УДК 004.896:539.3 DOI: http://dx.doi.org/10.20998/2078-9130.2016.46.88044

Д. В. БРЕСЛАВСЬКИЙ, М. В. КОНОВАЛОВ, О. А. ТАТАРІНОВА

ПРЕПРОЦЕСОРНІ ПРОГРАМИ ДЛЯ ПІДГОТОВКИ СКІНЧЕННОЕЛЕМЕНТНИХ СІТОК У ВИПАДКУ ВЕЛИКИХ ДЕФОРМАЦІЙ

Надано опис програмного забезпечення для препроцесорної підготовки даних при розв'язанні задач механіки деформівного твердого тіла методом скінченних елементів. Для випадку великих деформацій передбачено застосування узагальненого підходу Лагранжу-Ейлеру. Для генерації трьохвимірних моделей призматичних тіл застосовано програми тріангуляції двовимірних областей складної геометрії. Розглянуто основні алгоритми побудови сіток при розбитті матеріальних та додаткових «повітряних» областей, їхню програмну реалізацію виконано за допомогою мови програмування Python. Наведено приклади побудови сіток для моделей різних форм.

Ключові слова: метод скінченних елементів, скінченноелементна сітка, узагальнений підход Лагранжу-Ейлеру, препроцесорна програма.

Приведено описание программного обеспечения для препроцессорной подготовки данных при решении задач механики деформированного твердого тела методом конечных элементов. Для случая больших деформаций предусмотрено использование обобщенного подхода Лагранжа-Эйлера. Для генерации трехмерных моделей призматических тел использованы программы триангуляции двумерных областей сложной геометрии. Рассмотрены основные алгоритмы построения сеток при разбиении материальных и дополнительных «воздушных» областей, программная реализация которых выполнена с помощью языка программирования Python. Приведены примеры построения сеток для моделей различных форм.

Ключевые слова: метод конечных элементов, конечноэлементная сетка, обобщенный подход Лагранжа-Эйлера, препроцессорная программа.

The description of algorithms and software for Finite Element Method data preprocessing, which is designed for solution the problems of Solid Mechanics, is presented. Three dimensional Finite Element models for prismatic solids are regarded. For the case of large strains the arbitrary Lagrange-Euler approach is used. Two approaches which are realized in separate program unit are presented. Both of them are based on preliminary two-dimensional triangulation of the areas with complex shape. The first approach bases on the figure triangulation with use of triangles with optional shape, the second demands uniform mesh with rectangular triangles. The algorithms for meshing of material and additional 'air' areas with simultaneous generation of elements are regarded. The procedures of the generation of elements groups and their integration into the model are presented. The code design strategy with description of entity-relationship diagram is discussed. The graphical user interface for the data preprocessing including the matrix of index and co-ordinates array is described, the procedure of user's actions is presented. The program realization of above described algorithms was done by use of programming language Python. The examples of mesh building with information of each stage of meshing for models of different shape are presented.

Keywords: finite element method, finite element mesh, arbitrary Lagrange-Euler approach, preprocessor program.

Вступ. В останній час поширюється інтерес до розв'язання задач механіки деформівного твердого тіла у випадку, коли деформації є великими [1, 2]. Одним з поширених підходів є застосування при цьому узагальненого підходу Лагранжу-Ейлеру ALE [3, 4]. Згідно ньому, скінченноелементна сітка залишається нерухомою впродовж всього процесу деформування [3], а матеріал тіла, що деформується «тече» крізь неї. Для використання цього підходу застосовують методику, що потребує створення у скінченноелементних сітках двох зон – одна з яких містить «матеріальні» елементи, а інша – «повітряні».

Останні є необхідними для прийому матеріальних точок при деформуванні. Зрозуміло, що інші елементи, які на початку були «матеріальними», при цьому набувають ознаки «повітряних»

Реалізація описаного підходу потребує створення додаткового програмного забезпечення, в якому працюють такі алгоритми. Для його проектування та розробки у теперішніх умовах застосовують комбінації структурно-логічної та об'єктно-орієнтованих методологій [5], при чому останню застосовують для реалізації сервісних модулів скінченноелементного програмного забезпечення [6].

Програмну реалізацію алгоритмів препроцесорної обробки даних виконують з використання різних алгоритмічних мов, серед яких найбільш поширеними є Fortran, C++, Java, та в останні роки й швидко поширювану мову Python [7].

В статті розглянуто опис одного з варіантів препроцесорного програмного забезпечення, створеного на мові програмування Python для підготовки двох- та трьохвимірних моделей, які є потрібними при застосуванні в розрахунках методом скінченних елементів у випадку великих деформацій.

Опис алгоритмів. Розглянемо програмне забезпечення, що побудовано на базі раніш розроблених препроцесорів – *Divider* [8] та *Transformer* [9] для генерації трьохвимірних сіток для призматичних моделей чи одного *Divider* для двовимірних. На мові Python розроблено окремий додаток *RD*, в якому відлагоджено нову технологію розв'язання задач створення двохкомпонентних сіток.

У новостворену версію 2.7 програми *Divider* додано характеристику «матеріал елементу» з подальшою модифікацією *Transformer* та *Builder* [9] для підтримки альтернативного шляху до вирішення проблем

додавання й видалення нових елементів. При достатньо малій сітці розбиття вказані операції ототожнюються з переводом складу групи комірок з матеріальної моделі фігури на «повітря».

Робота з Divider'ом передбачає побудову контуру фігури, для якої потрібно провести розрахунки, шляхом створення вузлових точок. Кожна точка є складовою контуру – тому при початку роботи спершу треба додати його. Далі додаються нові чи редагуються існуючі вузли з необхідним розташуванням.

Після завершення формування моделі проводиться її розбиття на скінченні елементи та генерація файлу вхідних даних. Для створення з двовимірної сітки трьохвимірної цей файл передається до програми *Transformer*. Візуалізація нової сітки пройде у програмі *Builder*.

Запропонований алгоритм – розбиття моделі спільно з «матеріалом» слугує для прискорення подальшого маніпулювання її формою і дозволяє уникати коштовних операцій побудови додаткової сітки та її приєднання до початкової: «зниклі» у результаті деформування елементи змінюють «матеріал» з «фігури» на «повітря». Аналогічно, у випадку «появи» «нових вузлів» у фігурі (наприклад, при згині), відповідні «повітряні» елементи «переходять» до фігури. Такий процес у програмному представленні відповідає швидким операціям пошуку у масиві та перепризначення змінної-індикатору.

Відмінності розробленого на мові Python програмного додатку *RD* від раніш створеної програми *Divider* [8] полягає у способі побудови фігури (компонування прямокутниками проти завдання вузлів полігону) та формі трикутників (прямокутні проти довільних). Візуально зіставити роботу двох програм можна на рис. 1 та рис. 2.

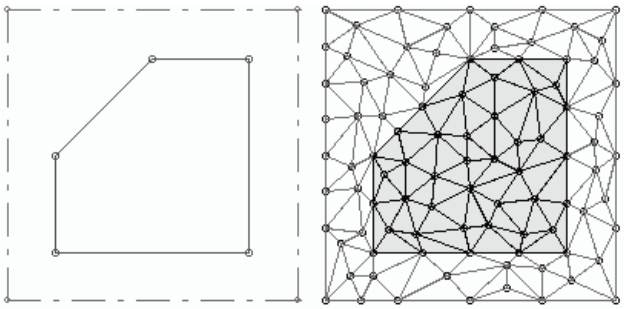

Рисунок 1 – Приклад роботи додатку Divider 2.7

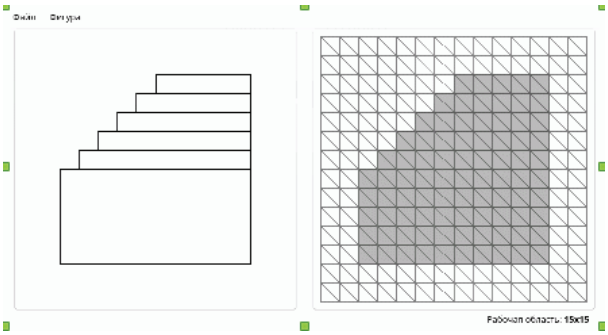

Рисунок 2 – Приклад роботи додатку *RD*

Надамо функціональний опис програми Divider. Користувач формує фігуру шляхом завдання точок замкненої лінії, що її окреслюють, після чого точки об'єднуються у прямокутники, виправляються порушені контури – і, нарешті, відбувається поділ завеликих за площею трикутників на менших до отримання фінальної сітки розбиття. Для цього існує два методи: методом серединної точки (з одного елементу створюється три, з спільною вершиною, що одночасно є центром старого трикутника) та поділом на два (обирається точка на одній з граней елементу).

Формування відбувається наступним чином. Після створення контуру з першим вузлом треба викликати його контекстне меню натиском правої клавіші миші і обрати пункт Split contour – з'явиться меню редагування новоствореного вузла і користувач отримує можливість завдання його положення. Дане меню можна викликати у будь-який момент і змістити вузол. Основною перевагою програми *RD* у порівнянні з *Divider* є те, що вона формує регулярну сітку. Також певним позитивним моментом є те, що *Divider* характеризується кодовою базою у 6000+ строк, у той час як додаток *RD* обмежується 2000+ строками (виграш досягнуто за рахунок побудови інтерфейсу у конструкторі та використання готових рішень *Qt* – супроти написання власних аналогів на *Swing*).

Недоліки: не всі типи фігур доступні для побудови (порівняйте рис. 1 та 2), та за підходом програми *RD* потрібно створювати дуже дрібну сітку, що не унебезпечує від похибок у розрахунках. В деяких випадках, для моделювання задач з прямокутними областями, рівномірна сітка нового додатку має переваги.

Розглянемо алгоритм роботи програми створення сітки з двома типами елементів. Базовим структурним блоком для алгоритму є прямокутник, з котрих користувач конструює фігуру. Його представлено наступними даними: координати верхнього лівого кута, ширина, висота, кількість вузлів сітки по горизонталі, кількість вузлів по вертикалі та кількості повітряних вузлів для кожної із сторін (починаючи з верхньої, за годинниковою стрілкою).

Слід зауважити, що повітряні вузли не входять у число «кількість вузлів по горизонталі чи вертикалі», а добудовуються понад це число з обох боків (знизу та гори). Також варто пам'ятати, що повітряні області мають один спільний вузол з фігурою і той завжди враховується при розбиті: тому повітряна сітка шириною у два вузли дає чотири елементи – у той час як сітка фігури шириною у два вузли задає лише два елементи (рис. 3, елементи фігури зафарбовано).

Сітка у програмі представлена множинами вузлів та трикутників (елементів). Вузол характеризується координатами, «матеріалом» (приймає значення «фігура»/«повітря») та своїм індексом у масиві вузлів. Елементи визначаються переліком індексів своїх вершин. «Матеріал» елементу розраховується за відповідної характеристикою його вершин: якщо хоч один з них є «повітряним», то увесь елемент вважається «повітряним».

Розбиття формується шляхом об'єднання складових сіток, що утворені на базі прямокутників, з яких зібрано фігуру. Прямокутники розбиваються згідно з

заданими користувачем при їх створенні параметрами (кількість вузлів, крок сітки) за наступною схемою: 1) створюються вузли майбутньої сітки; 2) перші два елементи першого рядка та два елементи другого умовно об'єднуються у прямокутник, який розділяється діагоналлю з верхнього лівого кута у правий нижній. 3) утворені діагоналлю трикутники (елементи) додаються до сітки; 4) виконуються зсув ліворуч на один вузол; 5) повторюються кроки 2-3 для відповідних вузлів. При досягненні кінця рядка відбувається перехід до першого вузла наступного рядку.

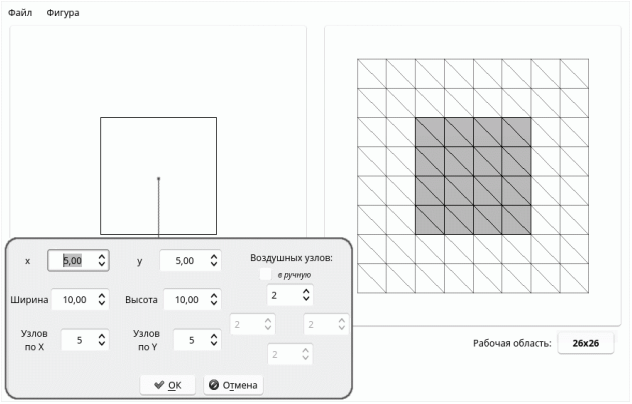

Рисунок 3 – Робоче вікно програми з накладеним зверху діалогом створення фігури (закруглений прямокутник)

Побудовані таким чином сітки переносяться на полотно цільової сітки за винятком вузлів (та елементів, що на них побудовано) розташованих у граничних зонах. Гранична зона – лінія крайніх нижніх чи правих вузлів сітки прямокутника, при умові, що той має сусіда (відповідно) знизу чи праворуч. Слід зауважити, що крайні верхні чи ліві вузли прямокутника, що має сусіда згори чи ліворуч, не вважаються присутніми у граничній зоні цього прямокутника і переносяться на даному етапі. Одночасно, з точку зору сусіднього згори/зліва прямокутника вони належать його граничній зоні і він їх не переносить на полотно цільової сітки. Причина такої поведінки в уникненні дублювання спільних вузлів.

Після того, як перенос вузлів завершено, починається обробка граничних зон. Беруться перші два вузли однієї сторони і перші два вузли сусідньої (на рис. 3 це, відповідно, верхні та нижні вузли лівого зафарбованого прямокутника) та на них формуються трикутники сітки. У випадку, коли сусідній прямокутник не має відповідного вузла, той спершу генерується і його так само використовують для генерації сітки.

На рис. 4 представлено процес розбиття фігури.

Збереження розбиття у файл відбувається за наступним алгоритмом:

1) сортуються прямокутники фігури за координатами (спершу по осі OX);

2) записується преамбула (замість N та M залишаємо пробіли);

3) для кожного прямокутнику записується його сітка, за винятком елементів із граничної зони;

4) опрацьовуються граничні зони (зберігаючи індекси новостворених вузлів у буфер, розраховуючи їх координати і матеріал);

5) для кожного примітиву записуються координати вузлів його сітки;

6) записуються координати вузлів з буферу вузлів граничних зон;

7) для кожного примітиву записується матеріал вузлів його сітки;

8) записується матеріал вузлів з буферу вузлів граничних зон;

9) записується поверх пробілів з кроку 2 дійсні величини N, M.

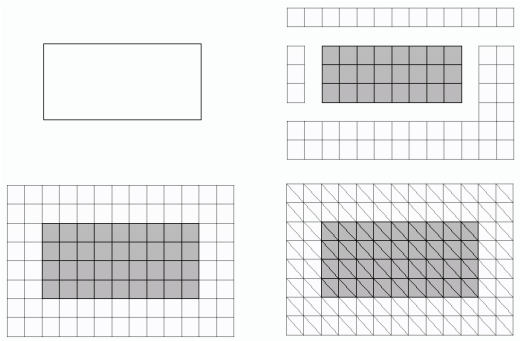

Рисунок 4 – Процес розбиття

Суттєвою перевагою такого рішення є те, що до сіток не висувається додаткових вимог, що обмежували б їх форму. Звичайно, аби об'єднати дві сітки, необхідно, щоб вузли відповідних сторін зійшлись для об'єднання, проте ніяких обмежень на «перпендикулярні» сторони не накладається. Згадане розширення функціоналу потребувало внесення коректив у алгоритми поділу трикутниками (задля збереження новостворених вузлами складу відповідних їм вузлів-предків).

Опис препроцесорної програми. Програмний додаток RD написано на мові програмування Python (v3.5) з графічним інтерфейсом, побудованим на фреймоворку Qt (v5.3). «Віджети» робочого та допоміжних вікон створено за допомогою його інструменту Qt Designer. За посередництво між Qt та Python відповідає бібліотека PyQt (v5.3).

Python – це інтерпретована об'єктно-орієнтована мова програмування високого рівня з динамічною семантикою. Структури даних високого рівня разом із динамічною семантикою та динамічним зв'язуванням роблять її привабливою для швидкої розробки програм [7], що у купі з зрілою стандартною бібліотекою й було основним критерієм при виборі мови програмування.

Програма має три ієрархічні рівні (рис. 5). На верхньому зосередження вся GUI-логіка, обробка натискань клавіш клавіатури чи миші, графічне представлення фігури та сітки розбиття.

Другим є абстракція фігури, що відповідає за наступні операції: компонування фігури з прямокутників; редагування фігури; управління робочім простором (розміщення позиціювання «камери», масштаб зображення);підготовка даних для малювання на верхньому рівні; основний алгоритм розбиття та збере-

ження розбиття.

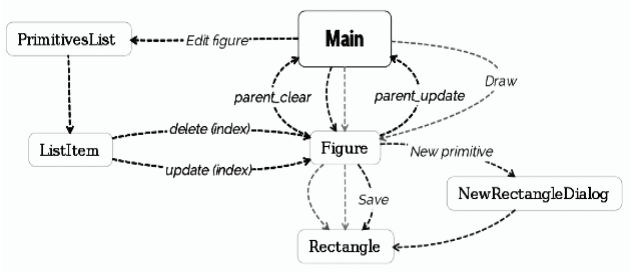

Рисунок 5 – Діаграма відношення між класами програми

Третій рівень: представлення прямокутника, розбиття (без граничних зон), відображення прямокутника.

Інтерфейс побудовано на базі Qt – крос-платформового інструментарію розробки програмного забезпечення, з використанням «обгортки» для мови Python - PyQt.

Робоче вікно програми розділено на дві області, розташовані горизонтально (рис. 6). Ліва представляє фігуру, права зображує її розбиття. Поле фігури надає контекстне меню по правому кліку мишею. У верхній частині вікна знаходиться панель дій з двома пунктами «Файл» та «Фігура».

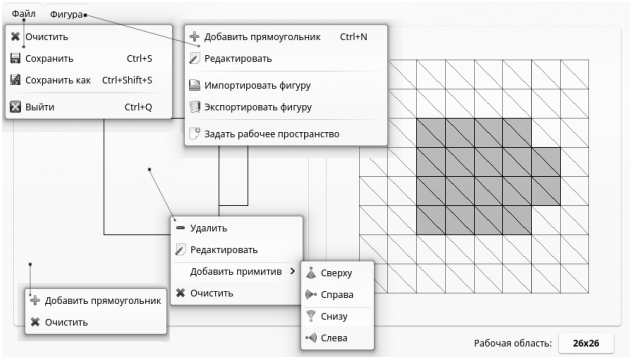

Рисунок 6 – Приклад робочого вікна із активними віджетами меню

Пункт «Файл» містить стандартні дії завдання нової моделі, її збереження тощо. :

Пункт «Фігура» містить наступні дії:

– «Додати прямокутник»: викликається діалог створення;

– «Редагувати»: викликається діалог з переліком усіх прямокутників для зміни параметрів прямокутнику чи його видалення з фігури;

– «Імпортувати фігуру»: завантажує з файлу фігуру (вона замістить ту фігуру, над якою велась робота до цього - якщо велася);

– «Експортувати фігуру»: зберігає у файл створену фігуру;

– «Задати робочий простір»: параметри відображення робочого простору (масштаб камери та її зміщення відносно робочого простору).

Контекстне меню прямокутнику викликається у випадку знаходження кліку над площею даного прямокутника. Містить дії:

– «Видалити»: видаляє даний прямокутник з фігури;

– «Редагувати»: дозволяє змінити параметри даного прямокутнику;

– меню «Додати примітив»: аналогічно пункту «Додати прямокутник» із тою різницею, що створений прямокутник матиме спільні вузли з даним (використовується саме для побудови багатокомпонентних фігур);

– «Очистити»: видаляє даний прямокутник.

Контекстне меню вільного простору містить дії:

– «Додати прямокутник»: викликається діалог створення;

– «Очистити»: видаляє даний прямокутник.

Інтерфейс програми виконано у конструкторі Qt Designer і збережено у XML-сумісному форматі інструменту, вікна програми ініціалізуються ними при завантаженні програми.

Для додавання/редагування прямокутника викликається окреме вікно (рис. 6), через яке користувач передає програмі параметри. У випадку добудови прямокутника до існуючого, на певні поля накладаються наступні обмеження для збереження сумісності сіток:

1) блокується відповідна координата аби фігури не розійшлися (відповідно – x-координата для додавання зліва/справа, y – для вертикального розширення);

2) крок відповідного поля «Ширина»/«Висота» виставляється рівним кроку сітки «батьківського» прямокутника, кількість елементів розраховується автоматично;

3) накладаються обмеження на другу координату, так щоб зсувом не можна було роз'єднати сітки і вони мали хоча б один спільний вузол (повітряні вузли не рахуються);

4) фіксується на позначці «0» кількість повітряних елементів зі сторони, до якою приєднано інший трикутник.

При виклику діалогу «Додати примітив» той автоматично попереджає перекриття елементів чи роз'єднання прямокутників (розбіжність вузлів) у результаті позиціювання. Однак під час редагування перевірка на дані помилки не здійснюється і користувач має пильнувати свої дії.

Конструювання фігури відбувається шляхом компонування прямокутників довільних розмірів. Завдання шару повітряних елементів навкруги відбувається шляхом введення кількості повітряних вузлів, що будуть згенеровані у той чи інший бік.

Висновки. В статті розглянуто дві препроцесорні програми, які призначено для генерації скінченноелементних сіток для розв'язання задач механіки деформівного твердого тіла при великих деформаціях. Обговорено альтернативні підходи до алгоритмічного забезпечення. Наголосимо, що підхід додатку *RD* забезпечить кращі результати за умов його модифікації на випадок границь у вигляді ламаних ліній.

Список літератури:

1. *Кукуджанов В. Н.* Компьютерное моделирование деформирования, повреждаемости и разрушения неупругих материалов и конструкций / *В. Н. Кукуджанов*. – М.: МФТИ, 2008. – 215 с.

2. *Рудаков К. М.* Алгоритми розв'язання крайових задач методом скінченних елементів при великих пружнопластичних деформаціях та з урахуванням пошкодженості структури матеріалу. Повідомлення 1. Логарифмічні деформації / *К. М. Рудаков, І. Л. Сидоренко* // Вісник НТУУ «КПІ». Машинобудування. – 2012. – Вип. 66. – С. 138-144.

3. *Муйземнек А. Ю.* Математическое моделирование процессов удара и взрыва в программе LS DYNA / *А. Ю. Муйземнек, А. А. Богач*. – Пенза: ИИЦ ПГУ, 2005. – 106 с.

4. *Hallquist J.* LS DYNA Theoretical Manual – Livermore Software Technology / *J. Hallquist*. – USA : Corporation Livermore, 2005. – 320 p.

5. *Одинцов И. О.* Профессиональное программирование. Системный подход / *И. О. Одинцов*. – СПб : БХВ-Петербург, 2002. – 512 с.

6. *Mackerle J.* Object-oriented programming in FEM and BEM: a bibliography (1990–2003) / *J. Mackerle* // Advances in Engineering Software. – 2004. – № 35. – P. 325-336.

7. *Маккини У. С.* Python и анализ данных: пер. с англ. / *У. С. Маккини*. – М.: ДМК-Пресс, 2009. – 482 с.

8. *Бреславский Д. В.* Программные средства для конечноэлементного моделирования двумерных задач теории ползучести / *Д. В. Бреславский, Ю. Н. Корытко, П. М. Лисак* // Вісник Національного технічного університету «ХПІ». – Х.: НТУ «ХПІ», 2007. – № 38. – С. 24-29.

9. *Бреславський Д. В.* Розробка препроцесорного програмного забезпечення для побудови трьохвимірних моделей / *Д. В. Бреславський, О. А. Татарінова, М. С. Кокорєв* и др. // XXIV Міжнародна науково-практична конференція. – Х.: 2016. – С. 50.

References (transliterated):

1. Kukudzhanov, V. N. Kompjuternoe modelirovanie deformirovanija, povrezhdaemosti i razrushenija. Moskow: MFTI, 2008. 215 p.

2. Rudakov K. M., Sydorenko I. L. Alhorytmy rozvyazannya krayovykh zadach metodom skinchennykh elementiv pry velykykh pruzhno-plastychnykh deformatsiyakh ta z urakhuvannyam poshkodzhenosti struktury materialu. Povidomlennya 1. Loharyfmichni deformatsiyi. Visnuk NTUU "KPI". 2012. Vol. 66. pp. 138-144.

3. Mujzemnek A. Ju., Bogach A. A. Matematicheskoe modelirovanie processov udara i vzryva v programme LS DYNA. Penza: IIC PGU, 2005. 106 p.

4. Hallquist J. LS DYNA Theoretical Manual – Livermore Software Technology. USA : Corporation Livermore, 2005. 320 p.

5. Odincov I. O. Professionalnoe programmirovanie. Sistemnyj podhod. Sankt-Peterburg : BHV-Peterburg, 2002. 512 p.

6. Mackerle J. Object-oriented programming in FEM and BEM: a bibliography (1990–2003). Advances in Engineering Software. 2004. No 35. pp. 325–336.

7. Makkini U. S. Python I analiz dannyh: per. s angl. Moskow : DMK-Press, 2009. 482 p.

8. Breslavskij D. V., Korytko Ju. N., Lisak P. M. Programmnye sredstva dlja konechnoelementnogo modelirovanija dvumernyh zadach teorii polzuchesti. Visnuk nacionalnogo tehnichnogo universitetu «KhPI». 2007. No 38. pp. 24-29.

9. Breslavskij D. V., et al. Rozrobka preprocesornogo programnogo zabezpechennja dlja pobudovi trohvimirnih modelej. XXIV Mizhnarodna naukovo-praktichna konferencija. Kharkiv, 2016. p. 50.

Надійшла (received) 20.09.2016

Бібліографічні описи / Библиографические описания / Bibliographic descriptions

Препроцесорні програми для підготовки скінченноелементних сіток у випадку великих деформацій / Д. В. Бреславський, М. В. Коновалов, О. А. Татарінова // Вісник НТУ «ХПІ». Серія: Динаміка і міцність машин. – Х.: НТУ «ХПІ», 2016. – № 46 (1218). – С. 16–20. – Бібліогр.: 9 назв. – ISSN 2078-9130.

Препроцессорные программы для подготовки конечноэлементных сеток в случае больших деформаций / Д. В. Бреславский, М. В. Коновалов, О. А. Татаринова // Вісник НТУ «ХПІ». Серія: Динаміка і міцність машин. – Х.: НТУ «ХПІ», 2016. – № 46 (1218). – С. 16–20. – Бібліогр.: 9 назв. – ISSN 2078-9130.

Preprocessors for Finite Element Method meshes generation for the case of large strains / D. V. Breslavsky, M. V. Konovalow, O. A. Tatarinova // Bulletin of NTU "KhPI". Series: Dynamics and strength of machines. – Kharkiv: NTU "KhPI", 2016. – № 46 (1218). – P. 16–20. – Bibliogr.: 9. – ISSN 2078-9130.

Відомості про авторів / Сведения об авторах / About the Authors

Бреславський Дмитро Васильович – доктор технічних наук, професор, завідувач кафедри комп'ютерного моделювання процесів та систем, НТУ «ХПІ», тел.: (057) 707-64-54, e-mail: brdm@kpi.kharkov.ua

Бреславский Дмитрий Васильевич – доктор технических наук, профессор, заведующий кафедры компьютерного моделирования процессов и систем, НТУ «ХПИ», тел.: (057) 707-64-54, e-mail: brdm@kpi.kharkov.ua

Breslavsky Dmytro Vasylovych – Doctor of Technical Sciences, Professor, Head of the Department of Computer Modeling of Processes and Systems, NTU "KhPI", tel.: (057) 707-64-54, e-mail: brdm@kpi.kharkov.ua

Коновалов Микола Володимирович – студент інженерно-фізичного факультету, НТУ «ХПІ», тел.: (057) 707-64-54, e-mail: mkonovalow@ukr.net

Коновалов Николай Владимирович – студент инженерно-физического факультета, НТУ «ХПИ», тел.: (057) 707-64-54, e-mail: mkonovalow@ukr.net

Konovalov Mykola Volodymyrovych – student of the Faculty of Physical Engineering, NTU "KhPI", tel.: (057) 707-64-54, e-mail: mkonovalow@ukr.net

Татарінова Оксана Андріївна – кандидат технічних наук, доцент, НТУ «ХПІ», тел.: (057) 707-60-58, e-mail: ok.tatarinova@gmail.com

Татаринова Оксана Андреевна – кандидат технических наук, доцент, НТУ «ХПИ», тел.: (057) 707-60-58, e-mail: ok.tatarinova@gmail.com

Tatarinova Oksana Andriivna – Candidate of Technical Sciences (Ph. D.), Docent, NTU "KhPI", tel.: (057) 707-60-58, e-mail: ok.tatarinova@gmail.com