УДК 004.4 : 004.92
DOI https://doi.org/10.26661/2786-6254-2025-1-12

ВЕБРЕАЛІЗАЦІЯ ПОСТПРОЦЕСОРА З ВІДКРИТИМ ПОЧАТКОВИМ КОДОМ

Гоменюк С. І.
доктор технічних наук, професор,
декан математичного факультету
Запорізький національний університет
вул. Університетська, 66, Запоріжжя, Україна
orcid.org/0000-0001-7340-5947
mf@znu.edu.ua

Гребенюк С. М.
доктор технічних наук, професор,
завідувач кафедри фундаментальної та прикладної математики
Запорізький національний університет
вул. Університетська, 66, Запоріжжя, Україна
orcid.org/0000-0002-5247-9004
gsm1212@ukr.net

Панасенко Є. В.
кандидат фізико-математичних наук, доцент,
доцент кафедри фундаментальної та прикладної математики
Запорізький національний університет
вул. Університетська, 66, Запоріжжя, Україна
orcid.org/0000-0002-9165-6149
panasenko.yevgeniy@gmail.com

Ключові слова: *метод скінченних елементів, WebGL, візуалізація, аксонометричні проєкції, тривимірні об'єкти, програмна реалізація, алгоритми проєктування, моделювання.*

У роботі розглядається один із найбільш перспективних напрямів підвищення ефективності проєктування сучасної техніки, а саме створення інструменту для проведення віртуального комп'ютерного експерименту замість натурних випробувань, завдяки чому істотно пришвидшується й здешевлюється процес розробки, а також забезпечується його більша безпечність.

Найчастіше на практиці для комп'ютерного моделювання широкого кола проблем використовується чисельний метод – метод скінченних елементів. Його застосування потребує наявності спеціалізованого програмного забезпечення – систем скінченно-елементного аналізу, які можуть бути умовно поділені на три основні підсистеми: 1) препроцесор (підготовка даних для розрахунку), 2) процесор (реалізація розрахунку) і 3) постпроцесор (аналіз результатів).

Практичний досвід показує, що стадія аналізу результатів (постобробка) за своєю складністю та тривалістю може значно перевищувати два попередні етапи чисельного моделювання. Саме тому від якості реалізації постпроцесора та зручності його використання користувачем напряму залежить ефективність автоматизованого проєктування в машинобудуванні та будівництві.

Переважна більшість сучасних постпроцесорів є невід'ємними складовими відповідних систем скінченно-елементного аналізу (наприклад, Ansys, MSC Nastran, COMSOL та ін.). Це фактично робить неможливим їх використання спільно зі сторонніми системами, орієнтованими на розв'язання певних вузьких класів задач, наприклад механіки композитів.

Крім того, останнім часом швидкими темпами розвиваються хмарні технології, що потребує створення відповідного програмного забезпечення, реалізація якого у вигляді вебзастосунків дасть можливість досягти реальної багатоплатформності, оскільки сучасні браузери фактично є апаратно незалежними віртуальними машинами, що працюють майже на всіх типах сучасної комп'ютерної техніки. З огляду на це розробка відокремлених програмних систем для автоматизації постаналізу результатів чисельних розрахунків, отриманих у різних системах чисельного моделювання, у вигляді вебзастосунків з відкритим початковим кодом на сьогодні є актуальною задачею.

WEB IMPLEMENTATION THE POSTPROCESSOR WITH AN OPEN-SOURCE CODE

Homeniuk S. I.
Doctor of Technical Sciences, Professor,
Dean of the Department of Mathematics
Universitetska str., 66, Zaporizhzhia, Ukraine
orcid.org/0000-0001-7340-5947
mf@znu.edu.ua

Grebenyuk S. M.
Doctor of Technical Sciences, Professor,
Head of the Department of Fundamental and Applied Mathematics
Zaporizhzhia National University
Universitetska str., 66, Zaporizhzhia, Ukraine
orcid.org/0000-0002-5247-9004
gsm1212@ukr.net

Panasenko Y. V.
Candidate of Physical and Mathematical Sciences, Associate Professor,
Associate Professor at the Department of Fundamental and Applied Mathematics
Zaporizhzhia National University
Universitetska str., 66, Zaporizhzhia, Ukraine
orcid.org/0000-0002-9165-6149
panasenko.yevgeniy@gmail.com

Key words: *finite element method, WebGL, visualization, axonometric projections, three-dimensional objects, software implementation, design algorithms, modeling.*

The paper considers one of the most promising areas for increasing the efficiency of designing modern equipment, namely, creating a tool for conducting a virtual computer experiment instead of full-scale tests, which significantly speeds up and reduces the cost of the development process, as well as ensuring its greater safety. Most often in practice, the numerical finite element method is used for computer modeling of a wide range of problems. Its application requires the availability of specialized software – finite element analysis systems, which can be conditionally divided into three main subsystems: 1) preprocessor (data preparation for calculation), 2) processor (calculation implementation) and 3) postprocessor (results analysis).
Practical experience shows that the stage of results analysis (postprocessing) in terms of its complexity and duration can significantly exceed the two previous stages of numerical modeling. That is why the efficiency of automated design in mechanical engineering and construction directly depends on the quality of the implementation of the postprocessor and the convenience of its use by the user.
The vast majority of modern postprocessors are integral components of the corresponding finite element analysis systems (for example, Ansys, MSC Nastran, COMSOL, etc.). This actually makes it impossible to use them together with third-party systems focused on solving certain narrow classes of problems (for example, composite mechanics).

In addition, cloud technologies have been developing rapidly recently, which requires the creation of appropriate software, the implementation of which in the form of web applications will allow achieving real cross platform, since modern browsers are actually hardware-independent virtual machines that run on almost all types of modern computer technology. That is why the development of separate software systems for automating post-analysis of the results of numerical calculations obtained in various numerical modeling systems in the form of open-source web applications is currently an urgent task.

Вступ. Однією з важливих задач, які постають перед сучасним машинобудуванням, є заміна натурних випробувань дослідного зразка віртуальним комп'ютерним експериментом, за результатами якого здійснюється прогнозування поведінки об'єкта на основі виконаних чисельних розрахунків. Це дає змогу як істотно здешевити, так і пришвидшити процес проєктування та створення нової техніки.

Найбільш поширеним способом реалізації комп'ютерної симуляції є використання методу скінченних елементів (МСЕ) [1]. У загальному випадку його застосування потребує використання відповідного програмного забезпечення, яке може бути поділено на три окремі підсистеми:

1) препроцесора, що автоматизує підготовку вихідних для розрахунку даних (створення геометричної моделі об'єкта розрахунку, її дискретизація на скінченні елементи (СЕ), задання крайових умов тощо);

2) процесора, який безпосередньо виконує розрахунок задачі за допомогою МСЕ;

3) постпроцесора, що автоматизує аналіз отриманих на попередньому етапі чисельних результатів.

Досвід практичного використання МСЕ показує, що стадія аналізу результатів (постобробка) за тривалістю та трудомісткістю часто може перевищувати всі попередні етапи моделювання. Саме тому від якості реалізації постпроцесора залежить не тільки зручність роботи інженера, а і якість проєктування нової техніки.

У більшості випадків постпроцесори є невід'ємними складовими відповідних систем скінченно-елементного аналізу або FEA (Finite Element Analysis) [2], як-от, наприклад, Ansys [3], MSC Nastran [4], COMSOL [5] та ін. [6], що унеможливлює їх використання спільно зі сторонніми процесорами або форматами файлів. Також слід заначити, що стрімке поширення хмарних технологій робить актуальною задачу створення FEA-систем, які б могли працювати у вигляді вебзастосунків, що дало б змогу також досягти реальної кросплатформності, оскільки сучасні браузери фактично є апаратно-незалежними віртуальними машинами, які працюють на більшості типів сучасної комп'ютерної техніки.

Таким чином, розробка постпроцесорів з відкритим початковим кодом, які б могли бути легко адаптовані для аналізу результатів розрахунків, отриманих сторонніми FEA-системами, і при цьому були реалізовані у вигляді вебдодатків, на сьогодні є актуальною задачею.

Постановка задачі

Загальний алгоритм проєктування можна представити таким чином (рис. 1).

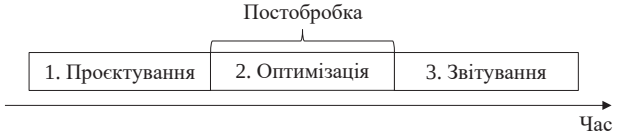

Рис. 1. Загальні стадії проєктування

Спочатку створюється цифрова модель об'єкта та виконується її попередній розрахунок. На наступному етапі здійснюється аналіз отриманих результатів та в разі потреби внесення відповідних коректив у вихідну модель (оптимізація). Основною задачею постобробки є перетворення високодеталізованих результатів розрахунків – великих масивів числових даних, отриманих із застосуванням МСЕ, у зручний для розуміння людиною формат. Отримана на цій стадії досліджень інформація (проєкції, графіки, діаграми, ізолінії тощо) використовується інженерами для перевірки валідності вихідної моделі, аналізу фізичної поведінки змодельованого об'єкта, а також на третьому етапі – під час створення відповідних звітів про підсумки досліджень.

Постобробка даних, отриманих з використанням МСЕ, потребує наявності спеціалізованого програмного забезпечення – постпроцесорів, використання яких можна вважати частиною інженерного принципу, заснованого на знаннях [7]. До найбільш відомих вбудованих постпроцесорів можна віднести Ansys Workbench [1, 3], Abaqus Viewer [9] та SolidWorks Simulation [10], які є невід'ємними складовими відповідних FEA-систем. Серед окремих програм можна виділити ParaView [11], TecPlot [12], MATLAB [13] та QuickField Postprocessor [14]. Крім того, для постобробки даних часто також використовують спеціалізовані бібліотеки, наприклад NumPy [15], Matplotlib [16], VTK [17] та ін. [19].

Незважаючи на велику кількість наявного програмного забезпечення, актуальність створення нових постпроцесорів зумовлена потребою в

підтримці нових спеціалізованих FEA-систем та ефективному використанні можливостей сучасних веб- та хмарних технологій.

Програмна реалізація постпроцесора

Найбільш поширеним способом постобробки результатів розрахунку є перетворення числових даних у графічні зображення, наочні для користувача. На сьогодні існує велика кількість способів візуалізації результатів розрахунку, наприклад, побудова дво- та тривимірних моделей, ізоліній, кольорових карт розподілу шуканої функції по поверхні об'єкта моделювання тощо.

Засоби побудови тривимірних сцен

Для створення тривимірних графічних сцен у вебзастосунках використовується сучасна технологія WebGL [19], яка працює спільно з вбудованою мовою програмування ECMAScript [20] у графічному елементі HTML5 Canvas. Для відображення за допомогою WebGL будь-якого об'єкта насамперед потрібно спеціалізованою мовою програмування GLSL [21] створити дві програми – вершинний і фрагментний шейдери, які вбудовуються у скрипт мовою JavaScript (яка є найбільш поширеним діалектом ECMAScript), що безпосередньо реалізують візуалізацію графічної сцени.

Вершинний шейдер фактично описує процедуру растерізації – перетворення фізичних координат об'єкта в координати екрана, в основі яких лежить побудова аксонометричних проєкцій. Задачею фрагментного (піксельного) шейдера є обчислення кольору кожного пікселя (наприклад, для врахування освітлення або інших ефектів).

Формула перетворення світових координат у координати відсікання (внутрішні координати відеоадаптера, які лежать в діапазоні від –1 до +1) може бути визначена таким чином:

$$v = \boldsymbol{P} \cdot s \cdot (p - c) - t,$$

де $\boldsymbol{P}$ – це матриця видового перетворення (вибирається заздалегідь залежно від типу вживаної проєкції);
p – вектор фізичних координат вершини;
c – вектор координат центру сцени;
s – вектор коефіцієнтів масштабування вздовж кожної осі координат;
t – вектор переносу (трансляції) сцени;
v – вектор екранних координат.

Відповідний вершинний шейдер для візуалізації дво- та тривимірних об'єктів розрахунку можна описати таким чином.

```
attribute vec4 a_position;
attribute vec3 a_normal;
attribute vec4 a_color;

uniform mat4 u_worldViewProjection;
uniform mat4 u_worldInverseTranspose;
uniform vec4 u_translation_center;
uniform vec4 u_translation;
uniform vec4 u_scale;

varying vec3 v_normal;
varying vec4 v_color;

void main() {
  gl_Position = u_worldViewProjection * (u_
scale * a_position –
                u_scale * u_translation_center)- u_
translation;

  // Orient the normal and pass to the fragment
shader
  v_normal = mat3(u_worldInverseTranspose) *
a_normal;
  v_color = a_color;
}
```

Тут атрибути a_position, a_normal та a_color визначають вихідні координати, колір і вектор нормалі (відповідно), що передаються в шейдер для кожної вершини, яка оброблюється.

Фрагментний шейдер, який обчислює інтенсивність освітлення пікселя залежно від його положення, можна описати так.

```
precision mediump float;
varying vec3 v_normal; // Passed in from the
vertex shader
varying vec4 v_color;

uniform vec3 u_reverseLightDirection;
uniform vec4 u_color;
uniform int u_is_mesh;

void main() {
    float light = dot(normalize(v_normal), u_
reverseLightDirection);
  if (u_is_mesh == 0)
    gl_FragColor = v_color;
  else
    gl_FragColor = u_color;
  gl_FragColor.rgb *= abs(light);
}
```

У цьому коді слід пояснити лише таке: на основі інформації про вектор нормалі, отриманої з вершинного шейдера, обчислюється інтенсивність освітлення вершини (light), а потім відповідне значення кольору вершини (gl_FragColor.rgb) множитиметься на модуль light для однакового зображення як зовнішньої, так і внутрішньої сторони поверхні. Також тут враховується необхідність використання іншого кольору для візуалізації границь СЕ (за ввімкнення відповідного режиму).

Алгоритм візуалізації розподілу функції по поверхні об'єкта

Для побудови кольорової карти розподілу певної функції по поверхні досліджуваного об'єкта достатньо реалізувати відповідний алгоритм для довільного трикутника, оскільки незалежно від типу вживаного СЕ будь-яку поверхню можна представити у вигляді скінченної множини трикутних межових елементів. Для візуалізації розподілу функції U по трикутному межовому елементу слід виконати його тріангуляцію на окремі трикутники, у межах яких значення функції, що досліджується, не змінюється (рис. 2).

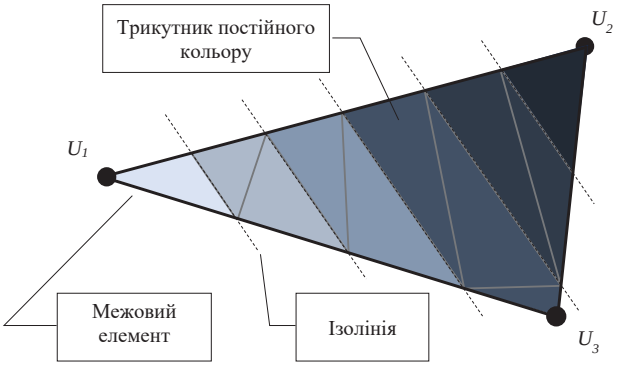

Рис. 2. Загальна схема візуалізації трикутника

З припущення, що ізолінії в межах окремого межового елемента є прямими, цей алгоритм можна описати таким чином:

1) за допомогою формули $c_j = \frac{U_j - U_{min}}{N(U_{max} - U_{min})}$ визначаються індекси кольорів у кожній вершині межового елемента (U_j – значення функції $в\ j-й$ вершині трикутника, U_{min}, U_{max} – відповідно мінімальне та максимальне значення функції U по всій області, N – кількість кольорів, що використовуються для побудови сцени);

2) якщо всі індекси рівні, то трикутник повністю замальовується кольором з відповідним індексом і алгоритм закінчується;

3) відповідно до кількості кольорових переходів між вершинами вихідного трикутника обчислюються координати відрізків, кінці яких лежать на перетинах ізоліній зі сторонами межового елемента;

4) з отриманих на попередньому кроці координат формуються й відображаються однокольорові трикутники (рис. 2), після чого алгоритм закінчує свою роботу [22].

Реалізація інтерфейсу користувача у вебсередовищі

Для реалізації графічного інтерфейсу користувача у вебзастосунку було використано JavaScript-бібліотеку з відкритим початковим кодом React [23], яка забезпечує високу швидкість роботи, є кросплатформною і такою, що легко масштабується. Крім того, за допомогою React можна створювати складні інтерфейси з маленьких ізольованих компонентів, що дає змогу легко розширювати функціональність вебзастосунку.

За допомогою React було розроблено класові компоненти, які реалізують такі елементи графічного інтерфейсу, як канва, смуги прокрутки, радіокнопки, чекбокси тощо, що використовуються для вибору потрібного користувачеві файлу даних і налаштування різних режимів відображення сцени. Загальний інтерфейс постпроцесору, який отримав назву Mesh, наведено на рис. 3.

Mesh може візуалізувати як дво- та тривимірні скінченно-елементні сітки (у форматах Gmsh [24], Netgen [25], QFEM [26]), так і результати розрахунків, отримані в системах скінченно-елементного аналізу QFEM і МІРЕЛА+ [27].

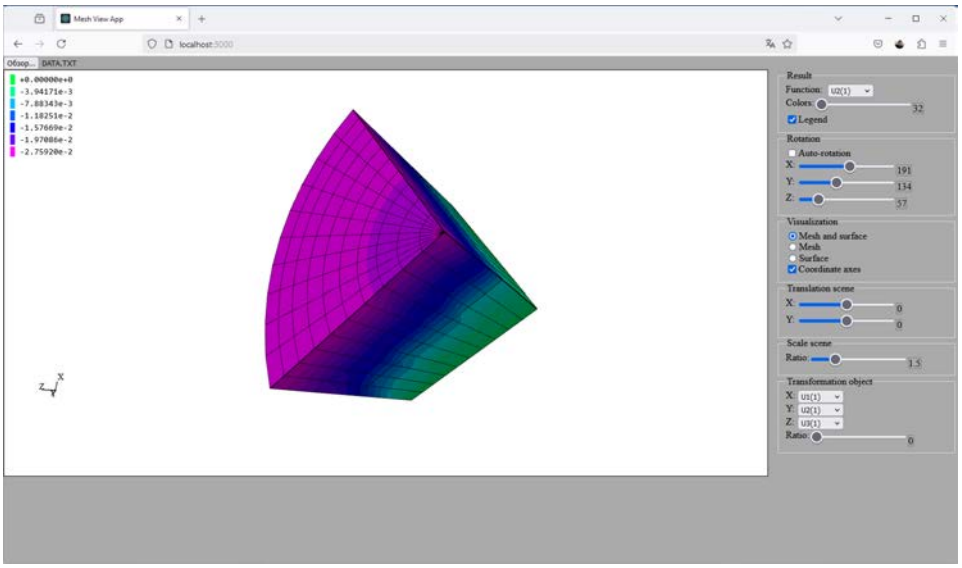

Рис. 3. Інтерфейс постпроцесора Mesh

Трансформація сцени

У Mesh користувач може вибирати для візуалізації потрібну йому функцію, а також різні режими її відображення. У поточній версії програми використовується спектральна шкала кодування. Мінімальні від'ємні значення позначаються фіолетовим кольором, нульові – зеленим, а максимальні додатні – червоним. У разі потреби користувач може змінювати кількість вживаних кольорів: від 32 до 256 (рис. 4), вмикати різні режими відображення сітки тощо.

Крім того, користувач може виконувати стандартні операції трансформації сцени: обертання, зсув і масштабування.

Для підвищення наочності користувач може також виконати трансформації безпосередньо об'єкта розрахунку, коли до координат вершин межових елементів додаються нормовані значення вибраних користувачем функцій (за усталеним налаштуванням – переміщень) (рис. 5).

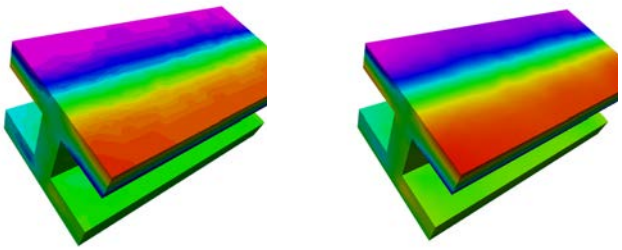

Рис. 4. Візуалізація об'єкта з використанням різної кількості кольорів

На рис. 5 наведено приклад візуалізації трансформації паливного бака літального апарата (з урахуванням симетрії розглядалася чверть конструкції) під дією внутрішнього тиску. Для моделювання використовувалися оболонкові трикутні СЕ. Наявність поперечних ребер жорсткості моделювалася завданням товщини відповідних елементів.

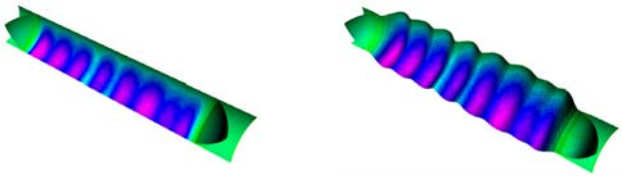

Рис. 5. Приклад відображення трансформації конструкції

Висновки. У роботі представлено програмну реалізацію постпроцесора Mesh з відкритим початковим кодом, якій можна використовувати для аналізу результатів скінченно-елементних розрахунків, отриманих із використанням низки сторонніх FEA-систем.

Головною перевагою програми Mesh порівняно з наявними аналогами є її реалізація у вигляді вебзастосунку, що робить її кросплатформною та такою, що може працювати підтримкою сучасних хмарних технологій.

Mesh надає користувачеві інтуїтивно зрозумілий графічний інтерфейс для візуалізації великих наборів даних, у тому числі за допомогою асинхронних методів обробки.

Як варіант подальшого вдосконалення Mesh передбачається розширення функціональності цього застосунку для постаналізу нестаціонарних задач.

Завантажити початковий код Mesh можна за посиланням [28].

ЛІТЕРАТУРА

1. Zienkiewicz O.C., Taylor R.L., Zhu J.Z. The Finite Element Method: Its Basis and Fundamentals. Sixth edition. Butterworth-Heinemann, 2016. 753 p.
2. Finite element analysis software. URL: http://surl.li/ukbjyp (дата звернення: 08.01.2025).
3. Engineering Simulation & 3D Design Software | Ansys. URL: https://www.ansys.com/ (дата звернення: 08.01.2025).
4. MSC Nastran – Multidisciplinary Structural Analysis. URL: http://surl.li/schezo (дата звернення: 08.01.2025).
5. COMSOL Multiphysics ® Modelling Software. URL: https://www.comsol.com/ (дата звернення: 08.01.2025).
6. Best Open-Source Finite Element Analysis Software. URL: http://surl.li/vmblnr (дата звернення: 08.01.2025).
7. Lee J.Y., Ahn S.Y. Interactive visualization of elasto-plastic behavior through stress paths and yield surfaces in finite element analysis. URL: http://surl.li/vmrnkm (дата звернення: 09.01.2025).
8. Ansys Workbench | Simulation Integration Platform. URL: http://surl.li/sjkenb (дата звернення: 09.01.2025).
9. Abaqus/CAE. URL: http://surl.li/ketspv (дата звернення: 09.01.2025).
10. SOLIDWORKS Simulation. URL: http://surl.li/iawbud (дата звернення: 09.01.2025).
11. ParaView – Open-source, multi-platform data analysis and visualization application. URL: https://www.paraview.org/ (дата звернення: 09.01.2025).
12. Tecplot Visualization and Analysis Tools for CFD Post-processing. URL: https://tecplot.com/ (дата звернення: 09.01.2025).

13. MATLAB – MathWorks. URL: http://surl.li/nnrxpk (дата звернення: 09.01.2025).
14. QuickField Postprocessor. URL: https://quickfield.com/post.htm (дата звернення: 09.01.2025).
15. NumPy. URL: https://numpy.org/ (дата звернення: 09.01.2025).
16. Matplotlib – Visualization with Python. URL: https://matplotlib.org/ (дата звернення: 09.01.2025).
17. VTK – The Visualization Toolkit. URL: https://vtk.org/ (дата звернення: 09.01.2025).
18. The PyVista Project. URL: https://pyvista.org/ (дата звернення: 10.01.2025).
19. WebGL Overview – The Khronos Group Inc. URL: https://www.khronos.org/webgl/ (дата звернення: 10.01.2025).
20. Home – Ecma International. URL: https://ecma-international.org/ (дата звернення: 10.01.2025).
21. WebGL Fundamentals. URL: https://webglfundamentals.org/ (дата звернення: 10.01.2025).
22. Гнєздовський О. В. Відкрита об'єктно-орієнтована архітектура систем скінченно-елементного аналізу : дисертація на здобуття наукового ступеня доктора філософії за спеціальністю 122 «Комп'ютерні науки». Запорізький національний університет, Запоріжжя, 2023. 124 с.
23. React. URL: https://react.dev/ (дата звернення: 18.01.2025).
24. A three-dimensional finite element mesh generator with built-in pre- and post-processing facilities. URL: https://gmsh.info/ (дата звернення: 19.01.2025).
25. Netgen/NGSolve. URL: https://ngsolve.org/ (дата звернення: 19.01.2025).
26. QFEM – Simple FEM Solver. URL: http://surl.li/eikjdv (дата звернення: 19.01.2025).
27. Гоменюк С. І., Козуб В. Ю. Особливості використання паралельних обчислень в пакеті прикладних програм «МІРЕЛА+». *Вісник Хмельницького національного університету*. 2022. № 6. Т. 2. С. 60–64.
28. Mesh. URL: https://surl.li/vnpczu (дата звернення: 31.01.2025).

REFERENCES

1. Zienkiewicz, O.C., Taylor, R.L., Zhu, J.Z. (2016). The Finite Element Method: Its Basis and Fundamentals. Sixth edition. Butterworth-Heinemann, 753 p.
2. Finite element analysis software. URL: http://surl.li/ukbjyp (accessed 08.01.2025).
3. Engineering Simulation & 3D Design Software | Ansys. URL: https://www.ansys.com/ (accessed 08.01.2025).
4. MSC Nastran – Multidisciplinary Structural Analysis. URL: http://surl.li/schezo (accessed 08.01.2025).
5. COMSOL Multiphysics ® Modelling Software. URL: https://www.comsol.com/ (accessed 08.01.2025).
6. Best Open-Source Finite Element Analysis Software. URL: http://surl.li/vmblnr (accessed 08.01.2025).
7. Lee, J.Y., Ahn, S.Y. Interactive visualization of elasto-plastic behavior through stress paths and yield surfaces in finite element analysis. URL: http://surl.li/vmrnkm (accessed 09.01.2025).
8. Ansys Workbench | Simulation Integration Platform. URL: http://surl.li/sjkenb (accessed 09.01.2025).
9. Abaqus/CAE. URL: http://surl.li/ketspv (accessed 09.01.2025).
10. SOLIDWORKS Simulation. URL: http://surl.li/iawbud (accessed 09.01.2025).
11. ParaView – Open-source, multi-platform data analysis and visualization application. URL: https://www.paraview.org/ (accessed 09.01.2025).
12. Tecplot Visualization and Analysis Tools for CFD Post-processing. URL: https://tecplot.com/ (accessed 09.01.2025).
13. MATLAB – MathWorks. URL: http://surl.li/nnrxpk (accessed 09.01.2025).
14. QuickField Postprocessor. URL: https://quickfield.com/post.htm (accessed 09.01.2025).
15. NumPy. URL: https://numpy.org/ (accessed 09.01.2025).
16. Matplotlib – Visualization with Python. URL: https://matplotlib.org/ (accessed 09.01.2025).
17. VTK – The Visualization Toolkit. URL: https://vtk.org/ (accessed 09.01.2025).
18. The PyVista Project. URL: https://pyvista.org/ (accessed 10.01.2025).
19. WebGL Overview – The Khronos Group Inc. URL: https://www.khronos.org/webgl/ (accessed 10.01.2025).
20. Home – Ecma International. URL: https://ecma-international.org/ (accessed 10.01.2025).
21. WebGL Fundamentals. URL: https://webglfundamentals.org/ (accessed 10.01.2025).
22. Hniezdovskyi, O.V. (2023). Vidkryta obiektno-oriientovana arkhitektura system skinchenno-elementnoho analizu [Open object-oriented architecture of finite element analysis systems]. *Doctor's thesis*. Zaporizhzhia National University, Zaporizhzhia. 124 p. [in Ukrainian].
23. React. URL: https://react.dev/ (accessed 18.01.2025).
24. A three-dimensional finite element mesh generator with built-in pre- and post-processing facilities. URL: https://gmsh.info/ (accessed 19.01.2025).
25. Netgen/NGSolve. URL: https://ngsolve.org/ (accessed 19.01.2025).
26. QFEM – Simple FEM Solver. URL: http://surl.li/eikjdv (accessed 19.01.2025).
27. Homeniuk, S.I., Kozub, V.Yu. (2022). Osoblyvosti vykorystannia paralelnykh obchyslen v paketi prykladnykh prohram "MIRELA+" [Peculiarities of using parallel computing in the package of applied programs "MIRELA+"]. *Bulletin of Khmelnytsky National University*, 6 (2), 60–64 [in Ukrainian].
28. Mesh. URL: https://surl.li/vnpczu (accessed 31.01.2025).