові, науковий ступінь, вчене звання)
затверджені Вченою радою факультету від "22" вересня 2021 року № 6

2. Строк подання студентом роботи 16 травня 2022 р.

3. Вихідні дані до роботи

1. *Yashchuk Yu. O. 1 , Tajs-Zielinska K. "*Solving topology optimization problems using cellular automata and mortar finite element method" / – Ivan Franko National University of Lviv, 1 Universytetska Str., 79000, Lviv, Ukraine 2, Cracow University of Technology, al. Jana Pawla II 37, 31-864, Cracow, Poland

2. *Савула Я. Г.* Числовий аналіз задач математичної фізики варіаційними методами./ – Львів: Видавничий центр ЛНУ імені Івана Франка, 2004. – 221 с.

3. *Hutton, David V*. Fundamentals of finite element analysis / – McGrawHill, 2004. – 494 p.

4. Зміст магістерської роботи (перелік питань, які потрібно розробити)

1. Дослідити задачу топологічної оптимізації та можливі алгоритми її вирішення
2. Дослідити задачу теорії пружності та обрати об’єкт дослідження
3. Обрати або створити алгоритм топологічної оптимізації
4. Розробити паралельну версію обраного/створеного алгоритму.
4. Створити програмну реалізацію однопотокового алгоритму
5. Створити програмну реалізацію того ж алгоритму з використанням паралелізму
6. Порівняти результати та час виконання

5. Перелік графічного матеріалу (з точним зазначенням обов'язкових креслень)

1. Зображення початкового об’єкту експеременту
2. Графічні зображення ітеративної зміни об’єкту
3. Графіки функції оцінки

6. Консультанти розділів роботи

Розділ	**Прізвище, ініціали та посада консультанта**	**Підпис, дата**	
		завдання видав	**завдання прийняв**

7. Дата видачі завдання 03.09.2021р.

Керівник роботи **Ящук Ю. О.**

(підпис) (прізвище та ініціали)

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів магістерської роботи	Строк виконання етапів роботи	Примітка
1.	Проведення теоретичних досліджень	Січень-Лютий 2022р.	
2.	Практична реалізація алгоритмів	Лютий-Квітень 2022р.	
3.	Оформлення роботи	Квітень-Травень 2022р.	

Dubei   **Дубей С. І.**

(підпис)   (прізвище та ініціали)

Зміст

Вступ

Завдяки методу скінченних елементів та сучасним обчислювальним можливостям стало можливим легко прогнозувати різного роду деформації об’єктів та різноманітну поведінку матеріалів не проводячи дорогі експерименти. (див. Рис 1)

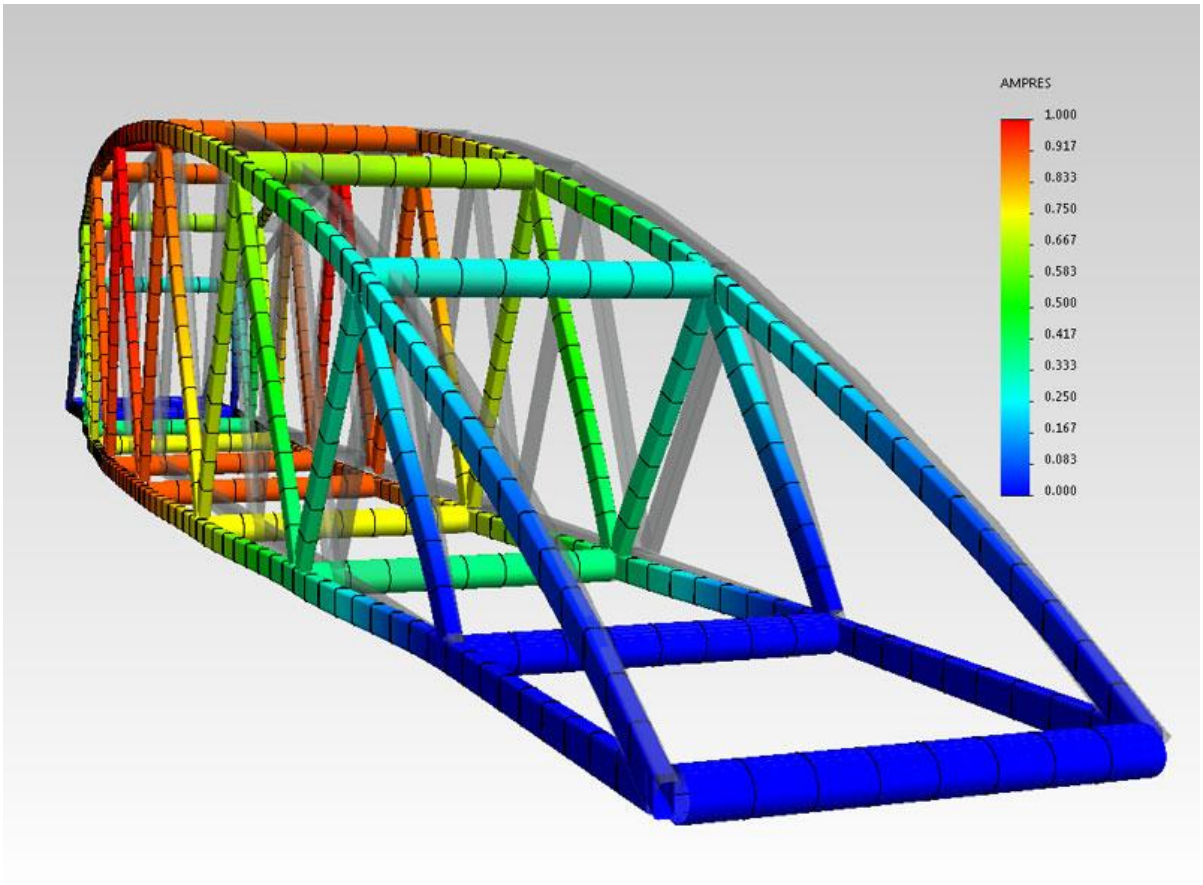


Рис. 1 **[8]**

А завдяки топологічній оптимізації, стало можливим зменшувати кількість матеріалу у деталі, при збереженні всіх характеристик виробу.
Власне, топологічна оптимізація є чудовим рішенням для адитивної промисловості такої як, наприклад, 3D printing.
До того ж багато програмного забезпечення для 3D моделювання містять фунціонал топологічної оптимізації за замовчуванням.
На рисунку 2, подано яскравий приклад топологічної оптимізації.

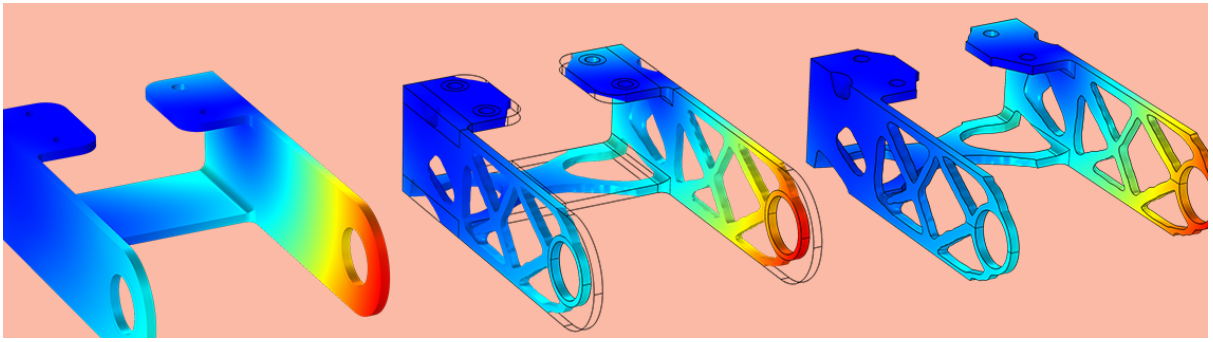


Рис. 2

Мабуть найпопулярнішим прикладом для топологічної оптимізації, може слугувати MBB (Messerschmitt Blohm Bölkow beam problem)

Отже, у нас є ось така металева площина, до якої ми прикладаємо зусилля зверху. **[4]**

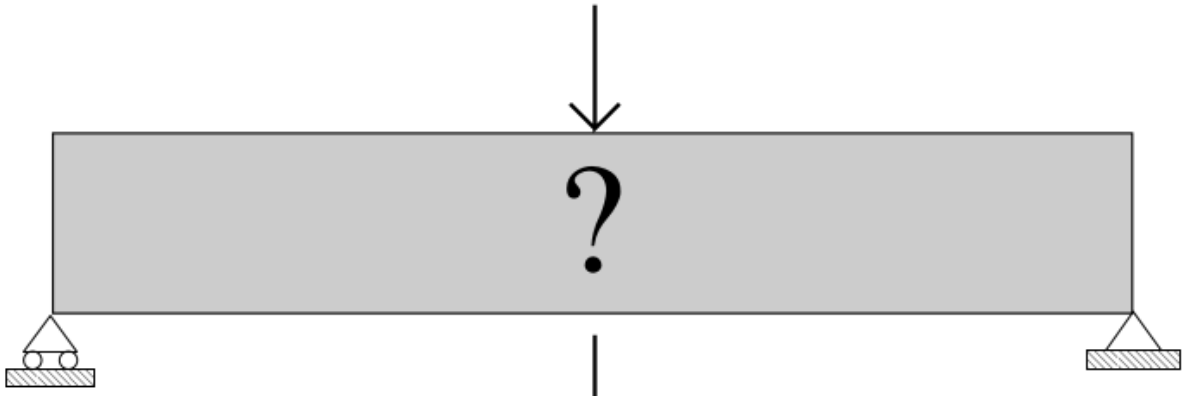


Рис. 3

Ось яким є результат топологічної оптимізації: **[4]**

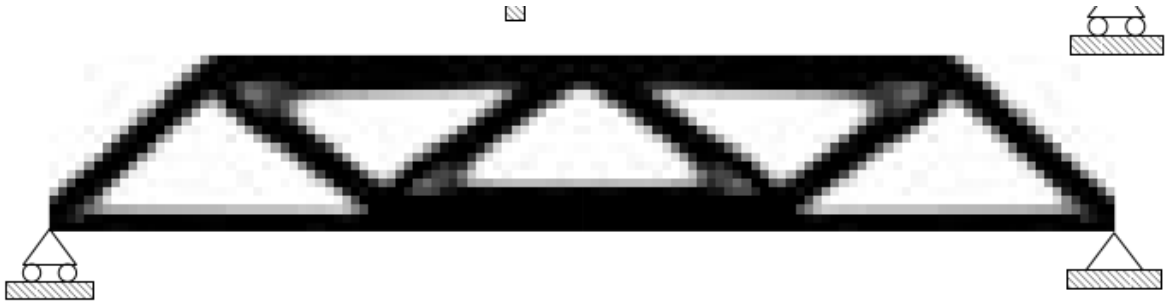


Рис. 4

Зокрема цей об’єкт буде взято як приклад для демонстрації алгоритму у цій дипломній роботі.

Предмет дослідження цієї дипломної роботи полягає у вивченні поведінки фізичного об'єкту та підтвердженню можливості використання методу Ньютона для топологічної оптимізації об'єкту у додатку з порівнянням однопотокового алгоритму з тим самим алгоритмом але таким, що включає у собі паралелізацію окремих моментів виконання алгоритму.

Щоб провести дослідження необхідно:

- Сформулювати та розв’язати задачу теорії пружності на цьому об’єкті математично.
- Змоделювати об'єкт дослідження у редакторі 3D моделей, або ж задати об'єкт програмно.
- Отримати результат моделювання у популярному форматі (*.vtk*, *.msh, тощо).*
- На заданому об'єкті розв'язати задачу теорії пружності програмно
- Запрограмувати та застосувати метод Ньютона.
- Візуалізувати вихідну площину та оцінити результати.
- Запрограмувати паралелізацію окремих елементів коду виконання та/або алгоритму
- Порівняти результати топологічної оптимізації та час виконання програми з паралелізмом та без.

Інструментарій:

1) Для створення 2D моделі - *Gmsh (або задати програмно)*
2) Для розв'язку задачі теорії пружності - програмну бібліотеку *deal.ii*
3) Для візуалізації - програмне забезпечення *Para-view(kitaware)/Visit*

Результати цього дослідження можуть бути використані як у приватних проектах так і в публічних, навіть у промисловості. Це і являє собою практичну цінність.

1. Методи розв’язування

1.1. Метод Ньютона

У цій роботі метод Ньютона застосовується для пошуку напряму. Нижче подано його формулювання як звичайного, так і для оптимізації.

Нехай $f\colon R \to R$ двічі диференційовна і кожна з похідних функції неперервна і не набуває значення нульна проміжку [a,b].

Якщо рівняння $f(x) = 0$ має корінь $x = \alpha$, а функція $\psi(x)$ неперервна в околі $x = \alpha$, то рівняння **[13]**

$$x = x - \psi(x)f(x)$$

Також має корінь $x = \alpha$. Функцію $\psi(x)$ можна підібрати так, що ітераційний процес буде збіжним.
У методі Ньютона функція $\psi(x)$ подана ось таким чином.

$$\psi(x) = \frac{1}{f'(x)}$$

x_{k+1} обчислюється наступним чином:

$$x_{k+1} = x_k - \frac{f(x_k)}{f'(x_k)}$$

За початкове наближення доцільно вибирати той край проміжку [a,b] для якого виконується умова: **[13]**

$$f(x_0)f''(x_0) > 0$$

Швидкість збіжності методу

$$|x_n - \alpha| \le \frac{M_2}{2m_1}|x_{n-1} - \alpha|^2$$

Де,
$m_1 = min_{[a,b]} |f'(x)|$
$M_2 = min_{[a,b]} |f''(x)|$

1.2. Метод Ньютона для оптимізації

Нехай $f\colon R \to R$ двічі диференційовна функція і ми розглядаємо ось таку задачу оптимізації:

$$min_{x \in R} f(x)$$

Метод Ньютона намагається вирішити цю задачу шляхом побудови послідовності $\{x_k\}$ з початкової точки $x_0 \in R$ що збігається до x_* з використанням розкладу функції f в точці x_k в ряд Тейлора другого порядку

$$f(x_k + t) \approx f(x_k) + f'(x_k)t + \frac{1}{2}f''(x_k)t^2$$

x_{k+1} обчислюється наступним чином:

$$x_{k+1} = x_k + t, \text{ де } t = -\frac{f'(x_k)}{f''(x_k)}$$

Отже, кінцеве формулювання методу Ньютона буде ось таким:

$$x_{k+1} = x_k - \frac{f'(x_k)}{f''(x_k)}$$

Збіжність методу Ньютона обчислюється ось таким чином:

$$\|x_{k+1} - x_*\| \leq \frac{1}{2}\|x_k - x_*\|^2, \qquad \forall k \geq 0$$

І є квадратичною.

Де $x_* = argmin\, f(x)$

1.3. Метод множників Лагранжа

Нехай потрібно знайти екстремум функції n змінних за m умов

$$g_j(x_1, x_2, \ldots, x_n) = 0, \; j = 1, 2, 3, \ldots m$$

Розглянемо систему скінченної кількості рівнянь:

$$\{x \in R^n : F_i(x) = 0, \; i, m\}$$

Формулювання функції Лагранжа:

$$L(x_1, x_2, \ldots, x_n) = F(x_1, x_2, \ldots, x_n) - \sum_{i=1}^{m} \lambda_i g_i(x_1, x_2, \ldots, x_n)$$

$\lambda = (\lambda_0, \lambda_1, \ldots, \lambda_m)$ - множники Лагранжа.

$g_1(x), g_2(x), \ldots, g_m(x)$ – умови.

Для того, щоб знайти умовний екстремум необхідно розв'язати систему n+m рівнянь із n+m змінними:

$$\frac{dF(x_1,x_2,....,x_n,\lambda_0,\lambda_1,...,\lambda_m)}{dx_i} = 0,\ \ i = 1,2,3,...n$$

$$\frac{dF(x_1,x_2,....,x_n,\lambda_0,\lambda_1,...,\lambda_m)}{d\lambda_i} = g_j(x_1,x_2,....,x_n),\ \ j = 1,2,3,...m$$

1.4 Метод скінченних елементів

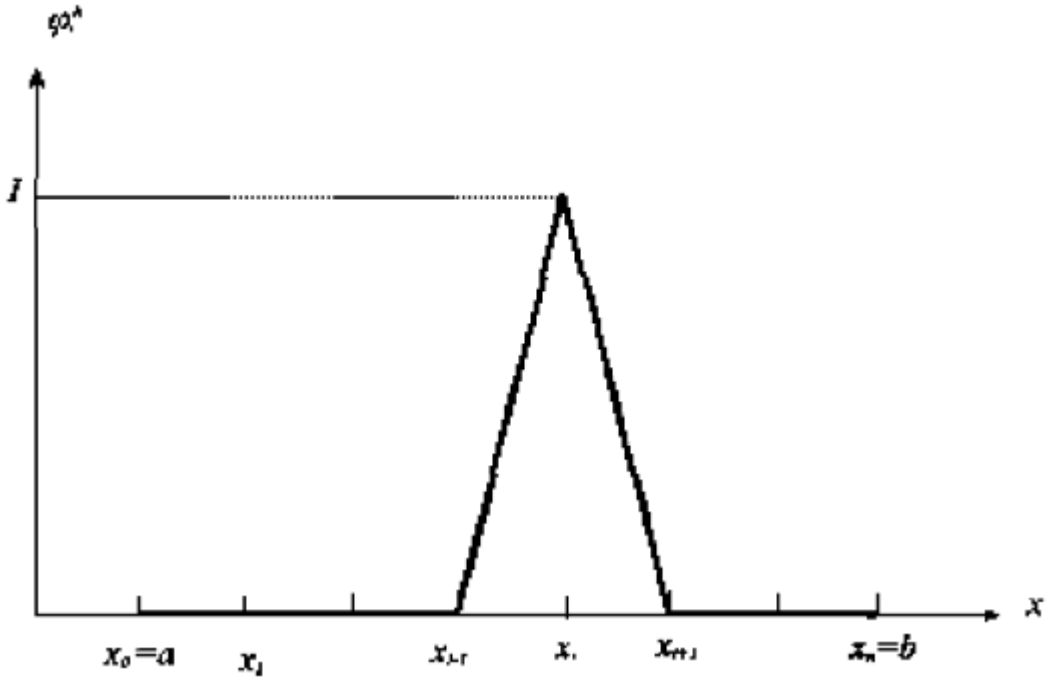
Розділимо відрізок $[a, b]$ на відрізки -- скінченні елементи, точками $x_i,\ i = 0,1,...,n\ \ x_0 = a,\ x_n = b$

Базисна функція $\varphi_i^h(x)$ задається співвідношенням: **[6]**

$$\varphi_i^h = \begin{cases} 0, & x_0 \le x < x_{i-1}; \\ \frac{x-x_{i-1}}{h_i}, & x_{i-1} \le x < x_i; \\ -\frac{x-x_{i+1}}{h_{i+1}}, & x_i \le x < x_{i+1}; \\ 0, & x_{i+1} \le x < x_n, \end{cases} \tag{16}$$

Де $h_i = x_i - x_{i+1}$.

Графік отриманої кусково-аналітичної функції:



Властивості ф-ції $\varphi_i^h(x)$: **[6]**

1. $\varphi_i^h(x) \in C_{[a,b]}$
2. $(\varphi_r^h, \varphi_s^h) = 0$, якщо $|r - s| \geq 2$
3. $\varphi_i^h(x_i) = \delta_{ij}$

З властивості 3. випливає: якщо

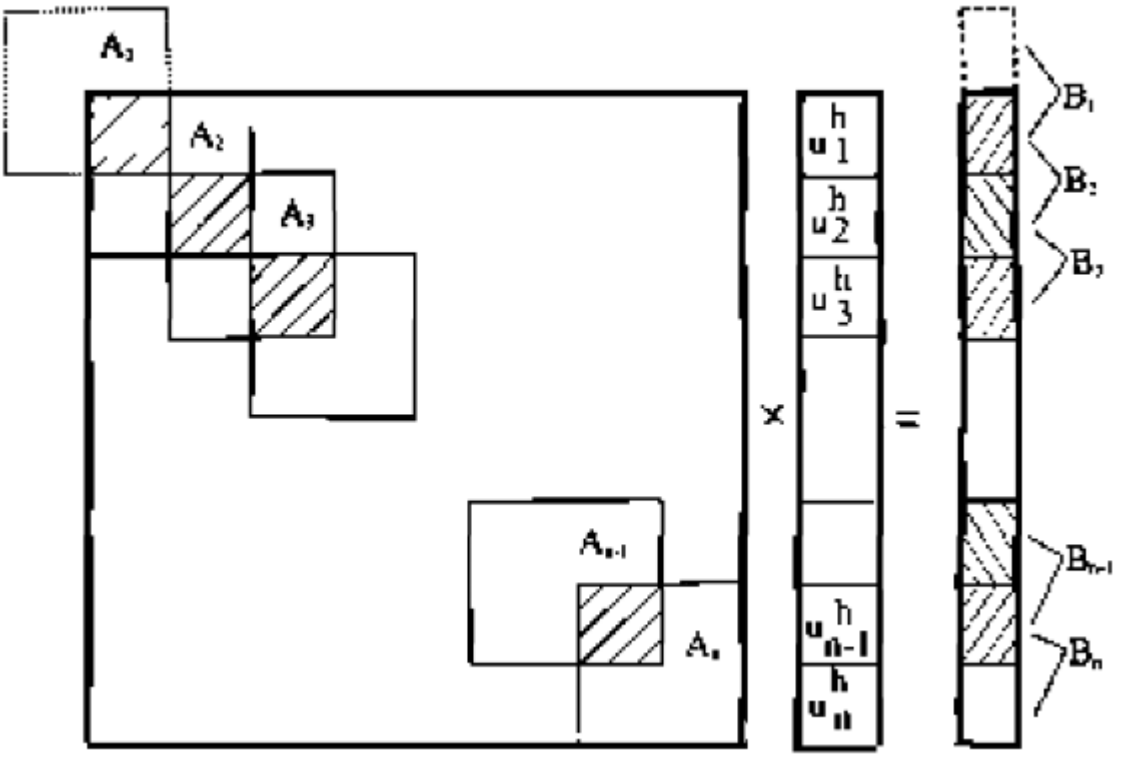
$$u_n = \sum_{i=1}^{n} \alpha_i \varphi_i^h(x) \tag{17}$$

То, $a_i = u_h(x_i)$. Введемо позначення $(u_h(x_i) = u_i^h)$

І тепер на скінченному елементі, u_i^hможна записати як:

$$u_h = u_{i-1}^h \varphi_{i-1}^h(x) + u_i^h \varphi_i^h(x) \tag{18}$$

Отже, власне суть методу полягає в апроксимації розв*язку,
І для цього нам необхідно звести диференціальне р-ня до системи лінійних рівнянь, яку ми можемо розв’язати будь-яким із відомих чисельних методів.
Схема формування с-ми лінійних рівнянь: **[6]**



Загальна процедура методу скінченних елементів (алгоритм):

Етап ініціалізації (препроцесингу): **[7]**

1. Задати площину
2. Тип елементів
3. Задати властивості матеріалів
4. Задати геометричні властивості елементів
5. Тріангулювати площину, та створити сітку з скінченних елементів
6. Задати фізичні обмеження
7. Задати напруги

Етап розв’язку задачі:

1. Програмно знайти розв’язок.

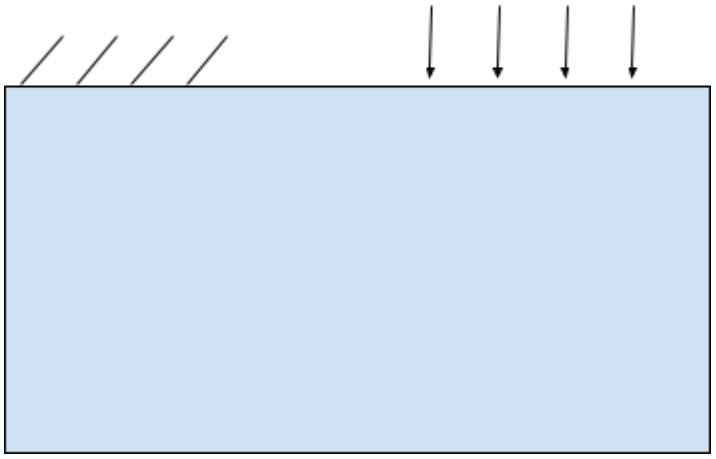
Етап отримання розв’язку (постпроцесинг):

1. Перевірити рівняння рівноваги
2. Відобразити розв’язок графічно.

2. Топологічна оптимізація

2.1. Базові означення теорії пружності

Нехай область Ω-тіло



σ - тензор напруження,
F - прикладені сили,
λ та ***μ***- коефіцієнти Ламе матеріалу в області Ω,
I - одиничний тензор,
tr - оператор сліду на тензорі,
ε - симетричний тензор деформації,
u - зміщення.

Напруження визначається як: **[3]**

$$\sigma = \lambda(\nabla \cdot u)I + \mu(\nabla u + (\nabla u)^T) \quad (4)$$

Також напруження можна визначити ось таким чином:

$$\sigma = C : \varepsilon(u) \quad (5)$$

C - тензор деформації,

Формулювання задачі оптимізації

$$min \| \sigma(u) \|_\infty$$

$$\text{за умови } |E| \leq V_{max}$$

$$\nabla \cdot \sigma(\rho) + F = 0 \text{ on } \Omega,$$

Де

$\|\sigma(u)\|_{\infty}$- можна переформулювати як $\int_{\Omega} \frac{1}{2}\, \sigma\colon \varepsilon\, dV$

Нам потрібно мінімізувати енергію деформації. **[5]**

$$min \int_{\Omega} \frac{1}{2}\, \sigma\colon \varepsilon\, dV$$

$$\text{за умови } || E || \leq V_{max}$$

$$\nabla\cdot\sigma + F = 0,$$

Значення цільової функції обчислюється за допомогою методу скінченних елементів де розв'язок є зміщенням.

2.2 Метод SIMP (Solid Isotropic Material with Penalization)

Метод SIMP дозволяє працювати з значеннями густини з проміжку $[0, 1]$. З цього випливає, що ці значення відповідають такій нерівності:

$$0 < \rho_{min} \leq \rho \leq 1$$

p_{min}- це константа яку оберемо як близько 10^{-3}, щоб уникнути

ймовірності появи нескінченної енергії напруження та максимально зберегти точність результатів.

Безпосереднім застосуванням ефекту цієї “густини” на реальному об’єкті буде просте множення тензору жорсткості C_0 на дану густину $C = \rho C_0$

Для отримання р-зку, що застосовний в реальному житті, тобто $\rho \in \{0, 1\}$, нам необхідно додати штраф.

Зробимо це за рахунок підняття густини до степеня р та множення на тензор жорсткості. **[5]**

$$C = \rho^p C_0 .$$

Завдяки описаному вище методу SIMP ми переформулюємо задачу оптимізації таким чином:

$$min \int_{\Omega} \frac{1}{2}\, \sigma(\rho)\colon \varepsilon(\rho) d\Omega$$

$$\text{за умови } \int_{\Omega} p(x)\, d\Omega = V_{max}$$

$$0 < \rho_{min} \leq \rho(x) \leq 1 ,$$

$$\nabla \cdot \sigma(\rho) + F = 0 \; on \; \Omega$$

За допомогою р-ня пружності, ми можемо знайти σ та ε.

2.3 Рівняння пружності

Рівняння пружності

$$\nabla \cdot \sigma(\rho) + F = 0 \; on \; \Omega$$

де

$$\sigma = C : \varepsilon = \rho^p C_0 : \varepsilon(u) = \rho^p C_0 : [\tfrac{1}{2}(\nabla u + (\nabla u)^T)]$$

u - наш розв'язок.

Також ми можемо записати тензор напружень з використанням коефіцієнтів Ламе:

$$\sigma = \rho^p(\lambda * tr(\varepsilon) I + 2\mu\varepsilon)$$

$$\sigma_{i,j} = \rho^p(\lambda \varepsilon_{k,k} \delta_{i,j} + 2\mu\varepsilon_{i,j})$$

Проінтегрувавши частинами цільову функцію частинами, отримаємо:

$$\int_\Omega \sigma(\rho) : (\nabla u + (\nabla u)^T) d\Omega + \int_\Omega (\nabla \cdot \sigma(\rho)) \cdot u d\Omega = \int_{\partial\Omega} t \cdot u d\partial\Omega,$$

в яке потім можна підставити лінійне рівняння пружності:

$$\int_\Omega \sigma(\rho) : (\nabla u + (\nabla u)^T) d\Omega = \int_\Omega F \cdot u \, d\Omega + t \cdot u d\partial\Omega,$$

Оскільки ми не розглядаємо випадок об'ємних сил, то все зводиться до такого вигляду:

$$\int_\Omega \sigma(\rho) : (\nabla u + (\nabla u)^T) d\Omega = \int_{\partial\Omega} t \cdot u d\partial\Omega,$$

Фільтр густини працює за рахунок введення нефільтрованої густини ϱ. Звичайна густина буде представлена як:

$$\rho = H(\varrho)$$

H це оператор, такий, що $\rho(x)$ - є середнім значенням ϱ в області навколо x

2.4 Повне формулювання задачі топологічної оптимізації

Задача мінімізації тепер, виглядає ось так:

$$min_{\rho,\varrho,u} \int_{\partial\Omega} u \cdot t d\partial\Omega$$

$$\text{за умови } \rho = H(\varrho)$$

$$\int_{\Omega} \rho^{p} \left(\frac{\mu}{2} (\varepsilon(v) : \varepsilon(u))\right) + \lambda(\nabla \cdot u \nabla \cdot v) d\Omega = \int_{\partial\Omega} v \cdot t \, d\partial\Omega$$

$$\int_{\Omega} \rho \, d\Omega = V$$

$$0 \leq \varrho \leq 1$$

Обмеження нерівності вирішуються шляхом: **[5]**

- введення люзової змінної,
- використання логарифмічних бар'єрів для забезпечення можливості використання методу внутрішньої точки.

α - параметр штрафу

Наступні люзові змінні будуть такими:

s_1 - люзова змінна, що відповідає нижній межі

s_2 - люзова змінна, що відповідає верхній межі.

Отримуємо нове переформулювання задачі:

$$\min_{\rho,\varrho,\mathbf{u},s_1,s_2} \int_{\partial\Omega} \mathbf{u} \cdot \mathbf{t} d\partial\Omega - \alpha \int_{\Omega} (\log(s_1) + \log(s_2)) \, d\Omega$$

$$\text{за умови } \rho = H(\varrho)$$

$$\int_{\Omega} \rho^{p} \left(\frac{\mu}{2} (\varepsilon(v) : \varepsilon(u))\right) + \lambda(\nabla \cdot u \nabla \cdot v) d\Omega = \int_{\partial\Omega} v \cdot t \, d\partial\Omega$$

$$\int_{\Omega} \rho \, d\Omega = V$$

$$\varrho = s1$$

$$1 - \varrho = s2$$

Із таким формулюванням ми можемо застосувати звичний нам підхід для розв'язування задач пошуку екстремуму з обмеженнями, отже побудувати Лагранжіан в якому ми об'єднуємо цільову функцію та обмеження, шляхом множення обмежень на множники Лагранжа.

y_1 - множник Лагранжа, що відповідає обмеженню пружності

y_2 - множник Лагранжа, що відповідає обмеженню фільтра згортки(convolution).

z_1 - множник Лагранжа, що відповідає нижній люзовій змінній

z_2 - множник Лагранжа, що відповідає верхній люзовій змінній

Отже, Лагранжіан буде мати наступний вигляд

$$\int_{\partial\Omega} u{\cdot}t\, d\partial\Omega - \alpha\int_{\Omega} log(s1) + log(s2))d\Omega -$$

$$-\int_{\Omega}\rho^{p}(\tfrac{\mu}{2}(\varepsilon(y_1):\varepsilon(u))) + \lambda(\nabla{\cdot}u\nabla{\cdot}y_1))d\Omega - \int_{\partial\Omega} y_1{\cdot}td\partial\Omega -$$

$$-\int_{\Omega} y_2(\rho - H(\varrho))\, d\Omega - \int_{\Omega} z_1(\varrho - s_1)d\Omega - \int_{\Omega} z_2(1 - s_2 - \varrho)\, d\Omega$$

Розв’язок задачі повинен задовольняти умовам Каруша-Куна-Таккера.

Сформулюємо ККТ умови нижче.

$d\{\cdot\}$ - тестова функція, яка природно об’єднана з варіаційними похідними Лагранжіана відносно функції $\{\cdot\}$.

Г - ділянка де прикладена сила, і застосовуються граничні умови Неймана.

Стаціонарності: **[5]**

$$\int_{\Omega} -d_\rho y_2 + p\rho^{p-1}d_\rho\,[\lambda(\nabla\cdot\mathbf{y}_1)(\nabla\cdot\mathbf{u}) + \mu\boldsymbol{\varepsilon}(\mathbf{u}) : \boldsymbol{\varepsilon}(\mathbf{y}_1)]\, d\Omega = 0 \quad \forall d_\rho$$

$$\int_{\Gamma} \mathbf{d_u}\cdot\mathbf{t}d\partial\Omega + \int_{\Omega} p\rho^{p}\,[\lambda(\nabla\cdot\mathbf{d_u})(\nabla\cdot\mathbf{y}_1) + \mu\boldsymbol{\varepsilon}(\mathbf{d_u}) : \boldsymbol{\varepsilon}(\mathbf{y}_1)]\, d\Omega = 0 \quad \forall\mathbf{d_u}$$

$$\int_{\Omega} -d_\varrho z_1 + d_\varrho z_2 + H(d_\varrho)y_2 d\Omega = 0 \quad \forall d_\varrho$$

Допустимість:

$$\int_\Omega \rho^p \lambda(\nabla \cdot \mathbf{d}_{\mathbf{y}_1})(\nabla \cdot \mathbf{u}) + \rho^p \mu \boldsymbol{\varepsilon}(\mathbf{d}_{\mathbf{y}_1}) : \boldsymbol{\varepsilon}(\mathbf{u}) d\Omega - \int_\Gamma \mathbf{d}_{\mathbf{y}_1} \cdot \mathbf{t} d\partial\Omega = 0 \quad \forall \mathbf{d}_{\mathbf{y}_1}$$

$$\int_\Omega d_{z_1}(\varrho - s_1) d\Omega = 0 \quad \forall d_{z_1}$$

$$\int_\Omega d_{z_z}(1 - \varrho - s_2) d\Omega = 0 \quad \forall d_{z_2}$$

$$\int_\Omega d_{y_2}(\rho - H(\varrho)) d\Omega = 0 \quad \forall d_{y_2}$$

Спряженість

$$\int_\Omega d_{s_1}(s_1 z_1 - \alpha) d\Omega = 0 \quad \forall d_{s_1}, \quad \alpha \to 0$$

$$\int_\Omega d_{s_2}(s_2 z_2 - \alpha) d\Omega = 0 \quad \forall d_{s_2}, \quad \alpha \to 0$$

Двоїста допустимість: **[5]**

$$s_{1,i}, s_{2,i}, z_{1,i}, z_{2,i} \geq 0 \quad \forall i$$

2.5 Процедура розв*язку

Метод Ньютона, застосований до наведених вище рівнянь, призводить до системи рівнянь, наведеної нижче.
Варіаційні похідні щодо змінної $\{\cdot\}$ беруться в напрямку $c\{\cdot\}$.

Стаціонарність:
Ці рівняння перевіряють, чи ми знаходимося в критичній точці цільової функції, коли вона обмежена.
Рівняння 1

$$\int_\Omega - d_p c_{y_2} + p(p-1)\rho^{p-2} d_p c_p [\lambda \nabla \cdot y_1 \nabla \cdot u + \mu \varepsilon(u)\varepsilon(y_1)] +$$
$$+ p\rho^{p-1} d_p [\lambda \nabla \cdot c_{y_1} \nabla \cdot u + \mu \varepsilon(u)\varepsilon(c_{y_1})] + p\rho^{p-1} d_p [\lambda \nabla \cdot y_1 \nabla \cdot c_u + \mu \varepsilon(c_u)\varepsilon(y_1)] d\Omega =$$

$$= \int_{\Omega} - d\rho z_1 + d\rho z_1 - d\rho y_2 + p\rho^{p-1} d_p [\lambda \nabla \cdot y_1 \nabla \cdot u + \mu \varepsilon(u) \varepsilon(y_1)] d\Omega$$

Рівняння 2 **[5]**

$$\int_{\Omega} p\rho^{p-1} c_p [\lambda \nabla \cdot y_1 \nabla \cdot d_u + \mu \varepsilon(du) \varepsilon(y)] + \rho^p [\lambda \nabla \cdot c_{y_1} \nabla \cdot d_u + \mu \varepsilon(d_u) \varepsilon(c_{y_1})] \, d\Omega$$

Рівняння 3

$$\int_{\Omega} - d_\varrho c_{z_1} + d_\varrho c_{z_2} + H(d_\varrho) c_{y_2} d\Omega = - \int_{\Omega} - d_\varrho z_1 + d_\varrho z_2 + H(d_\varrho) y_2 d\Omega$$

Допустимість: Ці рівняння перевіряють виконання обмежень рівності.
Рівняння 4

$$\int_{\Omega} p\rho^{p-1} c_p [\lambda \nabla \cdot d_{y_1} \nabla \cdot u + \mu \varepsilon(u) \varepsilon(d_{y_1})] + \rho^p [\lambda \nabla \cdot d_{y_1} \nabla \cdot c_u + \mu \varepsilon(c_u) \varepsilon(d_{y_1})] d\Omega =$$

$$= - \int_{\Omega} \rho^p [\lambda \nabla \cdot d_{y_1} \nabla \cdot u + \mu \varepsilon(u) \varepsilon(d_{y_1})] + \int_{\Gamma} d_{y_1} \cdot td \, \partial\Omega$$

Рівняння 5

$$- \int_{\Omega} d_{z_1} (c_\varrho - c_{s_1}) \, d\Omega = - \int_{\Omega} d_{z_1} (\varrho - s_1) \, d\Omega$$

Рівняння 6

$$- \int_{\Omega} d_{z_2} (- c_\varrho - c_{s_2}) \, d\Omega = - \int_{\Omega} d_{z_2} (1 - \varrho - s_2) \, d\Omega$$

Рівняння 7

$$- \int_{\Omega} d_{y_2} (c_p - H(c_\varrho) \, d\Omega = \int_{\Omega} d_{y_2} (\rho - H(\varrho)) \, d\Omega$$

Доповнююча нежорсткість: Рівняння перевіряють подолання бар'єру. В кінцевому розв'язку нам потрібно $s^T z = 0$ **[5]**
Рівняння 8

$$\int_{\Omega} d_{s_1} (c_{s_1} z_1 / s_1 + c_{z_1}) d\Omega = - \int_{\Omega} d_{s_1} (z_1 - \alpha / s_1) d\Omega, \quad \alpha \to 0$$

Рівняння 9

$$\int_{\Omega} d_{s_2}(c_{s_2} z_2/s_2 + c_{z_2})d\Omega = -\int_{\Omega} d_{s_2}(z_2 - \alpha/s_2)d\Omega, \quad \alpha \to 0$$

Двоїста допустимість:

$$s, z \geq 0$$

2.6 Алгоритм оптимізації (Нелінійний алгоритм)

Для проведення оптимізації, нам необхідно знайти напрям $c\{\cdot\}$, та розуміти, наскільки нам потрібно просунутися в заданому напрямі. Такий спосіб часто називають «лінійним пошуком» і він зводиться до питання про те, як вибрати довжину кроку $\alpha_k \in (0, 1]$, для такого ітеративного процесу $x_{k+1} = x_k + \alpha_k x_k$. **[11]**

В кінцевому підсумку нам необхідно буде покласти $\alpha_k = 1$, щоб реалізувати квадратичну збіжність методу Ньютона;
Однак, на перших ітераціях такий довгий крок насправді може погіршити ситуацію, тобто, метод може вивести нас на точку, у якій результат цільової функції погіршився або в якій обмеження задовольняються в меншій мірі ніж у точці x_k.

Для вирішення проблеми описаної вище проблеми оптимально застосувати алгоритм watchdog.

В реалізації цього алгоритму застосовано вектор x для представлення всіх первинних змінних – відфільтрованої та нефільтрованої густин, люзових змінних та зміщення – і використовуватимемо вектор y для представлення всіх спряжених векторів. (Інкрементальним) розв'язком нелінійної системи рівнянь, зазначеної вище, будемо називати Δx і Δy замість $c\{\cdot\}$. $\phi(x, y)$ - Функція оцінки(merit function)(подана і пояснена нижче)

Алгоритм застосовується із заданим параметром бар'єру і працює наступним чином: **[11]**
По-перше, поточна ітерація зберігається як стан, а оцінка(merit) стану watchdog записується. Потім робиться максимально можливий крок методу Ньютона. Якщо оцінка значно зменшилася порівняно із першим кроком, цей новий крок приймається. Якщо ні, робиться ще один максимально

можливий крок Ньютона, і оцінка знову порівнюється з оцінкою результату отриманого за допомогою watchdog. Якщо після деякої кількості (зазвичай від 5 до 8) кроків методу Ньютона оцінка не знизилася належним чином, алгоритм виконує масштабований крок Ньютона або зі збереженого стану watchdog, або з останньої ітерації, яка гарантує достатнє зниження оцінки, і цей крок приймається. Після прийняття кроку вимірюється норма KKT(Karush-Kunn-Tacker), і якщо вона досить мала, значення бар'єру зменшується. Якщо значення бар'єру недостатньо мале, то останнім прийнятим кроком вважається новий крок watchdog, і процес повторюється.
Обчислення норми KKT:

$$|x|_{l1} = \sum_{i=1}^{n} |x_i|$$

«Максимально здійсненний крок» є масштабуванням кроку Ньютона як у первинній, так і в двойственній змінних, заданих таким чином:

$$\beta^y = min\{1, max\ \beta \text{ таке, що } (z_{k+i} + \beta_{k+i}{}^{z} + \Delta z_{k+i})_j \geq \zeta z_{k+i,j}\ \forall j\}$$

$$\beta^x = min\{1, max\ \beta \text{ таке, що } (s_{k+i} + \beta_{k+i}{}^{s} + \Delta s_{k+i})_j \geq \zeta s_{k+i,j}\ \forall j\}$$

ζ — це «дріб до межі»(fraction to boundary), який дозволений на будь-якому кроці. Оскільки похідні стають погано обумовленими поблизу межі, цей метод замінює область довіри і необхідний для забезпечення гарних наближень у майбутньому.
Обираємо $\zeta = max\{0,8, 1 - \alpha\}$, що дозволяє рухатися ближче до межі, коли бар'єр стає меншим.
Окремо, нам потрібно мати справу з логарифмічним бар'єром, який ми використовували, щоб застосувати обмеження позитивності на люзові змінні(slack variable) s1, s2: у формулюванні остаточної оптимізаційної задачі, яку ми вирішуємо, було додано вираз:

$$-\alpha \int_{\Omega} (log(s_1) + log(s_2)) d\Omega$$

У монотонному методі, реалізованому тут, параметр бар'єру оновлюється щоразу, коли досягається певний рівень збіжності для поточного параметра

бар’єру. Ми використовуємо норму l_1 умов KKT для перевірки збіжності при кожному новому розмірі бар’єру.
Умова зупинки записується як:

$$\|KKT\|_{l_1} < c \cdot \alpha, \text{ де}$$

c – константа для будь-якого розміру бар’єру,
α – параметр бар’єру.

Бар’єр зменшується лінійно при більших значеннях і надлінійно при менших значеннях. На практиці це виглядає наступним чином:

$$\alpha_{k+1} = min\{\alpha^{1.2}_k, 0.6\alpha_k\}$$

2.7 Функція оцінки (Merit function)

Задача нелінійної оптимізації з обмеженнями:

$$min_{x \in R^n} f(x) \text{ за умов } \quad c_i(x) = 0 \; i \in \varpi$$
$$c_i(x) \geq 0 \; i \in L \qquad \textbf{[11]}$$

ϖ - скінченна індексна множина обмежень нерівності
L - скінченна індексна множина обмежень рівності.

У задачах оптимізації без обмежень на запитання, чи є вигідним крок у вибраному напрямі можна просто перевірити за допомогою цільової функції, яку ми намагаємося мінімізувати і застосувати такі умови, як умови Вульфа та Гольдштейна. **[5]**

У задачах оптимізації з обмеженнями питання полягає в тому, як збалансувати зменшення цільової функції з можливим збільшенням порушення обмежень: запропонований крок може зменшити цільову функцію, але бути далі від набору точок, які задовольняють обмеження, або навпаки. Це зазвичай вирішується за допомогою функції оцінки, яка поєднує два критерії.
Алгоритм, описаний вище, використовує «функцію оцінки». Функції оцінки використовуються, щоб визначити, чи є вигідним крок від x_k до запропонованої точки x_{k+1} .

Функція оцінки φ(x; μ) називається строгою(exact), якщо існує таке $\mu^* > 0$ число, таке що для будь якого $\mu > \mu^*$ будь який локальний розв'язок нелінійної задачі є локальним мінімумом функції оцінки φ(x; μ).

Тут ми використовуємо ось таку “строгу”(exact) **[11]** функцію оцінки l_1:

$$\phi(x,y) = \int_{\partial\Omega} u\cdot td\partial\Omega - \alpha\int_{\Omega}(log(s_1) + log(s_2)) + p\sum_i \left|\int_{\Omega} y_{2,i}(H(\varrho) - \rho)d\Omega\right| +$$

$$+ p\sum_i \left|\int_{\partial\Omega} y_{1,i}\cdot td\partial\Omega - \int_{\Omega} p^p\,[\lambda\nabla\cdot u\nabla\cdot y_{1,i} + \mu\varepsilon u\varepsilon y_{1,i}]d\Omega\right| +$$

$$+ p\sum_i \left|\int_{\Omega} z_1, i(s_1 - \varrho)d\Omega\right| + p\sum_i |z_{2,i}(1 - \varrho - s2)d\Omega|$$

Де,
p - параметр штрафу

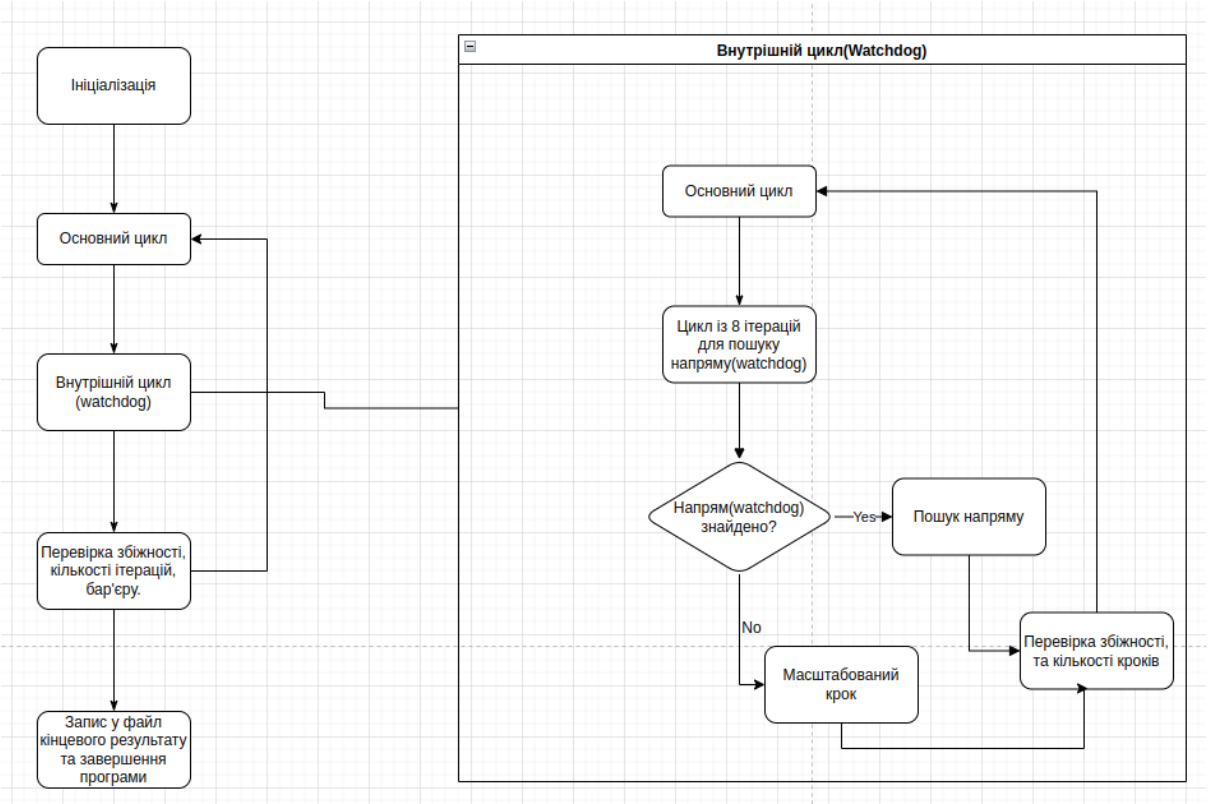
Параметр штрафу оновлюється таким чином:

$$p > \frac{\frac{1}{2}x^T\cdot H\cdot x - x^T\cdot\nabla f}{\left\|c_i\right\|_{l\infty}},\ i\in E,$$

Де
H - це Гессіан цільової функції,
X - це вектор змінних прямої задачі
F - це цільова функція,
c_i - це похибка з поточними обмеженнями рівності,
p - параметр штрафу

3. Програмна реалізація та її паралелізація

3.1. Загальна блок-схема програми:



3.2. Паралелізація алгоритму

Забезпечити паралелізацію алгоритму можна завдяки:

1) Паралелізації алгоритму в цілому
2) Паралелізації окремих елементів алгоритму.

В даній роботі розглядається тільки паралелізація окремих елементів(операцій) алгоритму.

У нашому випадку, складність полягає у тому, що дуже багато процедур виконуються синхронно без можливості їх адекватної паралелізації, хіба це було б можливо при паралелізації функціоналу, що імплементований у програмній бібліотеці, але на жаль така можливість відсутня.

Але, все ж можна розпаралелити, зокрема, операції перевірки обчислень.

Також варто додати, що було розглянуто розпаралелювання з оглядкою на потоки, а не процесори. Тому в такому разі на нас падає обмеження у вигляді використання розумної кількості потоків, аби уникнути

сповільнення програми ще більше, так як в реальності на звичайному ПК процесорний час використовує операційна система та його можуть використовувати інші програми, також варто не забувати, що сам факт переключення операційною системою між потоками теж може забагато використовувати процесорний час, така проблема виникає, якщо використовувати не адекватну кількість потоків, наприклад 100 або більше.
У оригінальному алгоритмі, варто поглянути на ось такі процедури:

1) Пошук напряму у watchdog алгоритмі.
2) Перевірка параметрів перед обчисленням масштабованого кроку.
3) Обчислення масштабованого кроку.

У всіх цих елементів алгоритму є спільна особливість, це багатократне обчислення функції оцінки. Обчислення функції оцінки здебільшого є незалежними операціями, а сама функція оцінки досить є важкою у обчисленні, тому є сенс розпаралелювання не самої функції оцінки (так як це трохи важко зробити через специфіку реалізації) а обчислень, що цю функцію використовують.
У нас є два варіанти як це зробити:

1. Запустити окремим потоком кожне обчислення функції оцінки де є незалежне обчислення, тобто процедури 1) 2) 3).
2. Розглянути паралелізацію кожного алгоритму окремо, і у випадку якщо є покращення у часі виконання хоч якоїсь із частин алгоритму, розглянути комбінацію розпаралелювання цих частин алгоритму.

Перший варіант здається привабливим тільки на перший погляд, адже у нас виникає проблема занадто великої кількості потоків описана вище.
Тому, розглядаємо другий варіант.
Алгоритм 1 - розпаралелювання процедури 1)
Алгоритм 2 - розпаралелювання процедури 2)
Алгоритм 3 - розпаралелювання процедури 3)

Лінійки, що позначені як ||| є під спробую розпаралелювання.

Псевдокод для процедури 1

```
for (k = 0; k < max_tries_to_search_with_watchdog; ++k) {
  update_step = find_max_step();
  if (k == 0) {
    first_step = update_step;
```

```
    merit_derivative =
|||             ((calculate_merit(watchdog_state +
                                  .0001 * first_step) -
|||              calculate_merit(watchdog_state)) / .0001)
    target_merit = calculate_merit(watchdog_state) +
                descent_requirement * merit_derivative;
   }

   nonlinear_solution += update_step;
   current_merit = calculate_merit(nonlinear_solution);

   if (current_merit < target_merit) {
        watchdog_step_found = true;      //Знайшли підходящий крок.
      break;
   }
}
```

Псевдокод для процедури 2

```
if (watchdog_step_found == false) {
   ++iteration_number;
   update_step = find_max_step();
   stretch_state = compute_scaled_step(nonlinear_solution,....);

||| if ((calculate_merit(nonlinear_solution) <
|||     calculate_merit(watchdog_state)) ||
        (calculate_merit(stretch_state) < target_merit)) {
       //Робимо масштабований крок з кінця watchdog
       nonlinear_solution = stretch_state;
 } else {
       //Робимо масштабований крок від початку watchdog
|||       if (calculate_merit(stretch_state) >
|||           calculate_merit(watchdog_state)) {
          nonlinear_solution =
                          compute_scaled_step(watchdog_state,
                                              First_step, ...);
   } else {
     ++iteration_number
     nonlinear_solution = stretch_state
     stretch_step = find_max_step()
     nonlinear_solution = compute_scaled_step(nonlinear_solution,
                                                 stretch_step.....)
    }
  }
}
```

Псевдокод для процедури 3

```
    merit_derivative =
|||     (calculate_merit(state + 1e-4 * max_step) -
|||      calculate_merit(state)) / 1e-4;

    for (unsigned int k = 0; k < max_linesearch_iterations; ++k) {
|||       if (calculate_merit(state + step_size * max_step) <
|||           calculate_merit(state) +
              step_size * descent_requirement * merit_derivative)
          break;
        else
          step_size = step_size / 2;
    }
```

Результати виконання програм:

Для спрощеного прочитання та сприйняття, винесемо таблицю вище ніж результати топологічної оптимізації.

Час виконання – усереднене значення із 10-ти запусків.

Алгоритм	Час виконання (у секундах)	Різниця у числових результатах
Без паралелізації	36	відсутня
Алгоритм 1 (2 потоки)	38	відсутня
Алгоритм 2 (2 потоки)	35.5	відсутня
Алгоритм 2 (4 потоки)	35.72	відсутня
Алгоритм 3 (2 потоки)	35.35	відсутня
Алгоритм 3 (4 потоки)	36	відсутня

Так як різниця у числових результатах відсутня, зображення не будуть дублюватися.

4. Чисельна реалізація

4.1. Реалізована задача

Розміри об’єкту :
Ширина: 1
Довжина: 6
Коефіцієнт Ламе: 1.0
Force: 1, а з напрямом -1;

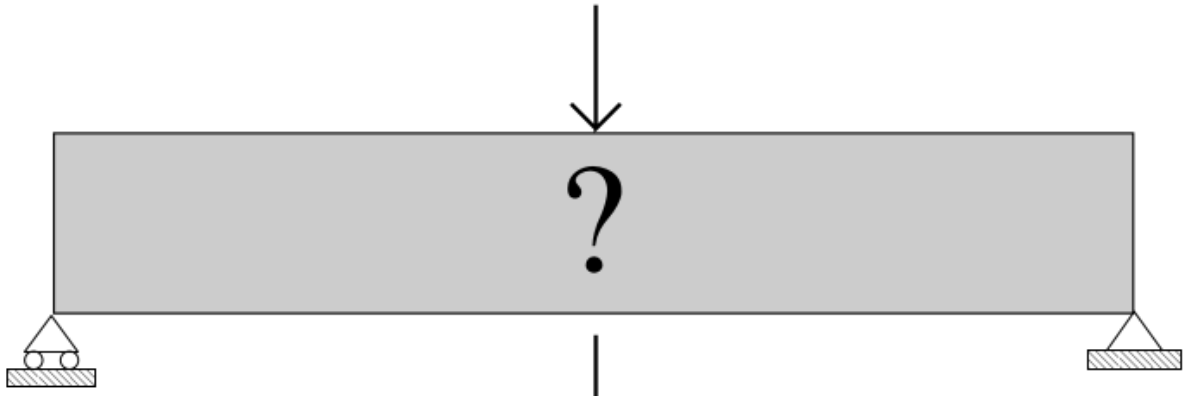


Рис. 4

Вектори напружень:

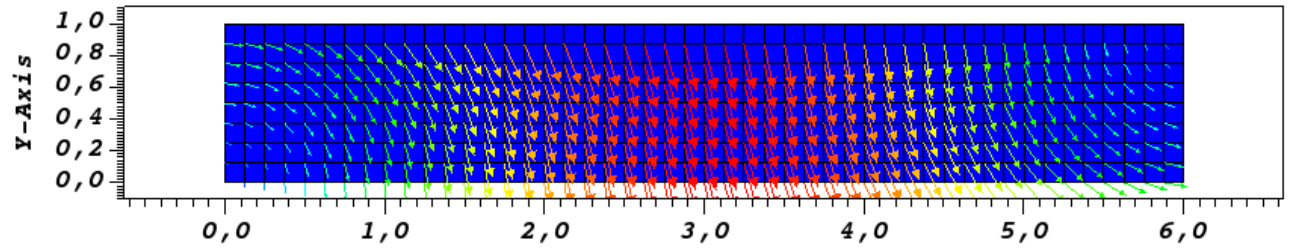


Рис.5

Графік зміни тестової норми

Рис. 6

Графік зміни цільової норми

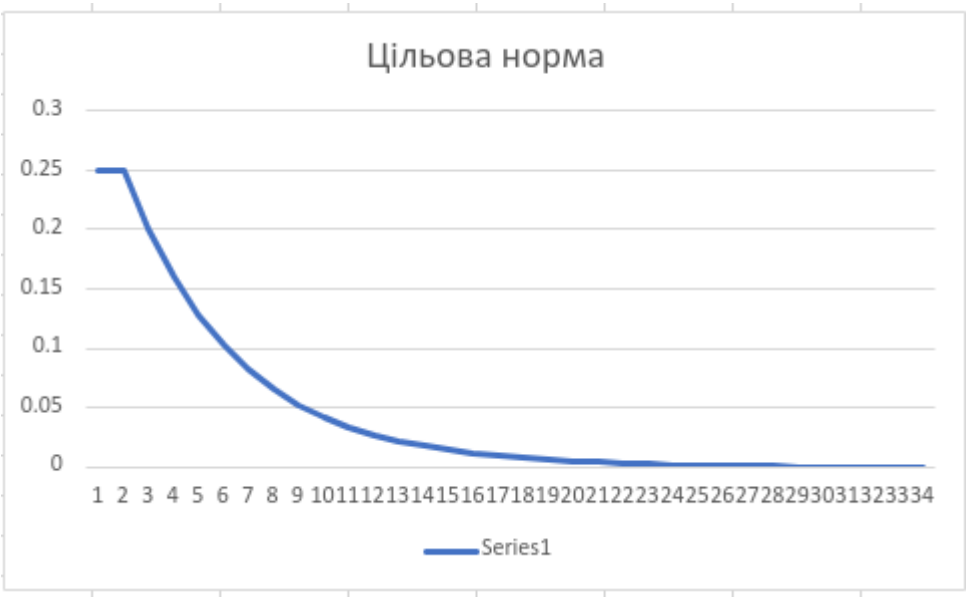


Рис. 7

Нижче подана “еволюція” проміжних результатів до кінцевого. При виконанні програми в мене вийшло близько 120 ітерацій, тож наведено тільки ті які є найбільш значущими для демонстрації.

Графічне зображення оптимізації:

Рис. 8

Рис. 9

Рис. 10

Рис. 11

Рис. 12

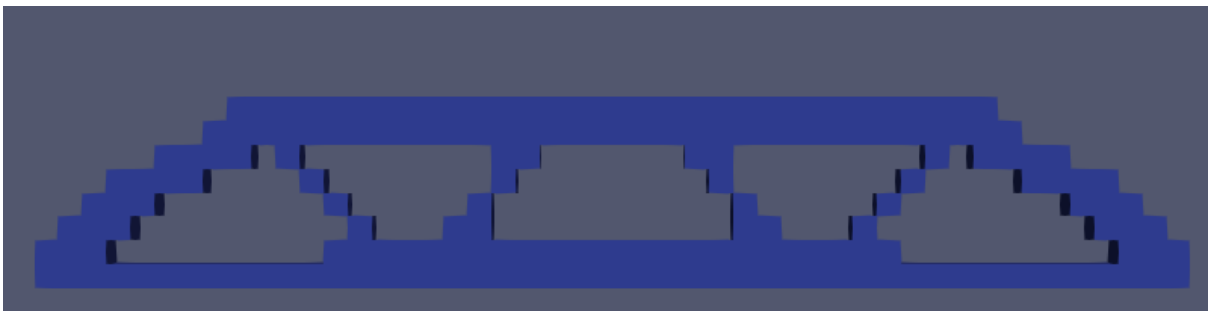


Рис. 13

Ось яким є кінцевий результат топологічної оптимізації:

Рис. 14

Зображення проміжних густин:

Ітерація 05

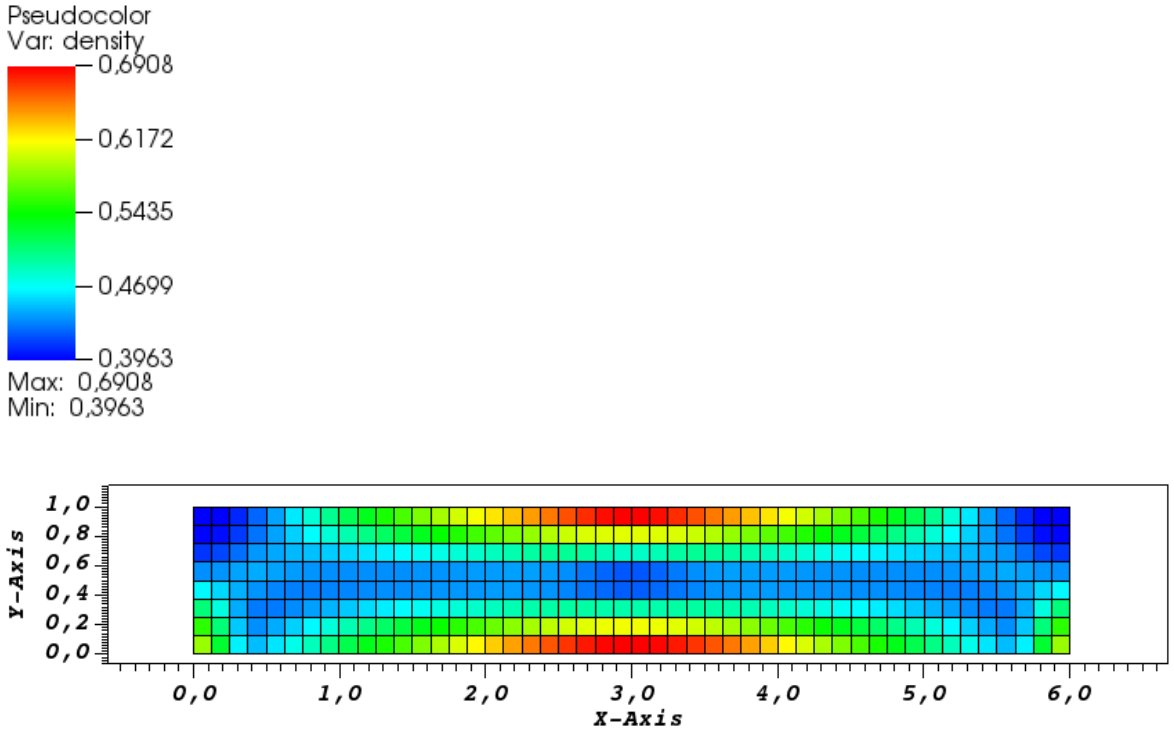


Рис.15

Ітерація 10

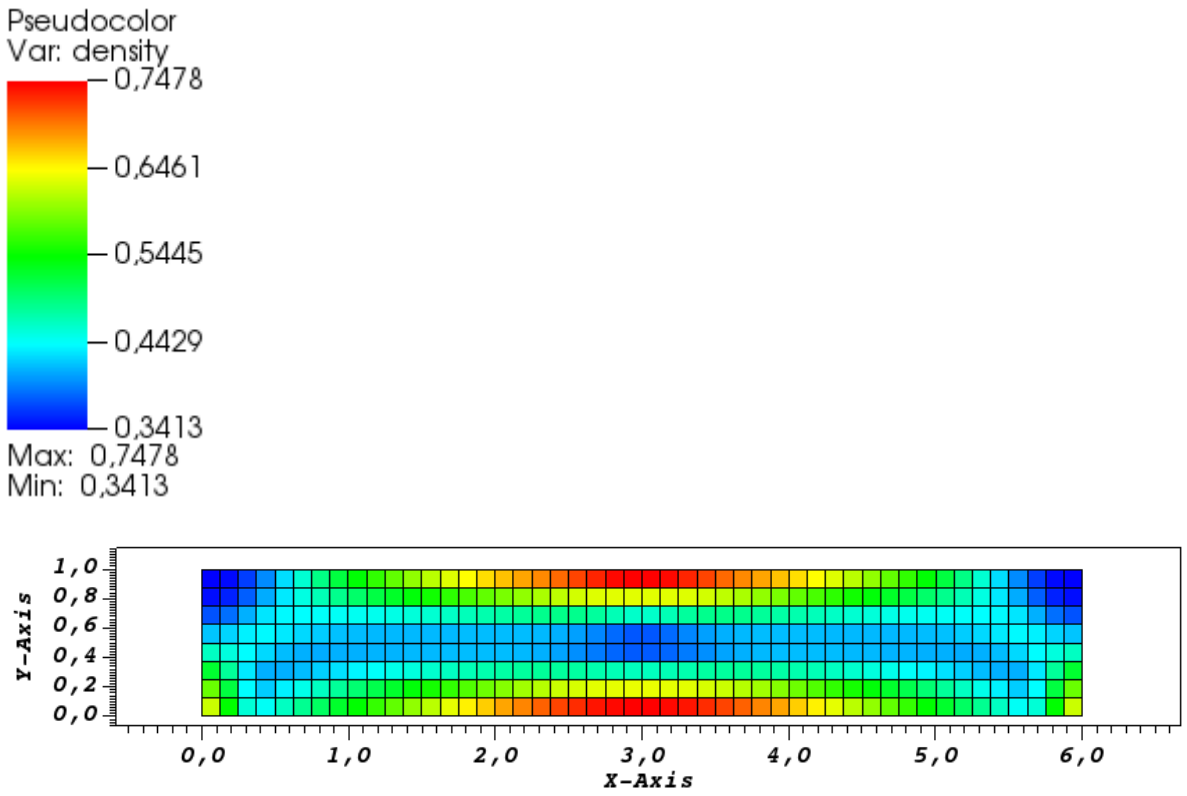


Рис.16

Ітерація 25

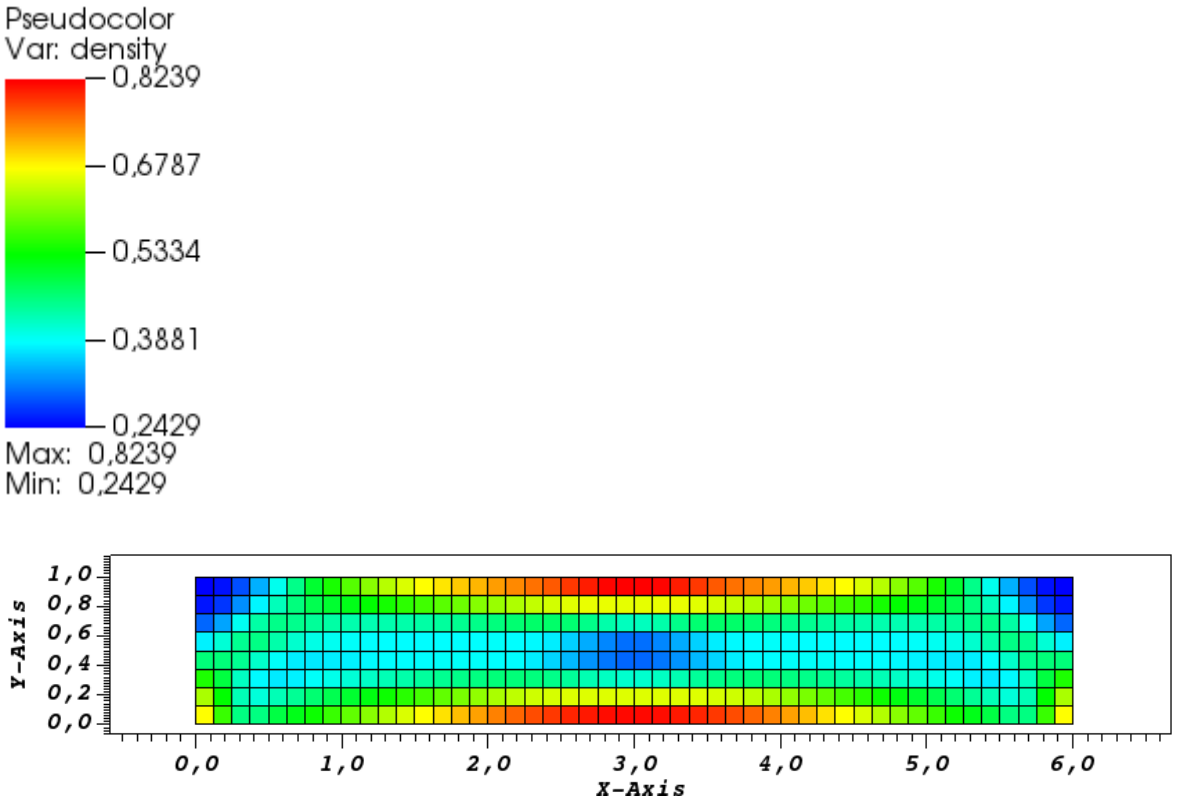


Рис.17

Ітерація 46

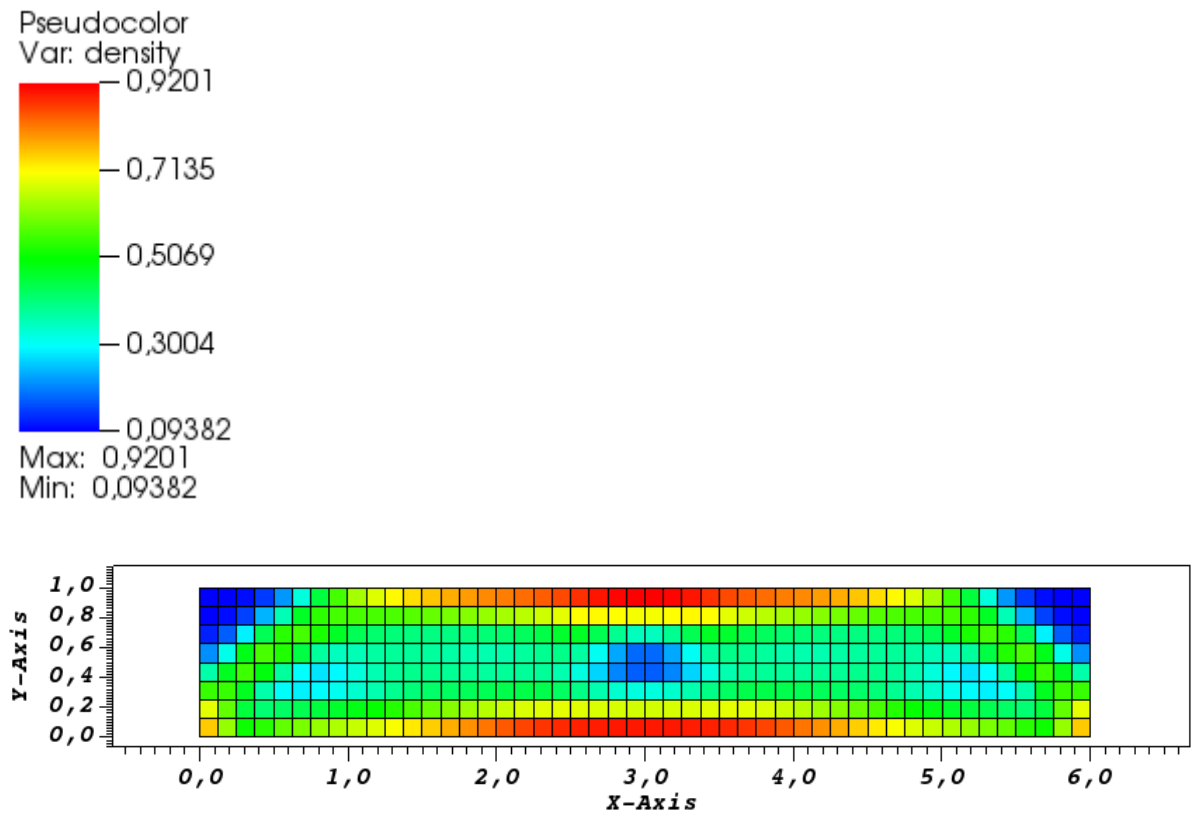
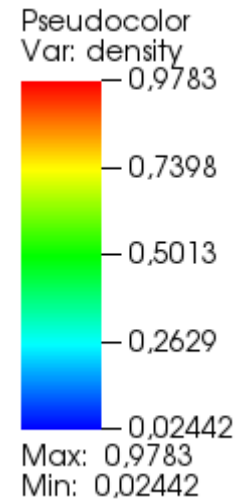


Рис.18

Ітерація 63



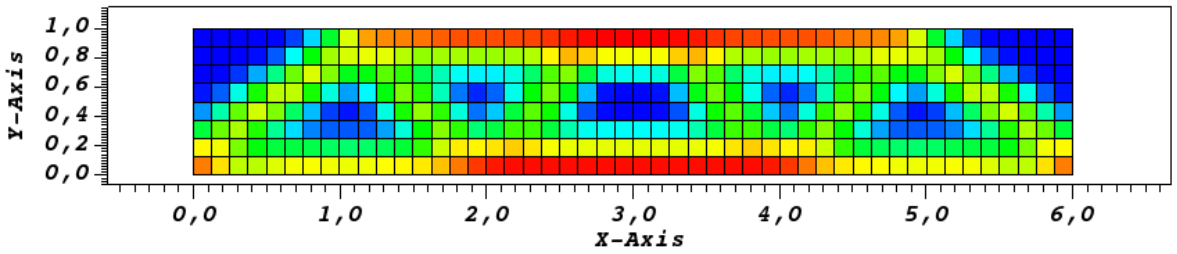
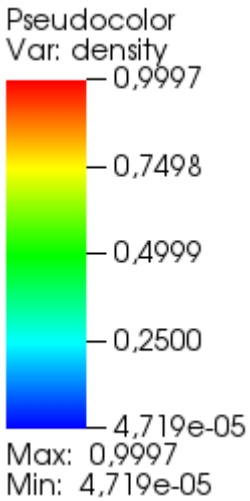


Рис.19

Ітерація 106



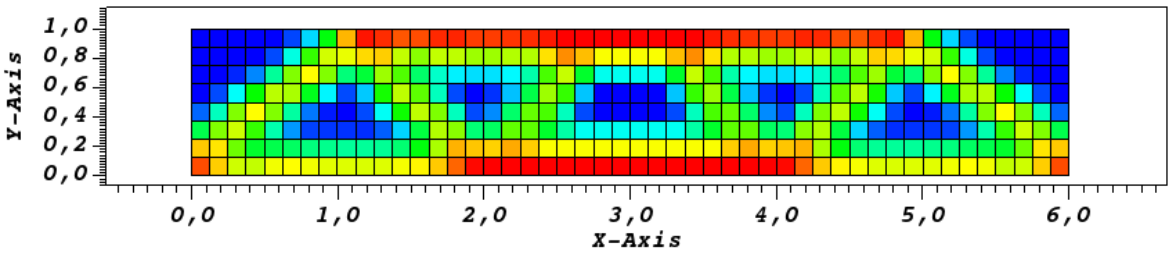


Рис. 20

4.2 Загальна оцінка отриманих результатів

В цілому, як результат оптимізації так і поведінка розглянутого об*єкту схожа з результатами інших досліджень, тобто з результатами використання інших алгоритмів. **[4]**

Тому, на мою думку, розв’язок повністю відповідає дійсності.

Зокрема, із наданих зображень графіку тестової норми ККТ ми можемо наочно побачити оптимізацію.

Однак, судити про перевагу паралелізованого алгоритму можна тільки частково, оскільки прискорення відбувається практично незначне.

Висновок

У цій дипломній роботі, було розглянуто та розв'язано задачу теорії пружності, задачу топологічної оптимізації, та порівняно розпаралелений та однопотоковий алгоритм.
Розв’язок задачі топологічної оптимізації в цілому, відкриває нові горизонти для стандартного та адитивного методів виробництв, так як дозволяє зменшити собівартість кінцевого продукту завдяки зменшенню витрат матеріалів на його виробництво, не завдаючи удару якості продукту.

Список використаної літератури

1. *Yashchuk Yu. O. 1 , Tajs-Zielinska K.* "Solving topology optimization problems using cellular automata and mortar finite element method" / – Ivan Franko National University of Lviv, 1 Universytetska Str., 79000, Lviv, Ukraine 2, Cracow University of Technology, al. Jana Pawla II 37, 31-864, Cracow, Poland
2. DEAL.II - an open source finite element library. Режим доступу посилання. https://www.dealii.org/
3. *fenicsproject.org* FEniCS Project. Режим доступу посилання. https://fenicsproject.org/pub/tutorial/pdf/fenics-tutorial-vol1.pdf
4. Sigmund, O.: A 99 Line Topology Optimization Code Written in MATLAB. Structural and Multidisciplinary Optimization 21, 120-127
5. Daniel Arndt, Wolfgang Bangerth, Bruno Blais, Marc Fehling, Rene Gassmöller, Timo Heister, Luca Heltai, Uwe Köcher, Martin Kronbichler, Matthias Maier, Peter Munch, Jean-Paul Pelteret, Sebastian Proell, Konrad Simon, Bruno Turcksin, David Wells, Jiaqi Zhang
   The deal.II Library, Version 9.3
6. *Савула Я. Г.* Числовий аналіз задач математичної фізики варіаційними методами./ – Львів: Видавничий центр ЛНУ імені Івана Франка, 2004. – 221 с.
7. *Hutton, David V.* Fundamentals of finite element analysis / – McGrawHill, 2004. – 494 p.
8. *FAE visualisation - Finite element method.* Режим доступу посилання. https://en.wikipedia.org/wiki/Finite_element_method#/media/File:FAE_visualization.jpg
9. Para-View - Visualization software. Режим доступу посилання. https://www.paraview.org/
10. Vizit - Visualization software. Режи доступу посилання. https://visit-dav.github.io/visit-website/
11. Jorge Nocedal, Steven J. Wright "Numerical optimization. Second edition" Springer 2006
12. A. Plaza De La Hoz "The fractal behaviour of triangular refined/derefined meshes"
13. Шахно С. М. Практикум з чисельних методів: навч.посібник / Шахно С.М., Дудикевич А.Т., Левицька С.М. - Львів: ЛНУ імені Івана Франка, 2013. - 432 с.