# РОЗДІЛ ІІІ. КОМП'ЮТЕРНІ НАУКИ

УДК 624.012.25: 539.386
DOI https://doi.org/10.26661/2786-6254-2022-2-08

## АЛГОРИТМ ПАРАЛЕЛЬНИХ ОБЧИСЛЕНЬ У МЕТОДІ СКІНЧЕННИХ ЕЛЕМЕНТІВ

**Гоменюк С. І.**
*доктор технічних наук, професор,*
*професор кафедри програмної інженерії*
*Запорізький національний університет*
*вул. Жуковського, 66, Запоріжжя, Україна*
*orcid.org/0000-0001-7340-5947*
*mf@znu.edu.ua*

**Козуб В. Ю.**
*асистент кафедри фізико-технічних систем та інформатики*
*Луганський національний університет імені Тараса Шевченка*
*вул. Коваля, 3, Полтава, Україна*
*orcid.org/0000-0003-2710-7206*
*v.y.kozub@gmail.com*

***Ключові слова:*** *метод скінченних елементів, матриця жорсткості, паралельні обчислення, OpenMP, напружено-деформований стан.*

Розв'язання прикладних задач фізики та механіки конструкцій потребує використання методу скінченних елементів з великою розмірністю розрахункових сіток. Крім того, в таких схемах одночасно використовуються різні типи скінченних елементів. Формування матриць жорсткості в такому випадку потребує значних обчислень за однаковою схемою для кожного скінченного елементу. У традиційному підході ці обчислення виконуються послідовно. Для розрахункових схем досить невеликого розміру, час розв'язування є незначним. У випадку великих розмірів сіток кількість та час розрахунків збільшуються, що потребує оптимізації обчислень з використанням алгоритмів паралельних обчислень. Для паралелізації методу скінченних елементів реалізовано управління роботою досить великої кількості процесів, організовано обмін даними між процесами. Під час виконання обрахунків виникає необхідність очікування моменту, коли завершиться виконання деякого етапу всіма процесорами. Час виконання програми визначається найповільнішою задачею з тих, що паралельно виконуються на різних процесорах. Для досягнення збалансування завантаження процесорів реалізовано рівномірний розподіл однаково вимірних задач. У системі із загальною пам'яттю, в яких обмін інформацією між процесорами відбувається за допомогою змінних, що зберігаються у загальній пам'яті, для забезпечення детермінованості виконання програми реалізовано поступове розпаралелювання програми. Для оптимізації обчислювальних процесів скінченних елементів обчислювального комплексу «МІРЕЛА+» розроблено алгоритми паралельного програмування для побудови матриць жорсткості скінченних елементів та розрахунку напружено-деформованого стану. Паралельна обробка даних реалізована за допомогою бібліотеки OpenMP, що забезпечує більшу швидкість розробки через зручність використання. На основі обчислювальних експериментів встановлено перевагу використання алгоритмів паралельних обчислень на початку розв'язування задач перед традиційними скінченно-елементними алгоритмами для великорозмірних розрахункових сіток.

# ALGORITHM OF PARALLEL CALCULATIONS IN THE FINITE ELEMENT METHOD

**Homeniuk S. I.**
*Doctor of Technical Sciences, Professor,*
*Professor at the Department of Software Engineering*
*Zaporizhzhia National University*
*Zhukovskoho str., 66, Zaporizhzhia, Ukraine*
*orcid.org/0000-0001-7340-5947*
*mf@znu.edu.ua*

**Kozub V. Yu.**
*Assistant at the Department of Physical and Technical Systems and Informatics*
*Luhansk Taras Shevchenko National University*
*Koval str., 3, Poltava, Ukraine*
*orcid.org/0000-0003-2710-7206*
*v.y.kozub@gmail.com*

***Key words:*** *finite element method, stiffness matrix, parallel computing, OpenMP, stress-strain state.*

The solution of applied problems of physics and mechanics of structures requires the use of the finite element method with a large dimension of calculation grids. In addition, different types of finite elements are used simultaneously in such schemes. The formation of stiffness matrices in this case requires significant calculations using the same scheme for each finite element. In the traditional approach, these calculations are performed sequentially. For computational schemes of sufficiently small size, the solution time is negligible. In the case of large grid sizes, the number and time of calculations increase, which requires optimization of calculations using parallel calculation algorithms. For the parallelization of the finite element method, work management of a sufficiently large number of processes is implemented, and data exchange between processes is organized. When performing calculations, there is a need to wait for the moment when the execution of a certain stage by all processors is completed. The execution time of the program is determined by the slowest task among those executed in parallel on different processors. In order to achieve load balancing of the processors, a separation of equally sized tasks is implemented. In a system with shared memory, in which the exchange of information between processors takes place with the help of variables stored in the shared memory, gradual parallelization of the program is implemented to ensure determinism of program execution. In order to optimize the computing processes of the finite elements of the "MIRELA +" computing complex, parallel programming algorithms have been developed to construct the stiffness matrices of the finite elements and calculate the stress-strain state. Parallel data processing is implemented using the OpenMP library, which provides faster development due to ease of use. On the basis of computational experiments, the advantage of using parallel computing algorithms at the beginning of problem solving over traditional finite element algorithms for large-scale valculation grids has been established.

**Вступ.** Одним з найбільш поширених методів розрахунку конструкцій є метод скінченних елементів (МСЕ). Залежно від виду задачі у разі скінченно-елементного моделювання розрахункові сітки сягають великих розмірів, зокрема, для композитних багатошарових конструкцій, конструкцій з концентраторами напружень, складених конструкцій. В такому випадку обчислення матриці жорсткості становить значну частину загального часу розв'язування задачі.

Необхідність вирішення складних тривимірних задач з мінімальними витратами призводить до потреби оптимізації обчислювальних процесів.

Використання кількох обчислювальних одиниць для вирішення обчислювальної задачі одночасно значно скорочує час розв'язку. За умови,

що обчислювальна задача може бути розбита на окремі, автономні одна від одної частини, які можна обробити одночасно, тим самим забезпечивши розв'язання задач за менший час за допомогою кількох обчислювальних одиниць, ніж за допомогою однієї обчислювальної одиниці. Обчислювальними одиницями можуть бути: один комп'ютер із кількома процесорами/ядрами, наприклад ноутбуки з двоядерними або чотириядерними процесорами; або суперкомп'ютер, що створюється з великої кількості таких комп'ютерів (або вузлів), з'єднаних мережею, що створюють кластер.

Підходи до паралелізації в МСЕ можна поділити на дві категорії – підходи з розподіленою пам'яттю та зі спільною пам'яттю [1]. У роботі з першою категорією зазвичай використовується високопродуктивна модель паралельного програмування Message Passing Interface (MPI). MPI – це легко масштабований засіб паралельного програмування, але за умови виконання завдання розподілу робочого навантаження, що зазвичай призводить до масштабних змін програми [2–4].

Паралелізація у другій категорії здебільшого здійснюється за допомогою спеціальних директив компілятора. Стандарт OpenMP [5] був розроблений для забезпечення базових конструкцій паралелізму у програмах на Fortran та C/C++. Використання OpenMP дає обмежений контроль над потоками порівняно з більш фундаментальним стандартом Pthreads [3]. Однак OpenMP забезпечує більшу швидкість розробки через зручність використання.

Можливість швидкого створення ефективних паралельних програм є серйозним аргументом у виборі засобів програмування. Технологія спочатку спроєктована таким чином, щоб користувач міг працювати з єдиним текстом для паралельної і послідовної програм. Звичайний компілятор на послідовній машині директиви OpenMP просто «не помічає», оскільки вони розташовані в коментарях (за винятком змінних оточення і спеціальних функцій). Додатковою особливістю OpenMP є можливість поступового, «інкрементального» розпаралелювання програми. Взявши за основу послідовний код, дається можливість крок за кроком додавати директиви, що описують паралельні конструкції. За такого підходу немає необхідності відразу писати паралельну програму цілком – її розробка ведеться послідовно. Це спрощує як процес програмування, так і налагодження програми.

Варто зауважити, що програми на MPI можуть не менш ефективно, ніж на OpenMP, виконуватися і на системах із загальною пам'яттю. З цієї точки зору технологія MPI є більш універсальною [6]. Однак програмувати на OpenMP, по-перше, значно зручніше, ніж з використанням технології MPI, а по-друге, технологія OpenMP значно економніше витрачає оперативну пам'ять на системах із загальною пам'яттю. Для завдань, що вимагають великого обсягу оперативної пам'яті, часто виявляється, що неможливо запустити таку кількість процесів на обчислювальній системі, щоб завантажити роботою всі обчислювальні ядра [7].

Опубліковані підходи до паралелізації МСЕ стосуються процедур розв'язання систем рівнянь. Проте майже відсутні публікації, присвячені застосуванню паралельних обчислень у формуванні матриць розрахункових рівнянь.

Метою роботи є розробка алгоритму паралелізації обчислень матриць жорсткості скінченних елементів для оптимізації розрахунків конструкцій.

**Огляд методів паралельних обчислень.** Використання паралельних програм потребує забезпечення керування роботою досить великої кількості процесів, організації обміну даними між процесами.

При цьому необхідно враховувати такі негативні особливості:

– через асинхронність доступу до даних може втрачатися детермінізм поведінки програми;

– можливість виникнення тупикових ситуацій;

– виникнення проблеми масштабованості програми і балансування завантаження обчислювальних вузлів.

У разі використання багатопроцесорних систем очікується, що швидкість виконання обчислень збільшиться в $n$ разів, якщо використовується n процесорів замість одного. Насправді прискорення виявляється значно меншим очікуваного. Зменшення зростання продуктивності у паралельних обчисленнях зумовлене двома факторами: перший пов'язаний з властивостями алгоритму програми [8], а другий – з технічними властивостями обчислювальної системи.

Суть проблеми полягає в тому, що кожна паралельна програма містить якусь частину непаралельного коду. На системах із загальною пам'яттю це та частина коду, яка виконується тільки головним потоком, а на системах з розподіленою пам'яттю це частина коду, яка виконується всіма процесорами. Припустимо, що відношення часу виконання непаралельної частини програми до всього часу виконання дорівнює $f$ ($0 < f < 1$). Якщо час виконання на одному процесорі дорівнює $t_s$, то час виконання паралельної частини – $(1-f)t_s$, а час виконання послідовної частини – $ft_s$. В ідеальному випадку у разі використання $n$ процесорів час виконання паралельної частини становитиме $(1- ft_s)/n$, а час виконання послідовної частини залишиться $ft_s$.

Таким чином, прискорення виконання програми на n процесорах становитиме:

$$S(n) = \frac{t_s}{ft_s + (1-f)t_s / n} = \frac{n}{1+(n-1)f}. \qquad (1)$$

Вираз (1) є формальним визначенням закону Амдала. При *n*, що прямує до нескінченності, прискорення становитиме 1/*f*. З цього закону випливає, що якщо частка непаралельності коду в програмі становить 5%, то неможливо отримати більш ніж 20-кратне прискорення програми, скільки б процесорів не використовувалося.

У разі розбиття великої задачі на більш дрібні підзадачі, які повинні виконуватися паралельно на різних процесорах, часто виникає ситуація, коли потрібно чекати завершення виконання деякого етапу всіма процесорами. У цьому випадку повний час виконання буде визначатися найповільнішою підзадачею. Тому у разі розробки паралельної програми надзвичайно важливо максимально рівномірно розподіляти обчислювальну роботу між процесорами. Досягнення збалансування завантаження є нетривіальним завданням.

У системах із загальною пам'яттю, в яких обмін інформацією між процесорами відбувається за допомогою змінних, що зберігаються у загальній пам'яті, синхронізація процесів особливо важлива. Якщо кілька процесів одночасно намагаються модифікувати одну і ту саму змінну, то необхідний спеціальний механізм синхронізації виконання процесів для забезпечення детермінованості виконання програми.

Ефективність паралельних програм на системах з розподіленою пам'яттю істотно залежить від комунікаційного середовища. Комунікаційне середовище характеризується двома параметрами: пропускною здатністю (bandwidth), яка визначає кількість байт, що передається за одиницю часу, і латентністю (latency), яка визначає час, що витрачається на підготовку до передачі повідомлення. Для великої кількості задач, в яких обміни даними нечасті і невеликі за обсягом, ці параметри є цілком задовільними, однак у випадках, коли в програмі передається багато дрібних повідомлень, такі параметри стають неприйнятними, і масштабованість програми виявляється надзвичайно низькою. У будь-якому випадку комунікаційні операції виконуються значно повільніше, ніж звернення до локальної пам'яті, тому найбільш ефективними будуть ті паралельні програми, в яких обміни зведені до мінімуму.

**Основні співвідношення методу.** Для дослідження ефективності паралелізації обчислень компонентів матриці жорсткості вибрано пакет прикладних програм «MIRELA+», призначений для розв'язування задач механіки деформованого твердого тіла. Для усунення ефекту «хибного зсуву» в комплексі використовується моментна схема скінченних елементів, сутність якої полягає в апроксимації полів переміщень та деформацій у вигляді розкладання в ряд по ступеневих функціях особливого виду, що дозволяє встановити досить простий зв'язок між компонентами розкладань та уникнути зайвих компонент розкладання деформацій

$$u_i = \sum_{p,q,r=0}^{m,n,l} w_i^{(pqr)} \psi^{(pqr)} = \sum_{p,q,r=0}^{m,n,l} w_i^{(pqr)} \frac{(x_1)^p}{p!} \frac{(x_2)^q}{q!} \frac{(x_3)^r}{r!},$$

$$\varepsilon_{ij} = \sum_{s,t,g} e_{ij}^{(stg)} \psi^{(stg)},$$

$$\varepsilon_{ij} = \mathbf{F}_{ij}^{s'} w_{s'} = \mathbf{F}_{ij}^{s'} \mathbf{A} \mathbf{u}_{s'}, \qquad (2)$$

де **A** – матриця перетворення, що встановлює зв'язок між функціями форми скінченного елемента та ступеневими базисними функціями.

У разі побудови спрощених моделей гіперпружних тіл використовуються гіпотеза нестисливості [9]. В комплексі «MIRELA+» реалізований підхід, що дозволяє уникнути спрощення. Функція змінення об'єму розкладається по ступеневих функціях і дозволяє врахувати слабку стисливість матеріалів (типу еластомерів) [10]

$$\theta_i = \sum_{\alpha,\beta,\gamma=0}^{m-1,n-1,l-1} \xi^{(\alpha\beta\gamma)} \psi^{(\alpha\beta\gamma)},$$

$$\xi^{(\alpha\beta\gamma)} = e_{ii}^{(\alpha\beta\gamma)} g^{ii}$$

$$\theta = \mathbf{F}_{\theta}^{s'} w_{s'} = \mathbf{F}_{ij}^{s'} \mathbf{A} \mathbf{u}_{s'} \qquad (3)$$

Для задач статики конструкцій варіаційне рівняння рівноваги лінійно пружного тіла має вигляд

$$\delta \iiint_v \left( \mu g^{kj} g^{li} \varepsilon_{kl} \varepsilon_{ij} + \frac{\lambda}{2} \theta^2 \right) dv = \iiint_v \rho P^i \delta u_i dv + \iint_s F^i \delta u_i ds, \qquad (4)$$

де μ, λ, ρ – механічні характеристики матеріалу, $P^i$, $F^i$ – об'ємні та поверхневі сили.

З огляду на використовувані методи апроксимації варіація енергії деформування має вигляд

$$\delta W = \iiint_v \delta\{e_{ij}\}^T \{\psi\} \mu g^{ik} g^{jl} \{e_{kl}\}^T \{\psi\} dv + \iiint_v \{\xi\}^T \{\psi\} \lambda \delta\{\xi_{kl}\}^T \{\psi\} dv$$

$$\mathbf{H}^{ijkl} = \int_{-1}^{1} \int_{-1}^{1} \int_{-1}^{1} 2\mu g^{ik} g^{jl} \{\psi_{ij}\} \{\psi_{kl}\}^T \sqrt{g} dx_1 dx_2 dx_3$$

$$\mathbf{H}^{\theta} = \int_{-1}^{1} \int_{-1}^{1} \int_{-1}^{1} \lambda \{\psi_{\theta}\} \{\psi_{\theta}\}^T \sqrt{g} dx_1 dx_2 dx_3$$

$$\delta W = \delta \mathbf{u}_{s'}^T \mathbf{K}^{s't'} \mathbf{u}_t + \delta \mathbf{u}_{s'}^T \mathbf{K}_{\theta}^{s't'} \mathbf{u}_t, \qquad (5)$$

де $\mathbf{K}^{s't'} = \mathbf{A}^T \mathbf{F}^T \mathbf{H} \mathbf{F} \mathbf{A}$,

$\mathbf{K}_{\theta}^{s't'} = \mathbf{A}^T \mathbf{F}_{\theta}^T \mathbf{H}^{\theta} \mathbf{F}_{\theta} \mathbf{A}$ – матриці жорсткості.

Процедура формування матриці жорсткості скінченного елементу містить такі кроки:

– обчислення коефіцієнтів розкладання переміщень та коефіцієнтів матриці А;

– чисельне інтегрування за схемою Гауса.

Для точок інтегрування виконуються такі процедури:

– обчислення матриць переходу від глобальної системи координат до локальної системи координат та матриці зворотного перетворення в точках інтегрування;

– обчислення часткових похідних для формування матриці F, що пов'язує переміщення та деформації у скінченному елементі.

**Результати досліджень.** Реалізація паралельної обробки в рамках дослідження виконана за допомогою бібліотеки OpenMP.

Спільна обробка великих масивів даних вимагає автономності операцій над ними для кожного циклу формування матриці жорсткості й обчислення параметрів напружено-деформованого стану.

Паралелізм досягається за рахунок розбиття ітерацій циклу обчислень на блоки, які рівномірно і паралельно розподіляються по потоках виконання. В рамках OpenMP для реалізації подібної схеми виконання застосовується директива DO перед циклом. Опція SCHEDULE(STATIC) задає блочно-циклічний розподіл ітерацій блоками розміром, що визначається діленням числа ітерацій на число потоків

```
!$OMP PARALLEL PRIVATE(S, SN)
!$OMP DO SCHEDULE(STATIC) ORDERED
DO 1 I=1,N
<Parallel loop code>
!$OMP ORDERED
WRITE (*) RESULT
!$OMP END ORDERED
1 CONTINUE
!$OMP END DO
!$OMP END PARALLEL
```

Директива PRIVATE дозволяє визначити такі змінні (масиви), які повинні бути локальними для кожного потоку. У разі входу в паралельну область для кожного потоку створюється окремий екземпляр, який не має ніякого зв'язку з оригінальною змінною поза паралельною областю. За замовчуванням компілятор визначає змінні, які були проініціалізовані до паралельної секції як спільні (SHARED) для всіх потоків виконання.

Перед завершенням циклу виконується процедура виводу результатів розрахунку скінченного елементу, що повинна виконатися в тому самому порядку, як і в послідовній версії циклу. Поведінку послідовного доступу за вихідним порядком забезпечує директива ORDERED.

Послідовні області OpenMP програми виконуються тільки основним потоком (рис. 1). Для підтримки паралелізму використовується схема Fork-Join. У разі входу в паралельну область (!$OMP PARALLEL) потік-майстер породжує додаткові потоки (виконується операція Fork). Після породження кожен потік отримує свій унікальний номер. Усі породжені потоки виконують один і той же код, що відповідає паралельній області. У разі виходу (!$OMP END PARALLEL) з паралельної області основний потік чекає завершення інших потоків, і подальше виконання програми продовжує тільки master (виконується операція Join).

Як приклад для перевірки та тестування представлено розрахунок квадратної плити затиснутої по контуру під дією згинального навантаження (рис. 2). Для розрахунку використано обчислювальний комплекс «МІРЕЛА+» із застосуванням моментної схеми скінченних елементів. Тестування виконувалось на пристрої, оснащеним процесором Intel i7-9750H (6 ядер та 12 потоків) та 16 ГБ RAM.

**Висновки**

З появою багатоядерних процесорів, системи із загальною пам'яттю стали самим масовими видом багатопроцесорних систем, а OpenMP, своєю чергою, надає один з найзручніших програмних інтерфейсів розробки для таких систем. Версія OpenMP програми також може бути скомпільована з використанням параметру компілятора «non-parallel» (директиви OpenMP будуть ігноруватися компілятором), що забезпечує переносимість коду на різні платформи. З ростом популярності комп'ютерів SMP зростає важливість наявності ефективних і кросплатформних паралельних кодів.

У роботі проведено аналіз наявних технічних засобів паралелізації, переваг і недоліків впро-

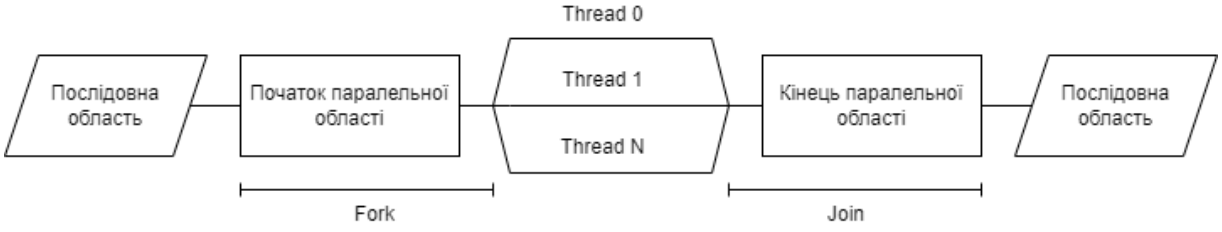


**Рис. 1. Процес паралельного виконання програми**

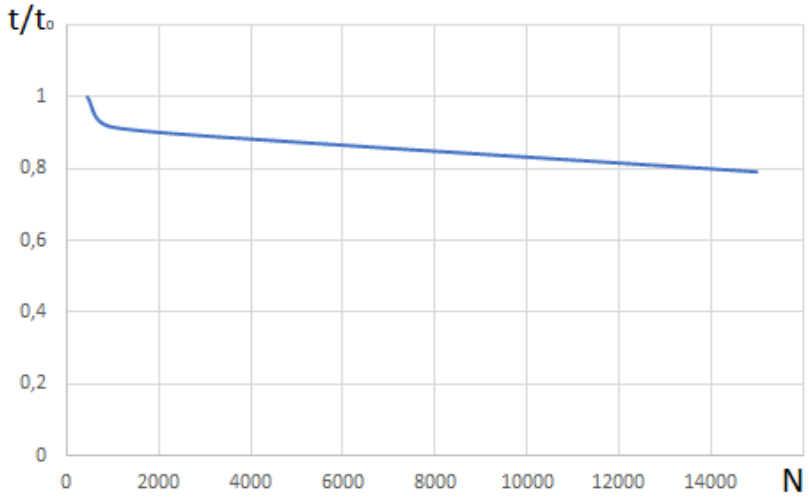


**Рис. 2. Порівняння швидкості розв'язку для сіток з різною кількістю скінченних елементів**

вадження паралельних рішень, використовуючи директиви OpenMP розроблено паралельну версію процедури обчислення для виконання цільових розрахунків на базі обчислювального комплексу «МІРЕЛА+». Тести продуктивності на представленому прикладі продемонстрували задовільне прискорення за допомогою паралельних секцій.

## ЛІТЕРАТУРА

1. Jarzebski P., Wisniewski K., Taylor R.L. On parallelization of the loop over elements in FEAP. *Computional Mechanics*. 2015. No. 56. Pp. 77–86.
2. Wozniak M., Bukowska A. Comparison of multi-frontal and alternating direction parallel hybrid memory iGR direct solver for non-stationary simulations. *Computer Science*. 2020. No. 21(4). Pp. 419–439.
3. Yamaguchi T., Kawase Y., Nagase A., Ishimura S. Performance Evaluation of 3-D Hybrid Parallel Finite Element Method by MPI/OpenMP *J. Japan Society of Applied Electromagnetics and Mechanics*. 2019. Vol. 27. No. 1. Pp. 85–90.
4. Pantalé O. Parallelization of an object-oriented FEM dynamics code: influence of the strategies on the Speedup. *Adv. Eng. Softw*, 2005. 36. Pp. 361–373.
5. OpenMP Application Programming. Interface. URL: https://www.openmp.org/ wp-content/uploads/ OpenMP-API-Specification-5-1.pdf (дата звернення: 26.11.2021).
6. Bozhanskii M., Patziak B. Parallelization of assembly operation in finite element method. *Acta Polytechnica,* 2020. No. 60(1), 25–37.
7. Amorim L., Goveia T., Mesquita R., Baratta I. GPU Finite Element Method Computation Strategy Without Mesh Coloring. *J. Microwaves, Optoelectronics and Electromagnetic Applications*, 2020. Vol. 19. No. 2. Pp. 252–264.
8. Amdahl G.M. Validity of the single-processor approach to achieving large scale computing capabilities *Proc. AFIPS Conference Proceedings,* 1967. Pp. 483–485.
9. Suchocki C., Jemiolo S. On Finite Element Implementation of Polyconvex Incompressible Hyperelasticity: Theory, Coding and Applications. *International Journal of Computational Methods*, 2020. Vol. 17. No 8. 1950049. URL: https://www.researchgate.net/publication/332969982.
10. Bazhenov V.A., Kozub Yu.G., Solodei I.I. Thermoelasticity of elastomeric constructions with initial stresses. *Strength of Materials and Theory of Structures,* 2020. No. 104. Pp. 299–308.

## REFERENCES

1. Jarzebski, P., Wisniewski, K., Taylor, R. (2015) On parallelization of the loop over elements in FEAP. *Computional Mechanics*. No. 56, pp.77–86.
2. Wozniak, M., Bukowska, A. (2020). Comparison of multi-frontal and alternating direction parallel hybrid memory iGRM direct solver for non-stationary simulations. *Computer Science*. No. 21(4), pp. 419–439.

3. Yamaguchi, T., Kawase, Y, Nagase, A., Ishimura, S. (2019). Performance Evaluation of 3-D Hybrid Parallel Finite Element Method by MPI/OpenMP. *Journal Japan Society of Applied Electromagnetics and Mechanics*. Vol. 27, No. 1, pp. 85–90.
4. Pantalé, O. (2005). Parallelization of an object-oriented FEM dynamics code: influence of the strategies on the Speedup. *Advances in Engineering Software*. No. 36(6), pp. 361–373.
5. OpenMP Application Programming. Interface. Retrieved from: https://www.openmp.org/ wp-content/ uploads/OpenMP-API-Specification-5-1.pdf (Last accessed: 26.11.2021).
6. Bozhanskii, M,, Patziak, B. (2020). Parallelization of assembly operation in finite element method. *Acta Polytechnica*. No. 60(1), pp. 25–37.
7. Amorim, L., Goveia, T., Mesquita, R., Baratta, I. (2020). GPU Finite Element Method Computation Strategy Without Mesh Coloring. *Journal Microwaves, Optoelectronics and Electromagnetic Applications*. Vol. 19. No. 2, pp. 252–264.
8. Amdahl, G.M. (1967). Validity of the single-processor approach to achieving large scale computing capabilities *AFIPS. Conference Proceedings.* 30, pp. 483–485.
9. Suchocki, C., Jemiolo, S. (2020). On Finite Element Implementation of Polyconvex Incompressible Hyperelasticity: Theory, Coding and Applications. *International Journal of Computational Methods*. Vol. 17, No. 8, 1950049. Retrieved from: https://www.researchgate.net/publication/332969982.
10. Bazhenov, V.A., Kozub, Yu.G., Solodei, I.I. (2020). Thermoelasticity of elastomeric constructions with initial stresses. *Strength of Materials and Theory of Structures*. No. 104, pp. 299–308.