УДК 004.94
А.П. Здолбіцький
Луцький національний технічний університет

ОБЄКТНО–ОРІЄНТОВАНА ПРОГРАМНА РЕАЛІЗАЦІЯ МАТЕМАТИЧНОЇ МОДЕЛІ ДЕФОРМУВАННЯ ДЕРЕВИНИ У ПРОЦЕСІ НЕІЗОТЕРМІЧНОГО ВОЛОГОПЕРЕНЕСЕННЯ

А.П. Здолбіцький. Обєктно-орієнтована програмна реалізація математичної моделі деформування деревини у процесі неізотермічного вологоперенесення. *У межах об'єктно-орієнтованого підходу розроблене прикладне програмне забезпечення для чисельної реалізації математичних моделей тепломасоперенесення і пружнов'язкопластичного деформування деревини під час сушіння. Розроблений програмний комплекс мовою програмування Java містить інформаційну модель та інтерфейс програмної системи у вигляді пакетів класів та відношень між ними з використанням графічних діаграм UML, компоненти програмного коду, обчислювальні схеми реалізації МСЕ. Розроблені класи відображають сутність об'єктно-орієнтованої реалізації методу скінченних елементів. Це створює можливість інтегрування розроблених програм до існуючих систем автоматизованого моделювання з метою розширення їх функціональних можливостей.*
Ключові слова: *діаграма UML, вологоперенесення, деформування, моделювання.*
Форм. 2. Рис. 11. Літ. 17.

А.П. Здолбицкий. Объектно-ориентированная программная реализация математической модели деформирования древесины в процессе неизотермических влагопереноса. *В рамках объектно-ориентированного подхода разработано прикладное программное обеспечение для численной реализации математических моделей тепломассопереноса и упруговязкопластического деформирования древесины при сушке. Разработанный программный комплекс на языке программирования Java содержит информационную модель и интерфейс программной системы в виде пакетов классов и отношений между ними с использованием графических диаграмм UML, компоненты программного кода, вычислительные схемы реализации МСЭ. Разработанные классы отражают сущность объектно-ориентированной реализации метода конечных элементов. Это создает возможность интегрирования разработанных программ существующих систем автоматизированного моделирования с целью расширения их функциональных возможностей.*
Ключевые слова: *диаграмма UML, влагоперенос, деформирования, моделирование.*

A.P. Zdolbitskyy. Object-oriented software implementation of mathematical models of deformation of wood in non-isothermal moisture transfer process. *Within an object-oriented approach developed application software for the numerical implementation of mathematical models and moisture transfer elastoviscoplastic deformation of wood during drying . The developed software package contains Java programming language and interface information model of software system in a package of classes and the relationships between them using graphical diagrams UML, component code, computational schemes of the Union. Developed classes reflect the essence of object-oriented implementation of the finite element method. This creates the possibility of integration with existing applications developed computer-aided modeling in order to extend their functionality.*
Keywords: *diagram UML, moisture transfer, deformation, modeling.*

Аналіз досліджень. На теперішній час існує порівняно небагато публікацій застосування об'єктно-орієнтованого підходу для програмної реалізації МСЕ [1-4], в яких запроектовані класи базових компонентів МСЕ (елементів, вузлів, граничних умов та навантажень). Їх аналіз дозволяє стверджувати про те, що безпосереднє створення об'єктів в програмному коді породжує необхідність перекомпілювання програми для нових даних. Потребує подальшого вдосконалення задача візуального створення та редагування об'єктів шляхом їх інтегрування з генератором розбиття геометричної області. У даній статті в рамках об'єктно-орієнтованого підходу розроблено алгоритмічне та програмне забезпечення для чисельної реалізації наведених у попередніх публікаціях [7,8] математичних моделей деформування деревини у процесі неізотермічного вологоперенесення.

Обєктно-орієнтований підхід для програмної реалізацій. Для побудови програмної моделі методу скінченних елементів необхідно здійснити об'єктно-орієнтований аналіз методу, виділити програмні моделі, які є самостійною сутністю об'єктно-орієнтованої реалізації алгоритмів МСЕ у контексті основних питань і термінів методу. З одного боку, програма методу скінченних елементів має бути визначена у термінах ООП – алгоритми, методи і дані (класи), а з іншого – основні поняття МСЕ мають бути формалізовані у контексті ООП.

Такий підхід дозволяє розкласти програмну систему у вигляді класів та відношень між ними. Пояснення зв'язків між спроектованими класами та взаємодію об'єктів цих класів здійснено за допомогою діаграм UML [5]. В окремі пакети виділено класи, які реалізують: геометричні та фізико-механічні характеристики об'єкта досліджень; розбиття області на скінченні елементи за допомогою сітки вузлів; визначення базисних функцій в межах скінченних елементів; обчислювальні класи (квадратури для чисельного інтегрування); інтерполяційні функції; розв'язання систем лінійних алгебраїчних рівнянь (СЛАР); класи, орієнтовані на конкретні обчислення матричної та векторної алгебри; класи збереження, введення та виведення даних; інтерфейс користувача.

Структура простої об'єктно-орієнтованої програми методу скінченних елементів наведена на рисунку 1.

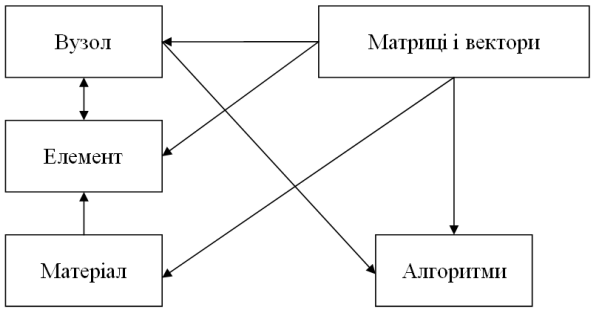

Рис. 1. Структура об'єктно-орієнтованої програми

Введемо наступні основні класи МСЕ: (*TUNode*) (Вузол), *Element* (Елемент), *Material* (Матеріал) і *Property* (Властивості). Нижче буде приведено опис їх функціонування у шаблоні ООП, їх основні атрибути і методи. Будуть визначені класи *List* і алгебраїчні класи (*Matrix*) Матриця і (*Vector*) Вектор.

Клас (*TUNode*) *в*ідіграє подвійну роль: з одного боку це точка спряження елементів (геометричний вузол), з іншого це інтерполяційний вузол, що відповідає за роботу з невідомими, розв'язуваної задачі, які називаються степенями свободи – *dof* і навантаженнями – *load*. Для того, щоб в глобальній системі число рівнянь дорівнювало числу невідомих необхідно в кожному вузлі задати один елемент з дуальної пари (*dof*, *load*). Ця дуальність впливає на вирішення глобальної системи рівнянь, тому що задане *dof* вводить невідоме початкове навантаження, тому глобальна змінна *dof* і глобальний вектор навантаження одночасно можуть містити невідомі значення. Пропонується зазначену дуальність враховувати на рівні методів вирішення так, щоб метод *Matrix_solve* одночасно обчислював невідомі складові степенів свободи і реакцій. *Node* повинен передавати задані *dof* і *load* у глобальну систему рівнянь і довизначати невідомі значення, отримані в результаті її вирішення.

Клас *Element* містить інформацію про розподілені параметри задачі. Його функції змінюються залежно від використання на попередній або завершальній фазі опрацювання інформації. В процесі попереднього опрацювання *Element* акумулює властивості, такі як геометрія та фізична поведінка, обираючи їх з поміж інших класів для подальшого опрацювання дискретних форм зображення вихідної неперервної задачі. Зазвичай, для стаціонарних задач – це матриці жорсткості елементів і вектори узагальненого навантаження, які повинні бути зібрані в глобальну систему рівнянь. Клас *Material* описує фізичні параметри моделі, що використовуються класом *Element*. Його задача полягає в керуванні параметрами таким чином, щоб *Element* в будь-який час був здатний обчислити елементи матриці жорсткості, тензор напруження, вектор зміщення і т. д. Клас *Property* вводиться для визначення факторів, котрі формально не належать до опису матеріальних властивостей середовища (матеріалу). Зазвичай, це різноманітні параметри, що належать до виду об'єкта, що моделюється. До них можуть належати, наприклад, товщина, площа поперечного перерізу, моменти інерції, і т. д. На запит ці параметри передаються класу *Element*.

Принципова структура об'єктно-орієнтованої програми методу скінченних елементів наведена на рисунку 2.

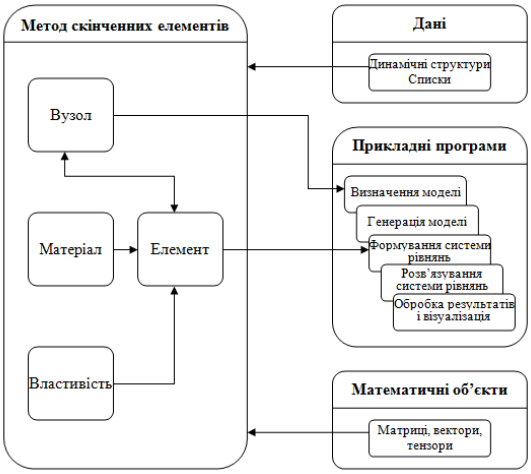

Рис. 2. Структура об'єктно-орієнтованої програми методу скінченних елементів

МСЕ завжди призводить до розрідженої матриці – з великою кількістю нулів поза головною діагоналлю. Зберігання нульових елементів різко знизило б ефективність методу. Тому конкретна реалізація МСЕ зазвичай використовує спеціальну структуру зберігання матричних елементів, мінімізуючи або зовсім виключаючи з неї нульові елементи [6]. В даній програмі для зберігання індексів ненульових елементів використовується уявлення розрідженої матриці у вигляді графа.

Слід зазначити, що доцільно, щоб вузли і елементи сітки мали «близьку» нумерацію. При цьому значно прискорюється опрацювання масивів, пошук потрібних вузлів та елементів, суттєво підвищується ефективність роботи постпроцесора, особливо в разі нестачі оперативної пам'яті і використання файлів підкачки. Для цього вихідна звичайно-елементна сітка має бути піддана процедурі перенумерації (перевпорядкування) вузлів та елементів.

Спосіб зберігання зв'язаних даних. Алгоритм методу скінченних елементів передбачає, що скінченні елементи та базисні функції взаємопов'язані. При програмуванні цих двох сутностей як класів між ними можна поставити зв'язок узагальнення: скінченний елемент (TUElem) наслідує порядок, що зберігається у класі базисних функцій (TUInterpolation) (рис. 3).

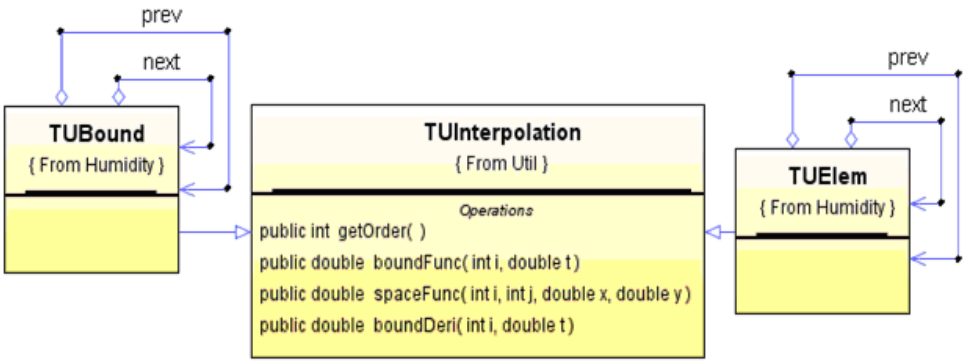

Рис. 3. Зв'язок узагальнення між базисною функцією та скінченним елементом

Скінченні елементи – це основа для обчислення подвійних інтегралів, що входять у варіаційні формулювання задачі тепломасоперенесення та задачі визначення напружено-деформівного стану [7,8]. Для обчислення одинарних інтегралів по границі області, що входять у співвідношення, визначений клас граничний елемент (TUBound), який також наслідує клас базисних функцій (TUInterpolation) (рис. 3).

Скінченні і граничні елементи будуються на певній сітці вузлів, якою покривається область задачі. Постає питання, як реалізувати зв'язок між елементом та вузлами, на яких він побудований. Відомі програмні реалізації методу скінченних елементів [4] передбачають використання масивів різних розмірностей. Зокрема перший масив зберігав усі вузли (їхні номери, координати, тощо), другий встановлював відповідність номерів скінченних елементів номерам вузлів, що їм належать, третій зберігав номери вузлів, що знаходяться на границі. Це вимагало значних зусиль при написанні та відлагодженні програмного коду.

В даній програмній реалізації МСЕ для збереження усієї множини вузлів, які накладаються на область задачі, використано двозв'язний замкнений список. Кожен вузол (*TUNode*) містить вказівник на попередній та на наступний вузли. Крім того шукані вузлові значення виділено в окрему структуру (*TUValues*). Для задачі тепломасоперенесення у неї входять температура та вологість у певний момент часу. Таким чином, кожен вузол (*TUNode*) – це точка (*Point*) з певними координатами, який містить вказівники на попередній та наступний вузли та масив структур вузлових значень (*TUValue*) для різних моментів часу (рис.4).

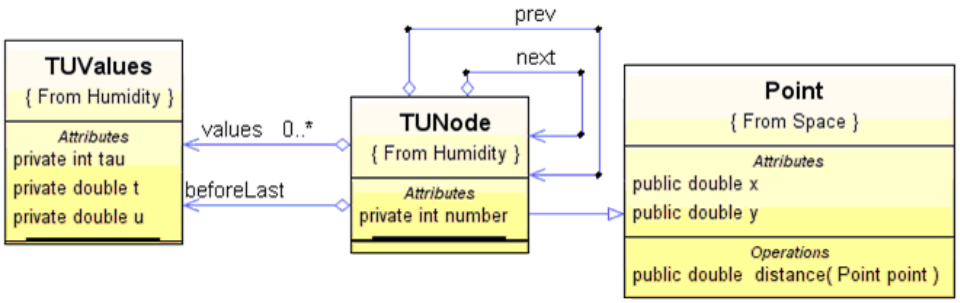

Рис. 4. Відношення між класами, що зберігають інформацію про вузли і вузлові значення

Всі об'єкти МСЕ-моделі зберігаються у зв'язаному списку, котрий визначається шаблоном, класом *List*. Саме там задаються підсписки *Node_List*, *Element_List*, *Material_List* і *Propert_List*. В них даними є адреси об'єктів визначеного типу, а методи, що наприклад, належать до класу *Element*, застосовуються до записів підсписку *Element_list*, що дає можливість у зв'язаному списку *List* імітувати традиційні методи циклічної обробки типу *for – do*, *while – do*, і *for – in*, що є стилем, який покращує зручність читання програми, наприклад, в модулі відповідальному за обчислення і формування глобальної матриці жорсткості. Її асемблування здійснюється циклом по всіх об'єктах (*elem*) класу *Element*, що перебуває в списку *Element_List* – (*ellist*), що може бути представлено на псевдокоді наступним способом:

```
For elem from ellist.start() to ellist.end do
        elem.assm_stiffness(K)
od,
```

де *start* повертає перший запис в списку. Символ *ellist.end* не є в дійсності методом, він вводиться лише для формалізації запису.

Розроблення структури класів та діаграми класів. Приступаючи до побудови архітектури програмної системи на мові *Java* з використанням середовища *IDE Eclipse Helios* необхідно визначити основні абстракції.

Точка входу в програму знаходиться в класі *Main*, в якому знаходиться функція *main*(). Звідси запускається вікно для введення даних – фрейм *InputFrame*, який успадковується від системного класу-вікна *JFrame* (рис. 5). На формі знаходяться поля для введення даних, які представлені спеціально розробленим класом *DoubleTextField*, для введення числових даних. Клас успадкований від *JTextField*. Основним методом в цьому класі є *getValue*(), який повертає введене значення типу *double*. Для того щоб компонент працював тільки з числовими значеннями перевантажено метод *createDefaultModel*(), який повертає об'єкт типу *DoubleTextDocument* [101, 124].

Після завершення введення даних відбувається ініціалізація класу для їх зберігання – *InputDataHolder* – це сінглтон, який створюється тільки один раз протягом одного сеансу використання програми і забезпечує доступ до вхідних даних в будь-якій точці. За створення цього класу відповідає метод *getInstance*():

```
private static InputDataHolder instance = null;
public static InputDataHolder getInstance()
    {
        if(instance == null){
            synchronized (InputDataHolder.class)
            {
                if(instance == null){
                    instance = new InputDataHolder();
                }
```

```
            }
        }
        return instance;
    }
```

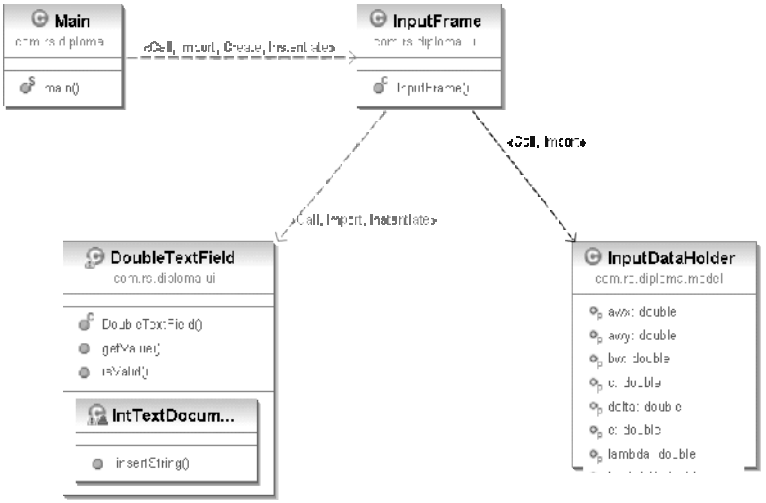

Рис. 5. Діаграма класів "Введення даних"

Розпочнемо декомпозицію системи з виділення абстракцій, які відповідатимуть за паралельні обчислення, це – об'єкт, який відповідатиме за тріангуляцію (*Triangulator*), потік для обчислень значень температури (*TFECalculator*), потік для обчислень вологості (*WFECalculator*).

Окремий потік (*Thread*) представляє собою клас для здійснення тріангуляції – розбиття на трикутники – *Triangulator* (рис. 6). Він успадковується він системного класу *Thread* і реалізовує інтерфейс *Runnable*. Відповідно до концепції багатопотоковості в *Java* основні операції виконуються в методі *run*(). Для доступу до початкових даних тут використовується клас-сінглтон *InputDataHolder*, в якому зберігаються вхідні дані програми. Для сповіщення інших об'єктів в програмі про закінчення процесу розбиття використовується інтерфейс *OnTriangulationListener*. В ньому викликається метод *onTriangulationFinished*() відразу після закінчення процесу.

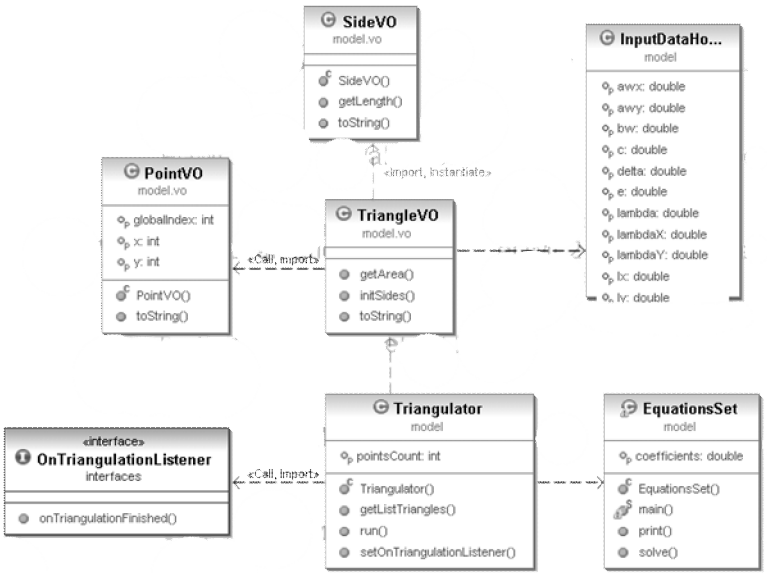

Рис. 6. Діаграма класів "Тріангуляція"

Для представлення простих абстракцій, таких як трикутник, точка, сторона та інших використано поняття об'єкта-значення (*value object*). Найпростішим з цих об'єктів є точка – *PointVO*, яка має дві координати (*x*, *y*) та глобальний індекс. Об'єкт *SideVO* описує одну сторону трикутника і будується на основі двох точок і має метод для обчислення довжини сторони.

Трикутник (*TriangleVO*) використовується в моделі для логічного представлення трикутника в програмі. Він, відповідно, складається з трьох точок і трьох сторін, також має метод для обчислення площі. Тут сторони обчислюються при початковій ініціалізації самого трикутника і ніде в програмі не перераховуються для збільшення швидкодії системи.

Клас *EquationSet* використовується для розв'язання системи рівнянь отриманих комбінуванням результатів тріангуляції і вхідних параметрів. Основним методом тут є *solve*(), який фактично є проксі-методом, що переадресовує запит до рекурсивної функції розв'язання системи, а потім повертає результат. Так зроблено для зручності введення рекурсії. Метод *solve*() звертається до методу *simplify*(), який спочатку спрощує систему, а потім виконує основні рекурсивні обчислення.

Обчислення температури і вологості матеріалу відбувається в двох основних класах – *WFETriangulator* (вологість) і *TFETriangulator* (температура) (рис. 7). Ці класи є потоками, які виконуються паралельно і взаємодіють між собою після кожної ітерації. Всі операції виконуються в методі run(), який включений в інтерфейс *Runnable*. Як глобальні члени класу винесені матриці *C*, *K*, і *F*. Результуючу матрицю можна дістати викликавши метод *getResultMatrix*(), який повертає масив результатів. Оскільки класи виконують велику кількість операцій з матрицями, то найпоширеніші з них доцільно було винести у клас-утиліту *MatrixUtils*, який і використовується при обчисленнях. Вхідні дані для обчислень потоки беруть в класі *InputDataHolder*.

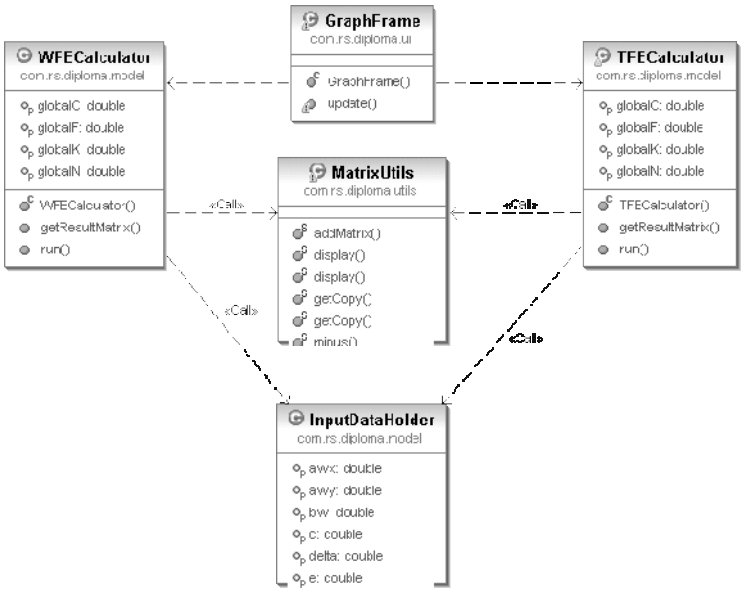

Рис. 7. Діаграма класів "Обчислення температури і вологості"

Спілкування між двома потоками забезпечує механізм синхронізації в *Java* заснований на методах *wait*, *notify* і *notifyAll*. Ці методи реалізовані, як *final*-методи класу *Object*, так що вони є в будь-якому *Java*-класі. Всі ці методи повинні викликатися тільки з синхронізованих методів.

Потік, який чекає виконання будь-яких умов, викликає у цього об'єкта метод *wait*, попередньо захопивши його монітор. На цьому його робота припиняється. Інший потік може викликати на цьому ж самому об'єкті метод *notify* (знову ж, попередньо захопивши монітор об'єкта), в результаті чого, чекає на об'єкті потік "прокидається" і продовжує своє виконання.

Об'єктом для відображення результатів розрахунків служить вікно *GraphFrame*, яке успадковане від системного класу *JFrame* (рис. 8). На його панелі можна побачити відображений зразок розбитий на трикутники.

В програмі можна виводити графіки двох типів: трьохвимірні і двохвимірні. Ця можливість реалізована з допомогою Java бібліотек: *jzy3D* і *JFreeChart* з відкритим кодом. Всі вікна для відображення графіків викликаються з класу *GraphFrame*.

JFreeChart є бібліотекою з відкритим кодом і розповсюджується під ліцензією *GNU Lesser General Public Licence* (*LGPL*), що дозволяє використовувати продукт в усіх додатках і включає такі функції: добре документований *API*, який підтримує велику кількість типів графіків; гнучкий дизайн, який просто впровадити; підтримка багатьох вихідних форматів, включаючи *Swing*-

компоненти, графічні файли (включаючи *PNG* і *JPEG*) формати векторної графіки (включаючи *PDF*, *EPS* і *SVG*).

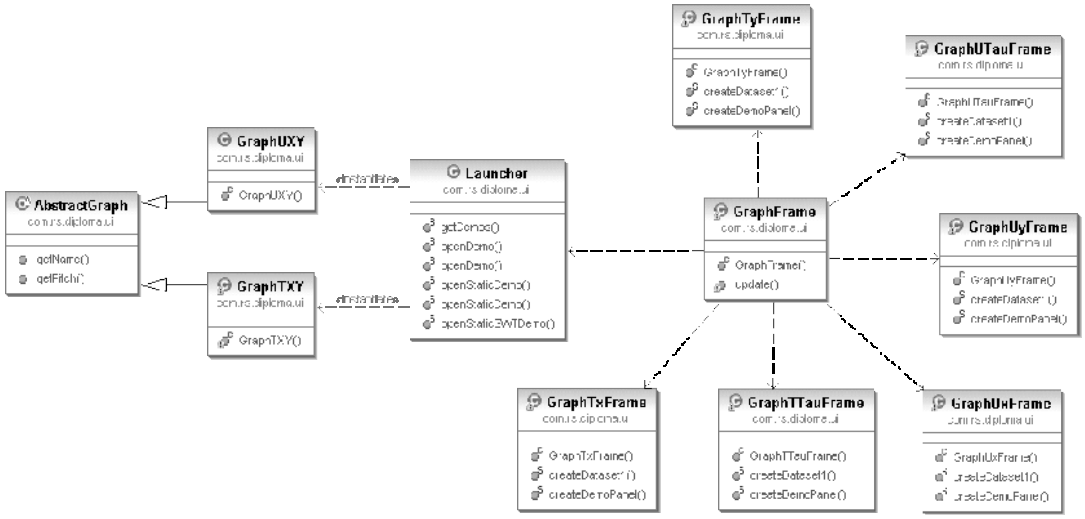

Рис. 8. Діаграма класів "Відображення графіків"

Для запуску вікон з трьохвимірних графіків в програмі використовується клас *Launcher*. Запуск нового вікна здійснюється зі статичного методу *openDemo*() в який потрібно передати об'єкт, що реалізовує інтерфейс *IFrame*. З його допомогою можна запустити вікна *GraphUXY* і *GraphTXY*, які попередньо мають бути визначені в методі *getDemos*():

```
public static List<IFrame> getDemos() throws IOException
{
  List<IFrame> demos = new ArrayList<IFrame>();
  demos.add(new GraphUXY());
  demos.add(new GraphTXY());
  return demos;
}
```

З метою узагальнення, класи *GraphUXY* і *GraphTXY* успадковуються від класу *AbstractGraph*, який реалізовує інтерфейс *IFrame*. В цих класах всі дані для побудови графіків формуються в класі *getCoordinates*(), який повертає список (*List*).

Для формування двовимірних графіків використовується базовий клас *ApplicationFrame*, який наслідують всі класи, що відповідають за відображення графіків. Набір даних для відображення тут формується в методі *createDataset()*.

Дослідження адекватності та верифікації математичних моделей. Для дослідження адекватності математичної моделі тепломасоперенесення [7,8] проведені порівняння результатів чисельного моделювання температурно-вологісних полів з незалежними теоретичними дослідженнями для часткових випадків, а також здійснено порівняння результатів моделювання з відомим експериментальними дослідженнями, виконаними для конкретних умов. В основі таких досліджень покладені окремі математичні моделі перенесення тепла і вологи зі сталими коефіцієнтами. Експериментальні дослідження для деревних взірців таких розмірів деревини бука наведені у працях [9,10]. Згідно них, в процесі експерименту зовнішні умови агента сушіння приймалися такими: t_c = 80 *℃*, φ = 60 *%*, v = 1 *м/с*; t_c = 70 *℃*, φ = 50 *%*, v = 2 *м/с*; t_c = 50 *℃*, φ = 40 *%*, v = 3 *м/с*. Відповідно, для першого випадку зміни (t_c, φ, v) теплофізичні параметри матеріалу є такі: ρ = 734 *кг/м*^{3}; C = 2,42 *кДж/(кг·К)*; коефіцієнти вологопровідності a_{mx} = 0,0106 *см*^{2}*/с*; a_{my} = 0,0148 *см*^{2}*/с*; ε = 0,15; критерій Косовича Ko = 5,05; критерії Ликова Lu_x = 0,0109; Lu_y = 0,0114; критерії Кірпічова Ki_{tx} = 0,0697; Ki_{ty} = 0,026; Ki_{ux} = 0,0077; Ki_{uy} = 0,00276; критерій Поснова Pn = 59,02. Для розрахунків у другому випадку зміни параметрів середовища (t_c, φ, v) теплофізичні параметри вибиралися: ρ = 718 *кг/м*^{3}; C = 2,28 *кДж/(кг·К)*; a_{mx} = 0,016 *см*^{2}*/с*; a_{my} = 0,022 *см*^{2}*/с*; ε = 0,15; Ko = 5,285; Lu_x = 0,0904; Lu_y = 0,1080; Ki_{tx} = 0,736; Ki_{ty} = 0,309; Ki_{ux} = 0,00567; Ki_{uy} = 0,00206; Pn = 56,43.

На рис. 9 та рис. 10 наведені результати порівняння розподілу температурно-вологісних полів з експериментальними дослідженнями [10,11] та даними чисельного експерименту з

математичними моделями зі сталими теплофізичними параметрами [12]. За критерій розходження між результатами використаний коефіцієнт множинної кореляції [13], який використовується для порівняння розрахункових та експериментальних даних перехідних процесів. Даний критерій визначає, чи існує кореляційний зв'язок між експериментальними та розрахунковими даними та має вигляд:

$$|t| = \frac{\left|\rho_{y\bar{y}}\right|}{\sqrt{1-\rho_{y\bar{y}}^2}}\sqrt{n-2}\,, \tag{1}$$

де n – кількість точок, для яких проводить порівняння величин. Даний критерій дозволяє встановити ступінь кореляційного зв'язку між величинами. Якщо виконується умова $|t| < t_\alpha$, то з ймовірністю $p = 1-\alpha$ можна стверджувати, що між величинами σ_{ij} та $\bar{\sigma}_{ij}$ існує «тісний» кореляційний зв'язок. У протилежному випадку з ймовірністю $p = 1-\alpha$ гіпотеза про такий зв'язок відкидається.

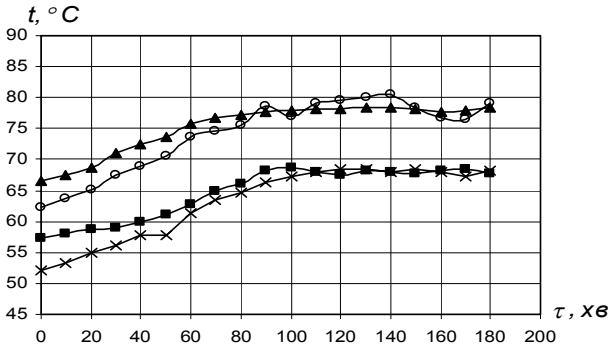

Рис. 9. Залежності розподілу температури у часі для t_c = 70 °C, φ = 50 %, v = 2 *м/с* (■, ▲ – експериментальні дані [11,12]; x, ○ – розрахункові значення за математичною моделлю)

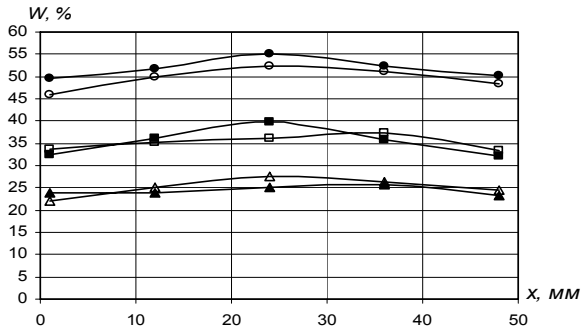

Рис. 10. Залежності розподілу вологості за товщиною пластини для t_c = 70 °C, φ = 50 %, v = 2 *м/с* (●, ■, ▲ – експериментальні дані [11,12] для τ = 30 *хв*, τ = 120 *хв*, τ = 120 *хв*; ○, □, Δ – розрахункові значення за математичною моделлю)

Коефіцієнт кореляції $\rho_{y\bar{y}}$ між розрахунковими (y) та експериментальними ($\bar{y}$) величинами визначається за відношенням:

$$\rho_{y\bar{y}} = \frac{\sum_{i=1}^{n}\left(y_i - y_{сер}\right)\left(\bar{y}_i - \bar{y}_{сер}\right)}{\sqrt{\sum_{i=1}^{n}\left(y_i - y_{сер}\right)^2 \sum_{i=1}^{n}\left(\bar{y}_i - \bar{y}_{сер}\right)^2}}, \tag{2}$$

де $y_{сер} = \frac{1}{n}\sum_{i=1}^{n} y_i$; $\bar{y}_{сер} = \frac{1}{n}\sum_{i=1}^{n} \bar{y}_i$.

Для верифікації математичної моделі використовувалися експериментальні дані [9,14]. Зокрема отримано, що для (t_c = 70 °C, φ = 50 %, v = 2 *м/с*; ρ = 720 *кг/м*^{3}) значення критерія і

коефіцієнта кореляції для розподілу температури такі: $\rho_{y\bar{y}} = 0{,}993$; $|t| = 11{,}58$; для (t_c = 80 *℃*, φ = 60 *%*, v = 1 *м/с*; ρ = 735 *кг/м*^{3}) маємо $\rho_{y\bar{y}} = 0{,}995$; $|t| = 14{,}083$. Аналогічно для (t_c = 70 *℃*, φ = 50 *%*, v = 2 *м/с*; ρ = 720 *кг/м*^{3}) значення критерія і коефіцієнта кореляції для розподілу вологості за товщиною дошки для моменту часу τ = 30 *хв* такі: $\rho_{y\bar{y}} = 0{,}992$; $|t| = 11{,}48$; для моменту часу τ = 120 *хв*: $\rho_{y\bar{y}} = 0{,}998$; $|t| = 22{,}28$.

Затабульоване t для температурного поля для n = 17, числі степеней вільності $f = n - 2 = 15$ і рівні значимості складає $t_{таб} = 4{,}48$. Оскільки умова $|t| \geq t_{таб}$ виконується для двох випадків, то з ймовірністю похибки $\alpha = 0{,}05$ (5*%*) можна стверджувати про адекватність характеру зміни наведених даних результатів чисельного моделювання температурних полів та їх експериментальних значень. Аналогічно для розподілу вологості ($n = 5, f = 3$) на рівні значимості $\alpha = 0{,}05$ маємо $t_{таб} = 4{,}27$. Оскільки, для двох значень часу (τ = 30 *хв*, τ = 120 *хв*) виконується умова $|t| \geq t_{таб}$, то з ймовірністю похибки можна стверджувати про адекватність чисельного моделювання вологісних полів з результатами експериментальних досліджень.

Тому перевіримо адекватність моделювання напружено-деформівного стану.

Порівняємо результати моделювання розвитку напружень у пружній та в'язкопружній областях.

У роботі [15,16] для різних моментів часу обчислено розподіл напружень у пружній області деформування у процесі сушіння. Обчислення здійснено для соснових дошок товщиною 20 см при таких параметрах середовища сушіння: температура T_c =88 °C, відносна вологість φ =55 %, швидкість руху агента сушіння φ =55 м/с. Із порівняння розподілів напружень у центрі вказаного матеріалу вздовж його товщини легко бачити, що характер напружень однаковий (рис. 11). Щодо відхилення отриманих чисельних результатів від наведених у [15,16], то максимальне відхилення становить не більше 7,9 % після перших трьох годин сушіння і спадає у процесі сушіння.

Інший спосіб показати адекватність результатів – порівняти чисельні результати в'язкопружного деформування деревини, розраховані для двовимірної задачі, із такими ж результатами, обчисленими іншими авторами для одновимірної задачі. Наприклад, у роботі [17] наведені результати розвитку напружень у деревині сосни розміром 20 мм під час сушіння при таких параметрах середовища сушіння: температура T_c =88 °C, відносна вологість φ =55 %, швидкість руху агента сушіння v =2 м/с. Із рис. 11 видно, що характер розвитку напружень однаковий, відхилення у перші години процесу сушіння досить значне, але не перевищує 12,4 %, і у процесі сушіння спадає.

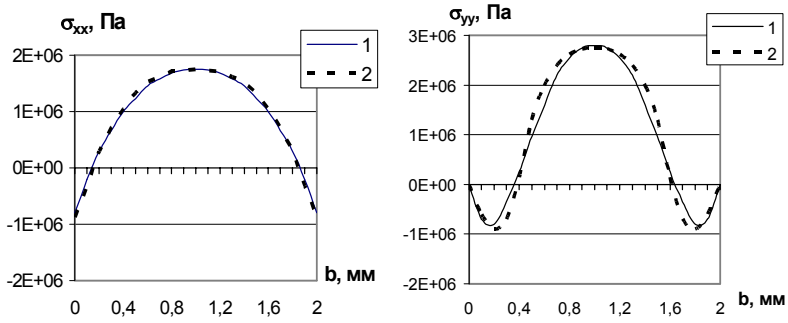

Рис. 11. Розподіл напружень (а) і (б) в центрі матеріалу вздовж товщини деревини сосни розміром 20×20 мм через 2,6 год процесу сушіння за методами: 1 – Рітца; 2 – МСЕ

Висновки. Розроблено прикладну програму чисельного розв'язування двовимірної математичної моделі,. При програмуванні використано об'єктно-орієнтований підхід, який дозволив реалізувати програму у вигляді множини об'єктів та класів, що взаємодіють між собою. Такі об'єкти як скінченний елемент, вузол, квадратура, апроксимаційна функція, СЛАР, тощо були реалізовані у вигляді окремих класів.

Описано за допомогою діаграм UML основні етапи роботи прикладної програми. Зокрема, на діаграмах класів представлено основні класи, які реалізують роботу програми, відношення та

зв'язки між ними. На діаграмах послідовності відображено впорядковану в часі взаємодію об'єктів під час виконання різних етапів обчислень.

Здійснено апробацію та верифікацію результатів роботи програми шляхом їхнього порівняння із наявними експериментальними даними та порівняння часткових випадків із відомими результатами.

Розроблене програмне забезпечення мовою Java і спроектовані діаграми UML та алгоритми реалізації методу скінченних елементів можуть бути використані для проектування систем автоматизованого моделювання та аналізу температурно-вологісних полів у капілярно-пористих матеріалах залежно від характеристик деревини.

ЛІТЕРАТУРА

1. **Конисов С.П.** Обьектно-ориентированный метод декомпозиции области / С.П. Конисов, И.В. Красноперов, В.Н. Рычков // Вычислительные методы и программирование. – 2003. – Т.4. – №1. – С. 176-193.
2. **Савула Я.Г.** Числовий аналіз задач математичної фізики варіаційними методами. – Львів: Вид-во ЛНУ«ім.Франка». – 2004. – 222 с.
3. **Соколовський Я.І.** Автоматизація процесів моделювання деформаційно-релаксаційних і тепломасообмінних процесів у капілярно-пористих матеріалах / Я.І. Соколовський, О.В. Мокрицька, І.М. Крошний, І.Д. Капран, А.П. Здолбіцький // Системний аналіз та інформаційні технології: матеріали Міжнародної науково-технічної конференції SAIT 2011. – Київ: ННК "ІПСА" НТУУ "КПІ". – 2011. – С. 159.
4. **Zimmermann T.** Object-orient finite element programming: I vgoverment principles / T. Zimmermann, Y. Duboes-Pelerin, P. Bomme // Computing Methods in Applied Mechanics and Engeeneering. – 1992. – Vol. 98 (no. 2). – P. 291-303.
5. **Rumbaugh J.** The unified modeling language: user guide reading / J. Rumbaugh, G. Booch // Addison-Weslyy. – 1999. – 346 p.
6. **Сегерлинд Л.** Применение метода конечних елементов. – М.: Мир. – 1979. – 378 с.
7. Ya.I. Sokolowskyy, M.B. Denjuk, A.B. Bakalets, A.P. Zdolbitskyy. Mathematical modeling of stressed-strained relaxation fields in wood drying process. // Наукові нотатки. Міжвузівський збірник (за галузями знань „Машинобудування та металообробка", „Інженерна механіка", „Металугрія та матеріалознавство"). – Луцьк, 2010. – Вип. 27. – С. 4-10.
8. **Соколовський Я.І.** Математична модель деформаційно-релаксаційних процесів у капілярно-пористих матеріалах з параметрами внутрішнього і зовнішнього тепломасоперенесення / Я.І. Соколовський, І.М. Крошний // Вісник Національного університету "Львівська політехніка": Комп'ютерні науки та інформаційні технології. – Львів: НУ "Львівська політехніка". – 2011. – Вип. 710. – С. 274-279.
9. **Шубин Г.С.** Сушка и тепловая обработка древесины. – М.: Лесная промышленность. – 1990. – 336 с.
10. **Perre P.** A physical and mechanical model able to predict the stress field in wood over a wide range of drying conditions / P. Perre, J. Passard // Drying Technology. – 2004. – Vol. 22 (no. 1-2). – P. 27-44.
11. **Salin J.-G.** Numerical prediction of checking during timber drying and a new mechano-soorptive creep model // Holz Roh- Werkstoff. – 1992. – Vol. 50. – P. 195-200.
12. **Hakan F.** Oztop. Numerical and experimental analysis of moisture transfer for cinvective drying of someproducts / F. Oztop Hakan, K. Akpinar Ebru // International Communications in Heat and Mass Transfer. – 2008. – Vol. 35 (Issue 2). – P. 169-177.
13. **Томашевский В.** Имитационное моделирование в среде GPSS / В. Томашевский, Е. Жданова. – М.: Бестселлер. – 2003.– 416 с.
14. **Pang S.** Modeling of stress development during and relief during steaming in Pinus radiate lumber // Drying Technology. – 2000. – Vol. 18 (no. 8). – P. 1677-1696.
15. **Уголев Б.Н.** Деформативность древесины и напряжения при сушке. – М.: Лесная промышленность. – 1971. – 174 с.
16. **Уголев Б.Н.** Древесиноведение с основами лесного товароведения: учеб. для лесотехн. вузов / Б.Н. Уголев // М-во образования Рос. Федерации, Моск. гос. ун-т леса. – Изд. 3-е, перераб. и доп. – М.: МГУЛ. – 2002. – 340 с.
17. **Соколовський Я.І.** Технологічні напруження і деформації деревини у процесі сушіння // Науковий вісник: Зб. наук.-техн. праць. – Львів: УкрДЛТУ. – 1999. – Вип.9.3. – С. 168-176.