ПРИКЛАДНА МАТЕМАТИКА

УДК 519.6 *http://dx.doi.org/10.30970/vam.2024.34.00000*

ПІДВИЩЕННЯ ТОЧНОСТІ ЧИСЕЛЬНОГО ІНТЕГРУВАННЯ В МСЕ З ВИКОРИСТАННЯМ НЕЙРОННИХ МЕРЕЖ

Д. Альянах, І. Дияк

Львівський національний університет імені Івана Франка,
вул. Університетська 1, Львів, 79000, Україна
e-mail: dmytro.alianakh@lnu.edu.ua, ivan.dyyak@lnu.edu.ua

У роботі використовуються методи машинного навчання для вибору оптимального порядку інтегрування елементів матриці жорсткості методу скінченних елементів для задачі осесиметричної теорії пружності з використанням чотирикутних сирендипових елементів. Розглянуто два підходи до підвищення точності інтегрування локальних матриць жорсткості: класифікатор, який прогнозує мінімальну кількість точок Гаусса–Лежандра, необхідну для досягнення заданої точності відносно еталонного інтегрування, та регресор, який оцінює вибір оптимальних вузлів та ваг квадратури. Навчальні вибірки згенеровано синтетично з нормалізацією геометрії елементів і обмеженнями на кути та координати. Обчислювальні експерименти показали, що наближення локальних матриць жорсткості до еталона в сенсі матричних норм не гарантує помітного зменшення похибки переміщень. У більшості проведених експериментів покращення від підбору вузлів та ваг не перевищували $10^{-3}\%$. Підбір кількості точок інтегрування підвищує точність обчислення елементів матриці жорсткості завдяки збільшенню порядку квадратури у випадках, де цього вимагає геометрична нелінійність відображення.

Ключові слова: рівняння в частинних похідних, осесиметрична задача теорії пружності, метод скінченних елементів, квадратури Гаусса–Лежандра, серендипові чотирикутні елементи, нейронні мережі, оптимізація рою частинок.

1. Вступ

Метод скінченних елементів (МСЕ) є одним з потужних методів чисельного розв'язування крайових задач математичної фізики, як лінійних так і нелінійних. Для отримання точнішого розв'язку в МСЕ зазвичай або підвищують порядок базисних апроксимацій, або збільшують густину сітки. Обидва підходи призводять до зростання розмірності системи, а отже збільшення використовуваної пам'яті та часу розв'язку задачі. Оскільки сітка в МСЕ складається не з квадратних скінченних елементів (СЕ), а з чотирикутних СЕ, межі яких можуть мати значну кривину, то використовують ізопараметричне перетворення, яке зводить обчислення елементів локальної матриці жорсткості до інтегрування на квадраті $[-1,1]\times[-1,1]$, для якого, зазвичай, використовується квадратура Гаусса–Лежандра.

У роботах [1] і [2] підбираються тільки ваги квадратурної формули, а в [3] порівнюються варіанти з підбором тільки ваг квадратури та тільки вузлів інтегрування або одне і друге одночасно. Оптимізація ваг і вузлів одночасно дає кращий результат. В [1] використовується сітка для генерування можливих значень ваг і виконується їх перебір для визначення оптимального значення. В [4, 5] використовують нейронну мережу з певними доданками у функції втрат для знаходження

оптимальних точок і ваг без набору даних з попередньо обчисленими оптимальними точками і вагами. В [2] квадратуру представляють як лінійний шар нейронної мережі, де на вхід є значення підінтегральної функції у вузлах квадратури, а матриця ваг є діагональною і її елементи є тренувальними параметрами. На виході отримують значення елемента локальної матриці жорсткості. У функції втрат мінімізується різниця між локальною матрицею жорсткості, отриманою при 30 вузлах квадратури і при сталій малій кількості вузлів, але з підібраними вагами. Після оптимізації, значення ваг діагональної матриці якраз і будуть оптимальними вагами квадратури. Також в [1], [3], [6] і [7] розглядають оптимізацію кількості вузлів квадратури. В [8] використовують різні графові нейронні мережі для класифікації NURBS-елементів (Non-Uniform Rational B-Splines — нерівномірні раціональні B-сплайни) відповідно до оптимальної кількості точок інтегрування. В [9] оптимізують і кількість і точки з вагами: у $1D$ – аналітично (правило «півточок»), а у $2D/3D$ – чисельно на макроелементах для точного інтегрування.

В наведених роботах акцентують увагу на точності обчислення локальних матриць жорсткості, але більшість не показує реальний приріст точності обчислення шуканих невідомих функцій (переміщень) для конкретних задач з певними областями і параметрами. У наведених роботах розглядають переважно елементи з лінійною апроксимацією та межами з невеликою кривиною.

Метою даної роботи є покращення точності обчислень переміщень в осесиметричній задачі, як наслідок підвищення точності інтегрування у матриці жорсткості для білінійних і біквадратичних СЕ. Це досягається на основі двох підходів: (i) прогнозування мінімальної кількості точок квадратури для обчислення інтеграла з заданою точністю з врахуванням форми скінченного елемента; (ii) прогнозування оптимальних значень для точок і ваг квадратури без збільшення порядку квадратури з врахуванням форми скінченного елемента. У роботі наведені результати щодо підвищення точності інтегрування для осесимитричної задачі теорії пружності у різних областях.

2. Методологія

У МСЕ для пошуку невідомих переміщень задача теорії пружності зводиться до розв'язування системи лінійних алгебраїчних рівнянь (СЛАР), яка складається з глобальної матриці жорсткості та вектора сил. Глобальна матриця жорсткості утворюється з локальних матриць СЕ, які обчислюються у випадку осесиметричної задачі теорії пружності за формулою [10]:

$$[\mathbf{k}^e] = \int_{\Omega^e} \mathbf{B}^T \, \mathbf{D} \, \mathbf{B} \; r \, \mathrm{d}r \, \mathrm{d}z, \tag{1}$$

де $\mathbf{B}$ — матриця похідних функцій форми скінченного елемента за координатами, $\mathbf{D}$ – матриця пружних коефіцієнтів, Ω^e — скінченний елемент.

Після використання ізопараметричного перетворення [10] формула набуде вигляду:

$$[\mathbf{k}^e] = \int_{-1}^{1} \int_{-1}^{1} \big[\mathbf{B}(\xi,\eta)\big]^T \, \mathbf{D} \, \big[\mathbf{B}(\xi,\eta)\big] \; r^e(\xi,\eta) \, \big|\mathbf{J}(\xi,\eta)\big| \, \mathrm{d}\xi \, \mathrm{d}\eta, \tag{2}$$

$$r^e = \sum_{i=1}^{s} R_i \, N_i^e(\xi, \eta), \tag{3}$$

де R_i – радіальна координата i-го вузла елемента, N_i^e – функція форми, асоційована з i-тим вузлом, s – кількість вузлів у скінченному елементі. Тут $\mathbf{J}(\xi, \eta)$ – матриця Якобі ізопараметричного відображення $(\xi, \eta) \mapsto (r, z)$ для елемента Ω_e. Координатне перетворення задаємо як

$$\mathbf{x}(\xi, \eta) = \begin{pmatrix} r(\xi, \eta) \\ z(\xi, \eta) \end{pmatrix} = \sum_{i=1}^{s} \mathbf{X}_i \, N_i^e(\xi, \eta), \qquad \mathbf{X}_i = \begin{pmatrix} R_i \\ Z_i \end{pmatrix},$$

де N_i^e — функції форми, R_i, Z_i — координати вузлів. Тоді матриця Якобі визначається як:

$$\mathbf{J}(\xi, \eta) = \begin{pmatrix} \dfrac{\partial r}{\partial \xi} & \dfrac{\partial r}{\partial \eta} \\ \dfrac{\partial z}{\partial \xi} & \dfrac{\partial z}{\partial \eta} \end{pmatrix} = \sum_{i=1}^{s} \begin{pmatrix} R_i \, N_{i,\xi} & R_i \, N_{i,\eta} \\ Z_i \, N_{i,\xi} & Z_i \, N_{i,\eta} \end{pmatrix}, \qquad |\mathbf{J}| = \det \mathbf{J}.$$

Тут $N_{i,\xi} = \partial N_i^e / \partial \xi$, $N_{i,\eta} = \partial N_i^e / \partial \eta$. Для переходу до похідних за фізичними координатами використовується

$$\begin{pmatrix} N_{i,r} \\ N_{i,z} \end{pmatrix} = \mathbf{J}^{-1}(\xi, \eta) \begin{pmatrix} N_{i,\xi} \\ N_{i,\eta} \end{pmatrix},$$

а матриця $\mathbf{B}(\xi, \eta)$ формується з $N_{i,r}, N_{i,z}$. Додатковий множник $r^e(\xi, \eta)$ у підінтегральному виразі відповідає осесиметричній постановці.

Для обчислення інтеграла в (2) використаємо двовимірну квадратуру Гаусса–Лежандра [11]:

$$\int_{-1}^{1} \int_{-1}^{1} f(r, z) \, \mathrm{d}r \, \mathrm{d}z \;\approx\; \sum_{i=1}^{k} \sum_{j=1}^{k} f(l_i, l_j) \, w_i \, w_j, \tag{4}$$

де l_i та l_j — корені полінома Лежандра степеня k, w_i та w_j — відповідні їм ваги.

Коли розглядається квадратний СЕ, то інтегрувати (2) за допомогою (4) можна з високою точністю для малої кількості вузлів. Коли межі СЕ мають значну кривину, то ізопараметричне перетворення є нелінійним і потребує більшої кількості точок інтегрування.

Можливі два варіанти покращення точності інтегрування, а саме – збільшення кількості вузлів інтегрування або підбір ваг і вузлів інтегрування.

3. Генерація набору даних

3.1. Генерування чотирикутних елементів

Замість того, щоб подавати нейронній мережі на вхід необроблені дані, тобто координати вершин чотирикутника, ми застосовуємо попереднє опрацювання ознак: нормалізуємо елемент (паралельне перенесення, поворот і масштабування), фіксуємо базові вузли. Нехай задано чотирикутних в системі координат (r, z) з вузлами (r_1, z_1), (r_2, z_2), (r_3, z_3), (r_4, z_4). Два нижні вузли чотирикутника фіксуємо в точках $(r_1, z_1) = (4, 0)$ та $(r_4, z_4) = (6, 0)$. Саме такі координати були вибрані, щоб мати достатньо місця для розміщення елементів з різної форми (з різною кривиною меж)

так, щоб координати по осі r не були близькі до 0 і щоб довжина нижнього ребра була рівна 2. Для чотирикутників точки (r_2, z_2) і (r_3, z_3) генеруємо в області $[0.5, 10] \times [0.5, 12]$ м. У випадку біквадратичної апроксимації генеруються два чотирикутники: один із прямим ребром, інший – зі зміщенням середньої точки ребра. Зміщення задаємо генератором псевдовипадкових чисел у діапазоні $[-0.15, 0.15]$ м по обох осях відносно середини прямого ребра. Зміщення точки в центрі застосовуємо лише для лівого та правого ребер, оскільки порівняння виконуватиметься саме в областях із такою кривиною меж. Зміщення для точки по центру робимо тільки для лівого та правого ребра, оскільки саме на областях із такою кривиною меж виконуватиметься порівняння.

Генерування нормалізованих чотирикутників, як наведено в [1–3], по координаті r не є можливим, бо підінтегральна функція у випадку осесиметричної задачі містить множник $1/r$. Задамо такі якісні обмеження на чотирикутники при генерації: мінімальний кут – 5°, максимальний – 175°, чотирикутник має бути опуклим.

До згенерованих чотирикутників також додаємо видовжені по осі r і по осі z прямокутники.

Перед тим як передати на вхід нейронній мережі координати точок, потрібно виконати алгоритм нормалізації чотирикутного елемента, який можна записати у вигляді:

$$\begin{aligned}
\mathbf{v} &= (v_r, v_z) = \mathbf{p_1} - \mathbf{p_0}, \\
L &= \|\mathbf{v}\| = \sqrt{v_r^2 + v_z^2}, \\
\theta &= -\operatorname{atan2}(v_z, v_r), \\
\mathbf{R} &= \begin{pmatrix} \cos\theta & -\sin\theta \\ \sin\theta & \cos\theta \end{pmatrix}, \\
\mathbf{q_i} &= \frac{\mathbf{R}\,(\mathbf{p_i} - \mathbf{p_0})}{L}, \quad i = 0, 1, 2, 3 \\
T &= 2, \\
\tilde{\mathbf{p}}_\mathbf{i} &= \mathbf{q_i}\, T + (4, 0),
\end{aligned}$$

де $\mathbf{p_0}$ – вектор координат першої фіксованої точки; $\mathbf{p_1}$ – вектор координат другої фіксованої точки; T – масштабний коефіцієнт цільової довжини базового відрізка після нормалізації; $(4,\ 0)$ – вектор координат $\mathbf{p_0}$ після нормалізації; $\operatorname{atan2}(z, r)$ – двоаргументна функція арктангенсу, що повертає кут $\varphi \in (-\pi, \pi]$ від додатного напряму осі r до вектора (r, z). У нашому випадку $\varphi = \operatorname{atan2}(v_z, v_r)$, а кут обертання, який застосовуємо до точок, $\theta = -\varphi = -\operatorname{atan2}(v_z, v_r)$. Функція $\operatorname{atan2}(z, r)$ описується формулою:

$$\operatorname{atan2}(y, x) = \begin{cases}
\arctan\left(\dfrac{y}{x}\right), & x > 0, \\
\arctan\left(\dfrac{y}{x}\right) + \pi, & x < 0 \text{ та } y \geq 0, \\
\arctan\left(\dfrac{y}{x}\right) - \pi, & x < 0 \text{ та } y < 0, \\
\dfrac{\pi}{2}, & x = 0 \text{ та } y > 0, \\
-\dfrac{\pi}{2}, & x = 0 \text{ та } y < 0, \\
\text{не визначена}, & x = 0 \text{ та } y = 0.
\end{cases}$$

Також за потреби здійснюємо осьове симетричне відображення елемента відносно осі r. Нехай $\bar{z} = \frac{1}{s}\sum_{i=0}^{s-1}(\tilde{p_i})_z$ та $\varepsilon = 10^{-7}$, $(\tilde{p_i})_z$ – координата z точки $\tilde{\mathbf{p}}_\mathbf{i}$. Якщо $\bar{z} < -\varepsilon$, то віддзеркалюємо точки відносно осі r:

$$\mathbf{S} = \begin{pmatrix} 1 & 0 \\ 0 & -1 \end{pmatrix}, \qquad \tilde{\mathbf{p}}_\mathbf{i} \leftarrow \mathbf{S}\,\tilde{\mathbf{p}}_\mathbf{i}, \quad i = 0, \ldots, M-1,$$

де M – кількість точок у чотирикутнику.

Таким чином використовується матриця повороту для вирівнювання елемента, потім виконується його масштабування, паралельне перенесення та за потреби здійснюємо осьове симетричне відображення елемента відносно осі r.

Для пришвидшення генерації знаходження отримальних значення кількості вузлів чи значень вузлів та ваг для кожного чотирикутника було використано багатопотоковість (20 потоків).

Перегенеровувати набір даних для тренування потрібно у випадку коли змінилися наступні параметри: матриця $\mathbf{B}$ або $\mathbf{D}$ (якщо коефіцієнти матриці перестали бути сталими), розмірність задачі зросла, межі кривини СЕ в МСЕ виходять за допустимі в згенерованому наборів даних або коли змінився порядок апроксимації.

3.2. Знаходження оптимальної кількості точок квадратури

Для знаходження оптимальної кількості вузлів інтегрування, будемо почергово порівнювати норму модуля різниці матриць жорсткості елемента, отриманої при 30 вузлах і на поточній кількості вузлів інтегрування – від 2 (у випадку біквадратичних апроксимацій від 3) до 30. Критерієм зупинки перебору кількості точок інтегрування є досягнення порогового значення норми (максимальна різниця матриць жорсткості елемента при різній кількості точок квадратури) 10^{-7}. Генеруємо 200 тисяч вхідних чотирикутників (з яких 10 тисяч прямокутників) і вихідних оптимальних значень кількості точок інтегрування.

На рис. 1 бачимо, що при збільшенні точності, розподіл зсувається вправо до більшої кількості точок.

3.3. Знаходження оптимальних точок і ваг квадратурної формули

Для знаходження оптимальних значень точок і ваг квадратури застосуємо еволюційний алгоритм PSO (particle swarm optimization, оптимізація рою частинок) [12, 13]. Нехай потрібно оптимізувати N_p параметрів. Тоді i-тий індивід та його швидкість на n-тій генерації подаємо у вигляді одновимірних векторів

$$\mathbf{x}_i^n = \left(x_{i,1}^n,\, x_{i,2}^n,\, x_{i,3}^n, \ldots, x_{i,N_p-2}^n,\, x_{i,N_p-1}^n,\, x_{i,N_p}^n\right), \tag{5}$$

$$\mathbf{v}_i^n = \left(v_{i,1}^n,\, v_{i,2}^n,\, v_{i,3}^n, \ldots, v_{i,N_p-2}^n,\, v_{i,N_p-1}^n,\, v_{i,N_p}^n\right), \tag{6}$$

де $x_{i,j}^n$ — j-та координата в просторі $\mathbb{R}^{N_p}$ для i-ого індивіда, $v_{i,j}^n$ — відповідна компонента швидкості. Координати відповідають параметрам квадратури (точки і ваги), тобто формат запису індивіда такий $(p_1^r, p_1^z, w_1, p_2^r, p_2^z, w_2, ...)$, де p_i — номер пари точка-вага в квадратурі.

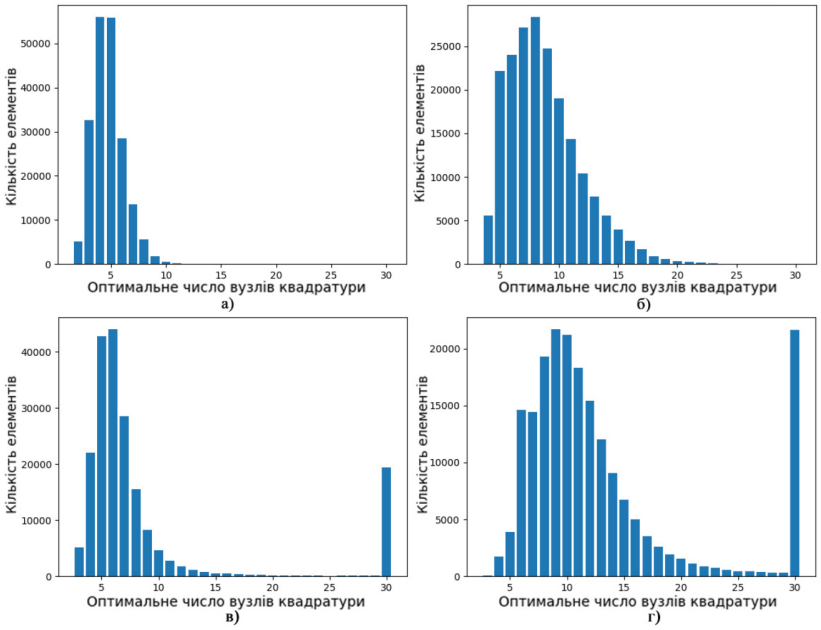

Рис. 1. Розподіл оптимальної кількості точок інтегрування: білінійний елемент для 200 000 згенерованих чотирикутників: порогові значення а) 10^{-3} та б) 10^{-7}; біквадратичний елемент для 100 000 згенерованих чотирикутників: порогові значення в) 10^{-3} та г) 10^{-7}

Оновлення положення та швидкості:

$$\mathbf{v}_i^n = \alpha\,\mathbf{v}_i^{n-1} + \beta\big(\mathbf{g}^n - \mathbf{x}_i^n\big)U_1 + \gamma\big(\mathbf{p}_i^n - \mathbf{x}_i^n\big)U_2, \tag{7}$$

$$\mathbf{x}_i^{n+1} = \mathbf{x}_i^n + \mathbf{v}_i^n, \tag{8}$$

де $\mathbf{g}^n$ — найкращий індивід популяції, $\mathbf{p}_i^n$ — найкраще положення i-го індивіда до генерації n включно, α, β, γ — сталі, U_1, U_2 — випадкові величини, рівномірно розподілені на $[0, 1]$. Замість одного з індивідів включаємо найкращий індивід за всі генерації.

Оптимальні значення $\alpha = 0.729$ та $\beta = \gamma = 1.49445$ обрано згідно з [14, 15]. Розмір популяції в алгоритмі було вибрано 250.

Одного початкового індивіда $\mathbf{x}_i^0$ беремо з квадратури Гаусса–Лежандра; інші початкові значення отримуємо додаванням до квадратурних вузлів випадкового збурення, рівномірно розподіленого на відрізку $[-0.1, 0.1]$. Початкові швидкості $\mathbf{v}_i^0$ задаємо як незалежні випадкові вектори, компоненти яких рівномірно розподілені на $[-0.001, 0.001]$.

Функція придатності:

$$\mathrm{Fitness}(\boldsymbol{\xi}, \boldsymbol{\eta}, \mathbf{w}) = \frac{L(\boldsymbol{\xi}^0, \boldsymbol{\eta}^0, \mathbf{w}^0)}{L(\boldsymbol{\xi}, \boldsymbol{\eta}, \mathbf{w})}, \qquad L(\boldsymbol{\xi}, \boldsymbol{\eta}, \mathbf{w}) = \big\|\mathbf{k}^e(\boldsymbol{\xi}, \boldsymbol{\eta}, \mathbf{w}) - \mathbf{k}^{e,\mathrm{exact}}\big\|, \tag{9}$$

де $\mathbf{k}^{e,\mathrm{exact}}$ — «точна» матриця жорсткості, обчислена за допомогою 30 вузлів у кожному напрямку (900 точок на елемент); $\boldsymbol{\xi}^0, \boldsymbol{\eta}^0$ — вузли та $\mathbf{w}^0$ — ваги Гаусса–

Лежандра; $\boldsymbol{\xi}, \boldsymbol{\eta}$ — вузли та $\mathbf{w}$ — ваги, для яких обчислюємо відповідні функції; $\|\cdot\|$ — сума квадратів елементів (норма Фробеніуса).

Якщо Fitness > 1, то знайдені точки й ваги дають точніше значення інтеграла, ніж звичайні точки і ваги квадратури Гаусса–Лежандра.

Критерії зупинки: 500 ітерацій або відсутність покращення протягом останніх 100. Оскільки оптимізація кожного елемента незалежна, обчислення легко розпаралелити; ефективне використання масивів з бібліотеки `NumPy` суттєво прискорює обчислення матриці жорсткості. Експерименти виконано на 20-потоковому процесорі Intel® Core™ i9-12900H. Генерація 1.2×10^4 (експоненційна форма запису числа, де «×» позначає множення) зразків (із них 2000 для прямокутників) для білінійної апроксимації тривала ≈ 7 год, для біквадратичної ≈ 17 год.

4. Тренування нейронних мереж

Для навчання нейронної мережі була використана бібліотека PyTorch на Python.

4.1. Оптимальна кількість точок квадратурної формули

Вхідними ознаками нейронної мережі будуть координати вільних вузлів згенерованих елементів. Для задання кількості вихідних вузлів використаємо індикаторне кодування ознак (one-hot encoding) [16]. Це спосіб представлення категоріальних змінних у вигляді бінарних векторів. Для змінної, що має C різних категорій, кожній категорії відповідає унікальний вектор завдовжки C, у якому саме один елемент набуває значення 1, а всі інші 0. Функція втрат являє собою перехресну ентропію, яка задається формулою

$$\ell(\mathbf{y}, \mathbf{p}) = -\frac{1}{N} \sum_{j=1}^{N} \sum_{i=1}^{C} y_{i,j} \, \ln\bigl(p_{i,j}\bigr),$$

яка вимірює різницю між істинним виродженим розподілом (із масою 1 у правильному класі) $\mathbf{y_j}$ та передбаченим моделлю розподілом ймовірностей $\mathbf{p_j}$. Тут N – кількість прикладів у вибірці, C – кількість класів, $\mathbf{y} = (y_{i,j}) \in \{0,1\}^{C \times N}$ – індикаторна матриця (для кожного j вектор $\mathbf{y_j} = (y_{1,j}, \ldots, y_{C,j})^\top$ має рівно одну «1»), $\mathbf{p} = (p_{i,j}) \in [0,1]^{C \times N}$ – матриця передбачених ймовірностей (для кожного j вектор $\mathbf{p_j} = (p_{1,j}, \ldots, p_{C,j})^\top$ задовольняє $\sum_{i=1}^{C} p_{i,j} = 1$), i – індекс класу, j – індекс прикладу, $\ell(\mathbf{y}, \mathbf{p})$ – середнє значення перехресної ентропії по вибірці.

Як функцію активації було взято $tanh$, нейронна мережа має 3 приховані шари по 80 нейронів кожен, швидкість навчання 10^{-3}, параметр зменшення ваг 10^{-4}, їх було визначено емпірично. В якості алгоритму оптимізації був вибраний AdamW [17]. Набір даних був розділений на тренувальний 80%, валідаційний 10% та тестовий 10%.

Для випадку біквадратичних апроксимацій було проведено 10000 епох тренування, для білінійного 5000. Втрати після тренування для білінійного випадку на тренувальній вибірці: 2.1243, валідаційній: 2.1418, тестовій: 2.1355. Для біквадратичного випадку на тренувальній вибірці: 1.1109, валідаційній: 1.2939, тестовій: 1.2856.

4.2. Оптимальні точки і ваги квадратурної формули

На вхід нейронної мережі подаємо вектор координат вільних вузлів елемента. На вихід очікуємо вектор з оптимальних точок і ваг квадратури. Як функцію активації було взято tanh, нейронна мережа має 3 приховані шари по 80 нейронів кожен, швидкість навчання 10^{-3}, параметр зменшення ваг 10^{-4}, визначені емпірично. В якості алгоритму оптимізації був вибраний AdamW [17]. Набір даних був розділений на тренувальний 80%, валідаційний 10% та тестовий 10%. Функція втрат це середньоквадратична похибка між очікуваним вектором точок і ваг та передбаченим нейронною мережею:

$$\mathrm{MSE}(\mathbf{y}^{\mathrm{exp}}, \mathbf{y}^{\mathrm{pred}}) = \frac{1}{m}\sum_{i=1}^{m}(y_i^{\mathrm{exp}} - y_i^{\mathrm{pred}})^2,$$

де $\mathbf{y}^{\mathrm{exp}}$ — вектор довжини m очікуваних значень цільової змінної, $\mathbf{y}^{\mathrm{pred}}$ — вектор довжини m відповідних прогнозів нейромережі для того самого прикладу.

Було проведено 7570 епох тренування для білінійного випадку (0.35 хв) і 4595 епох для біквадратичного випадку (0.25 хв) і досягнуто порогового значення функції втрат на тренувальній вибірці 7.5×10^{-3}. Втрати для білінійного випадку на тренувальній вибірці 7.5×10^{-3}, на валідаційній 7.6×10^{-3} і на тестовій 7.6×10^{-3}. Для біквадратичного випадку на тренувальній вибірці 6.9×10^{-3}, на валідаційній 6.4×10^{-3} і на тестовій 6.5×10^{-3}.

Для білінійного випадку нейронна мережа буде мати 3 прихованих шари по 80 нейронів, 4 числа на вхід та 12 чисел на вихід. Для біквадратичного випадку нейронна мережа буде мати 3 прихованих шари по 80 нейронів, 8 чисел на вхід та 27 чисел на вихід.

Для випадку, коли апроксимуються точки і ваги квадратурної формули нейронною мережею, кількість виходів нейронної мережі можна обчислити за формулою $b^d(d+1)$, де b – кількість точок квадратури по одній осі, d – вимірність скінченного елемента.

На рис. 2 зображено розподіл значень функції придатності для згенерованих 12000 білінійних та 12000 біквадратичних чотирикутників.

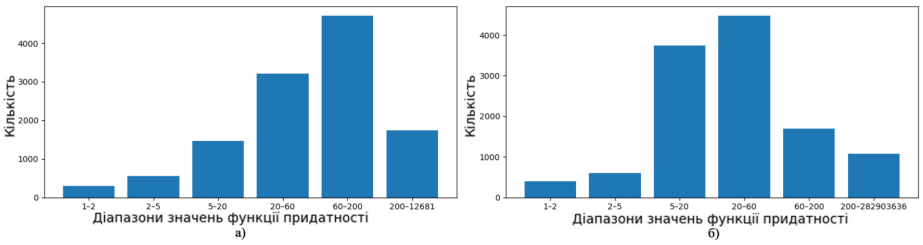

Рис. 2. Діапазони значень функції придатності для випадку а) білінійної та б) біквадратиної апроксимації з відповідною кількістю елементів, для якої змогли досягнути таких значень

5. Результати обчислювальних експериментів

Надалі всі величини подаються в безрозмірній формі. Нехай L_* — характерна довжина (наприклад, для прямокутника — його ширина; для кільця — внутрішній радіус), а $\sigma_* = \mu$ — характерна пружна стала з розмірністю напружень (модуль зсуву).

Вводимо безрозмірні змінні

$$\hat{r} = \frac{r}{L_*}, \quad \hat{z} = \frac{z}{L_*}, \quad \hat{\mathbf{u}} = \frac{\mathbf{u}}{L_*}, \quad \hat{\boldsymbol{\sigma}} = \frac{\boldsymbol{\sigma}}{\sigma_*}, \quad \hat{P} = \frac{P}{\sigma_*}.$$

Тут $\hat{r}$, $\hat{z}$ – безрозмірні радіальна і осьова координати відповідно; r, z – радіальна і осьова координати; $\hat{\mathbf{u}}$ – безрозмірний вектор переміщень, $\mathbf{u}$ – вектор переміщень; $\hat{\boldsymbol{\sigma}}$ – безрозмірний тензор напружень, $\boldsymbol{\sigma}$ – тензор напружень; $\hat{P}$ – безрозмірний тиск (навантаження), P – тиск.

Тоді отримані переміщення наводимо у частках L_*, а напруження – у частках σ_*.

Розглянемо осесиметричну задачу з такими безрозмірними параметрами:

$$\hat{\mu} = 1, \qquad \nu = 0.3, \qquad \hat{P} = \frac{1}{0.7} \times 10^{-5}.$$

Тут $\hat{\mu} = 1$ випливає з вибору $\sigma_* = \mu$. Оберемо $L_* = 1$ м. Коефіцієнти матриці $\mathbf{D}$ сталі в безрозмірній постановці та залежать лише від ν. Завдяки лінійності теорії пружності розв'язок масштабується лінійно за навантаженням:

$$\hat{\mathbf{u}}(\hat{P}) = \hat{P}\,\hat{\mathbf{u}}(1), \qquad \hat{\boldsymbol{\sigma}}(\hat{P}) = \hat{P}\,\hat{\boldsymbol{\sigma}}(1).$$

де $\hat{\mathbf{u}}(\hat{P})$ та $\hat{\boldsymbol{\sigma}}(\hat{P})$ – відповідно поля переміщень і напружень при значенні параметра навантаження $\hat{P}$. Результати для інших значень тиску отримуються масштабуванням.

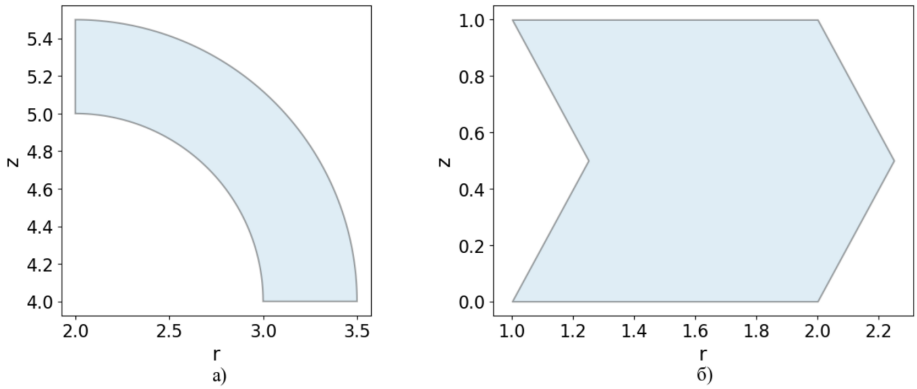

Рис. 3. Геометрії областей у фізичних координатах (r, z): а) чверть кільця з центром $(2, 4)$; внутрішній радіус 1 м, зовнішній 1.5 м; б) область із зигзагоподібними лівою та правою границями

Для оцінки покращення точності інтегрування розглядаємо такі області: прямокутна $\hat{r} \in [1, 2]$, $\hat{z} \in [0, 1]$; чверть кільця з центром $(2, 4)$ (Рис. 3а): внутрішній радіус 1, зовнішній 1.5; та область із зигзагоподібними лівою і правою границею (Рис. 3б).

Для всіх областей використано однакові безрозмірні крайові умови: на лівій границі прикладено тиск $\hat{P} = \frac{1}{0.7} \times 10^{-5}$ у радіальному напрямку (умова Неймана); на правій границі — нульовий тиск ($\hat{P} = 0$); на нижній та верхній границях задано нульове осьове переміщення $\hat{u}_z = 0$ (умова Діріхле). Для чверті кільця «ліва» та «права» межі відповідають внутрішній та зовнішній дузі.

5.1. Знаходження оптимальної кількості точок квадратури

Перша область це чверть кільця з сіткою 1x2 елементи з внутрішнім радіусом 1, зовнішнім 1.5 і центром в (2, 4). На цій області з використанням біквадратичних елементів вдалося зменшити похибку обчислених переміщенньь $\hat{u}_r$ на 0.5% при порівнянні значень переміщень з використанням 3 точок інтегрування для кожної осі та кількості точок, обчисленої нейронною мережею. Для прямокутних сіток 1x2, 8x1 та 20x1 з біквадратичною апроксимацією, вдалося зменшити відносну похибку на 2.5×10^{-2}% до 1.0×10^{-3}%, на 3.0×10^{-7}% та на 1.3×10^{-9}% до 1.0×10^{-10}% відповідно. Для білінійної апроксимації на прямокутній сітці 1x2 і на "зигзаг" сітці 1x2 вдалося зменшити відносну похибку на 3.5×10^{-1}% і на 2.5×10^{-1}% відповідно. Найбільше покращення спостерігається для біквадратичної апроксимації на чверті кільця, оскільки межі її елементів мають значну кривину.

5.2. Знаходження оптимальних точок і ваг квадратурної формули

Для підбору оптимальних точок і ваг квадратури навчалася нейронна мережа, яка точніше обчислює інтеграли для локальної матриці жорсткості, а відповідно її саму.

Таблиця 1

Порівняння значень $\hat{u}_r$ (невідомих радіальних переміщень) при використанні 2 та 30 точок і ваг Гаусса–Лежандра та підібраних 2 точок і ваг для одного білінійного елемента. Перша колонка вказує на порядковий індекс точки в сітці. NN точки — точки і ваги, обчислені нейронною мережею

Індекс	$\hat{u}_r$ (2 точки)	$\hat{u}_r$ (30 точок)	$\hat{u}_r$ (NN точки)
0	2.2322×10^{-4}	2.2317×10^{-4}	2.2290×10^{-4}
1	1.9244×10^{-4}	1.9255×10^{-4}	1.9231×10^{-4}
2	1.7980×10^{-4}	1.7988×10^{-4}	1.7963×10^{-4}
3	1.5424×10^{-4}	1.5419×10^{-4}	1.5398×10^{-4}

Матриця жорсткості була обчислена точніше в сенсі норми, але значення переміщень не значно покращилися. Найбільше покращення на 1.1×10^{-3}% було на прямокутній сітці 1x2. На інших розглянутих областях значення переміщень залишилося приблизно такими самими.

Також був обчислений приріст точності обчислення переміщень для білінійного елемента з межами зі значною кривиною, в якого було найбільше значення функції придатності, а саме 12681. Було розглянуто сітку 1x1 елемент. Максимальний модуль різниці елементів для $\mathbf{k}^{e,\text{exact}}$ і $\mathbf{k}^{e,2}$ становить ≈ 0.97, в той час як для

$\mathbf{k}^{e,\text{exact}}$ і $\mathbf{k}^{e,\text{NN}}$ максимальний модуль різниці елементів становить 3.9×10^{-3}, де $\mathbf{k}^{e,2}$ і $\mathbf{k}^{e,\text{NN}}$ – матриці жорсткості елемента, обчислені з використанням 2 точок квадратури Гаусса–Лежанда для інтегрування та з використанням кількості точок інтегрування, обчисленої нейронною мережею. Але з Таблиці 1 бачимо, що використовуючи підібрані точки і ваги точність переміщень погіршилася на 0.1% від значень при двох точках Гаусса–Лежандра в порівнянні з 30. Для біквадратичного елемента з найкращим значенням функції придатності (в цього елемента середні точки ребер знаходяться близько одна до одної) значення переміщень в деяких точках є більш точними, а в деяких менш точними (коливання від -1.3×10^{-5} до 3.6×10^{-5}, у відносних значеннях похибка досягала 137%). Тобто обчислення матриці жорсткості більш точно з підібраними точками і вагами квадратури не гарантує більш точного обчислення переміщення. Необхідно орієнтуватися на похибку розв'язку (зокрема в енергетичній нормі).

Оскільки покращення точності інтегрування через підбір ваг і точок є ресурсозатратним на стадії генерування набору даних для тренування і приріст точності не є значним, а може і погіршитися, то краще використовувати підбір кількості точок інтегрування, що є швидким на етапі генерування набору даних, надійнішим, щодо очікуваного приросту точності обчислення переміщень, але менш швидким на етапі чисельного інтегрування, оскільки буде братися більша кількість точок для обчислення інтегралу.

6. Висновки

У роботі досліджено вплив точності інтегрування за допомогою квадратурної формули на кінцевий результат МСЕ для осесиметричних задач із межами ізопараметричних чотирикутників зі значною кривиною та білінійною/біквадратичною апроксимаціями. Порівняно два підходи: прогноз мінімально необхідної кількості точок Гаусса–Лежандра та підбір самих вузлів і ваг квадратури. Обчислювальні експерименти на чверті кільця та різних прямокутних сітках показали, що навіть коли локальні матриці жорсткості наближаються до еталону в сенсі матричних норм, це не завжди приводить до помітно точніших переміщень. Найбільші покращення від підбору вузлів/ваг виявились малими або коливалися від покращення до погіршення від точки до точки в елементі та в багатьох випадках значення переміщень навіть погіршилися відносно стандартної 2×2 квадратури.

Натомість класифікація потрібної кількості точок (без зміни самих вузлів і ваг) дала стабільний і передбачуваніший ефект: за рахунок збільшення порядку квадратури там, де це справді потрібно геометрії, вдавалось систематично зменшувати похибку переміщень. Для розв'язаних задач на 0.5% зменшилася похибка обчислення переміщень. Такий підхід є менш ресурсозатратним на етапі побудови набору даних, надійніший з погляду збіжності, хоча й потенційно повільніший під час чисельного інтегрування.

Підбір вузлів і ваг підвищує точність інтегрування локальних матриць, але не завжди приводить до стабільного покращення точності переміщень, що підкреслює різницю між точністю інтеграла та точністю кінцевого розв'язку. Для покращення точності розв'язків, варто першочергово застосовувати модель, що підбирає кількість точок.

Інтеграція розробленого підходу з адаптивним згущенням сітки на основі використання оцінювачів похибки [18] зробить методику придатною для великих інжене-

рних моделей.

Список використаної літератури

1. Oishi A. Computational mechanics enhanced by deep learning / A. Oishi, G. Yagawa // Computer Methods in Applied Mechanics and Engineering. – 2017. – Vol. 327. – P. 327–351. – doi: 10.1016/j.cma.2017.08.040.
2. Yu M. Learned Gaussian quadrature for enriched solid finite elements / M. Yu, S. Kim, G. Noh // Computer Methods in Applied Mechanics and Engineering. – 2023. – Vol. 414. – Art. 116188. – doi: 10.1016/j.cma.2023.116188.
3. Yagawa G. Computational Mechanics with Deep Learning: An Introduction / G. Yagawa, A. Oishi // Cham, Switzerland: Springer. – 2022. – xiv, 402 p. – ISBN 978-3-031-11846-3. – doi: 10.1007/978-3-031-11847-0.
4. Daviet G. Neurally Integrated Finite Elements for Differentiable Elasticity on Evolving Domains / G. Daviet, T. Shen, N. Sharp, D.I.W. Levin // ACM Transactions on Graphics. – 2025. – Vol. 44, № 2. – Art. 20. – P. 1–17. – doi: 10.1145/3727874.
5. Teijeiro T. Machine Learning Discovery of Optimal Quadrature Rules for Isogeometric Analysis / T. Teijeiro, J.M. Taylor, A. Hashemian, D. Pardo // Computer Methods in Applied Mechanics and Engineering. – 2023. – Vol. 416. – Art. 116310. – doi: 10.1016/j.cma.2023.116310.
6. Vithalbhai S.K. Artificial neural network assisted numerical quadrature in finite element analysis in mechanics / S.K. Vithalbhai, D. Nath, V. Agrawal, S.S. Gautam // Materials Today: Proceedings. – 2022. – Vol. 66, Part 4. – P. 1645–1650. – doi: 10.1016/j.matpr.2022.05.254.
7. Gautam S.S. Transfer learning enhanced deep neural network application in Gauss quadrature for isogeometric analysis / S.S. Gautam, D. Nath, D. Neog // Engineering Applications of Artificial Intelligence. – 2025. – Vol. 145, № 2. – Art. 110182. – doi: 10.1016/j.engappai.2025.110182.
8. Nath D. Igrnet: A Robust Graph Neural Network Framework for Classifying Nurbs-Based Elements in Isogeometric Analysis with Application to Contact Mechanics / D. Nath, S.K. Das, D.R. Neog, S.S. Gautam. – Preprint, 2025. – 68 p.
9. Hughes T.J.R. Efficient quadrature for NURBS-based isogeometric analysis / T.J.R. Hughes, A. Reali, G. Sangalli // Computer Methods in Applied Mechanics and Engineering. – 2010. – Vol. 199, № 5–8. – P. 301–313. – doi: 10.1016/j.cma.2008.12.004.
10. Zienkiewicz O.C. The Finite Element Method: Its Basis and Fundamentals / O.C. Zienkiewicz, R.L. Taylor, J.Z. Zhu. – 7th ed. – Oxford, UK: Butterworth-Heinemann (Elsevier), 2013. – 756 p. – ISBN 978-1-85617-633-0.
11. Kress R. Numerical Analysis / R. Kress. – New-York: Springer, 1998. – xii, 326 p. – ISBN 978-0-387-98408-7. – doi: 10.1007/978-1-4612-0599-9.
12. Kennedy J. Particle Swarm Optimization / J. Kennedy, R.C. Eberhart // Proc. IEEE Int. Conf. Neural Networks. – Perth, Australia. – Nov. 1995. – 1995. – Vol. 4. – P. 1942–1948.
13. Poli R. Particle swarm optimization / R. Poli, J. Kennedy, T. Blackwell // Swarm Intelligence. – 2007. – Vol. 1, № 1. – P. 33–57. – doi: 10.1007/s11721-007-0002-0.
14. Clerc M. The particle swarm explosion, stability, and convergence in a multidimensional complex space / M. Clerc, J. Kennedy // IEEE Transactions on Evolutionary Computation. – 2002. – Vol. 6, № 1. – P. 58–73.
15. Eberhart R.C. Comparing inertia weights and constriction factors in particle swarm optimization / R.C. Eberhart, Y. Shi // Proc. IEEE Conference on Evolutionary Computation (ICEC). – 2000. – Vol. 1. – P. 84–88.
16. Burkov A. The Hundred-Page Machine Learning Book / A. Burkov. – Quebec City, Canada: Andriy Burkov. – 2019. – 160 p. – ISBN 978-1-9995795-0-0.

17. Kingma D.P. Adam: A method for stochastic optimization / D.P. Kingma, J. Ba // Proc. 3rd Intl. Conf. on Learning Representations (ICLR 2015). – San Diego, CA, USA. – May 7–9, 2015. – 2015. – doi: 10.48550/arXiv.1412.6980.
18. Dyyak I. Investigation of stress error estimator in elasticity problems / I. Dyyak, Yu. Yashchuk // Contemporary Problems of Mathematics, Mechanics and Computing Sciences / N.N. Kizilova, G.N. Zholtkevych (eds.). – Kharkiv, Ukraine: Publishing House PPB Virovec' A.P., 2011. – 396 p. – P. 33–43.

Стаття: надійшла до редколегії 02.09.2025
доопрацьована 24.09.2025
прийнята до друку 15.10.2025

ENCHANCING THE ACCURACY OF NUMERICAL INTEGRATION IN THE FEM USING NEURAL NETWORKS

D. Alianakh, I. Dyyak

Ivan Franko National University of Lviv,
1, Universytetska str., 79000, Lviv, Ukraine,
e-mail: hrefdmytro.alianakh@lnu.edu.ua, ivan.dyyak@lnu.edu.ua

This work employs machine learning methods to select the optimal order of integrating the entries of the finite element method stiffness matrix for axisymmetric elasticity problems using quadrilateral serendipity elements. We consider two approaches to improving the accuracy of integration of local stiffness matrices: a classifier that predicts the minimum number of Gauss–Legendre points required to achieve a prescribed accuracy relative to reference integration, and a regressor that estimates optimal quadrature nodes and weights. The training datasets were generated synthetically, with normalization of element geometry and constraints on angles and coordinates. Computational experiments showed that bringing local stiffness matrices closer to the reference in matrix norms does not guarantee a noticeable reduction in displacement error. In most experiments, the improvements from tuning quadrature nodes and weights did not exceed $10^{-3}\%$. Selecting the number of points yields benefits by increasing the quadrature order in cases where the geometric nonlinearity of the mapping requires it.

Key words: partial differential equations, axisymmetric problem of elasticity, finite element method, Gauss–Legendre quadrature, quadrilateral serendipity elements, neural networks, particle swarm optimization.