Міністерство освіти і науки України
Національний технічний університет
«Дніпровська політехніка»

Інститут електроенергетики
(інститут)

Факультет інформаційних технологій
(факультет)

Кафедра Програмного забезпечення комп'ютерних систем
(повна назва)

ПОЯСНЮВАЛЬНА ЗАПИСКА
кваліфікаційної роботи ступеня

магістра
(назва освітньо-кваліфікаційного рівня)

студента *Яриловця Віктора Ігоровича*
(ПІБ)

академічної групи *122м-19-1*
(шифр)

напряму підготовки *122 Комп'ютерні науки*
(код і назва напряму підготовки)

на тему: *Методи, алгоритми та інформаційна технологія розрахунку і візуалізації температурних полів*

_____ *В.І. Яриловець*

Керівники	**Прізвище, ініціали**	**Оцінка за шкалою**		**Підпис**
		рейтинговою	**інституційною**	
розділів кваліфікаційної роботи				
спеціальний	Проф. Мещеряков Л.І.			
економічний	Доц. Касьяненко Л.В.			

Рецензент				

Нормоконтролер	Доц. Сироткіна О.І.			

Дніпро
2020

Міністерство освіти і науки України
Національний технічний університет
«Дніпровська політехніка»

ЗАТВЕРДЖЕНО:
Завідувач кафедри
Програмного забезпечення комп'ютерних систем
(повна назва)

_____ І.М. Удовик
(підпис) (прізвище, ініціали)

« » _____ 20 20 Року

ЗАВДАННЯ
на виконання кваліфікаційної роботи магістра

спеціальності *122 Комп'ютерні науки*
(код і назва спеціальності)

студенту *122М-19-1* *Яриловцю Віктору Ігоровичу*
(група) (прізвище та ініціали)

Тема кваліфікаційної роботи *Методи, алгоритми та інформаційна технологія розрахунку і візуалізації температурних полів*

1 ПІДСТАВИ ДЛЯ ПРОВЕДЕННЯ РОБОТИ

Наказ ректора НТУ «Дніпровська політехніка» від 22.10.2020 р. № 888-с

2 МЕТА ТА ВИХІДНІ ДАНІ ДЛЯ ПРОВЕДЕННЯ РОБІТ

Об'єкт досліджень – процес візуалізації теплових полів на мові програмування C#.

Предмет досліджень – методи, моделі та засоби моделювання та візуалізації теплових процесів.

Мета роботи – підвищення швидкості та точності виконання програм з моделювання та візуалізації температурних полів.

3 ОЧІКУВАНІ НАУКОВІ РЕЗУЛЬТАТИ

Наукова новизна результатів роботи полягає в удосконалення методів та алгоритмів моделювання та візуалізації теплових полів.

Практична цінність полягає у тому, що результати роботи, отримані в ході дослідження, можуть застосовуватися як в розробці програм для візуалізації теплових полів, так і в практиці викладання дисциплін, пов'язаних з моделюванням теплових процесів.

4 ЕТАПИ ВИКОНАННЯ РОБІТ

Найменування етапів робіт	**Строки виконання робіт (початок – кінець)**
Аналіз теми та постановка задачі	10.09.2020-30.09.2020
Пошук оптимальних шляхів вирішення задачі моделювання теплових процесів засобами .NET Framework.	01.10.2020-31.10.2020
Створення на базі існуючих алгоритмів та методів програмного забезпечення для візуалізації теплових полів для удосконаленні моделювання процесів теплообміну	01.11.2020-16.12.2020

Завдання видав _____ проф. Мещеряков Л.І.
(підпис) (посада, прізвище, ініціали)

Завдання прийняв до виконання _____ *Яриловець В.І.*
(підпис) (прізвище, ініціали)

Дата видачі завдання: *10.09.2020 р.*

Термін подання кваліфікаційної роботи до ЕК ***17.12.2020 р.***

РЕФЕРАТ

Пояснювальна записка: 84 с., 10 мал., 3 додатка, 52 джерела.

Об'єкт дослідження: процес візуалізації теплових полів на мові програмування C#.

Предмет дослідження: методи, моделі та засоби моделювання та візуалізації теплових полів.

Мета магістерської роботи: підвищення швидкості та точності виконання програм з моделювання та візуалізації температурних полів.

Методи дослідження. Для виконання поставлених завдань були використані наступні методи та інструменти:

- методи теорії теплопровідності;
- вирішення системи лінійних алгебраїчних рівнянь (СЛАР) методом кінцевих різниць;
- Visual Studio, C#, .NET Framework, OpenGL, Tao Framework.

Наукова новизна полягає в удосконаленні методів та алгоритмів моделювання та візуалізації теплових полів.

Практичне значення роботи. Результати роботи, отримані в ході дослідження, можуть застосовуватися як в розробці програм для візуалізації теплових полів, так і в практиці викладання дисциплін, пов’язаних з моделюванням теплових процесів.

У розділі «Економіка» визначена трудомісткість розробленої інформаційної системи, проведений підрахунок вартості роботи по створенню програми та розраховано час на його створення.

Список ключових слів: програмне забезпечення, теплообмін, теплове поле, термограма, теплопровідність, візуалізація, C#, OpenGL, Tao Framework, .NET Framework.

ABSTRACT

Explanatory note: 84 pages, 10 figures, 3 applications, 52 sources.

Object of research: process of visualization of thermal fields on C#.

Subject of research: methods, models and tools of modeling and visualization of thermal fields.

Purpose of Master's thesis: increasing the speed and accuracy of programs for modeling and visualization of temperature fields.

Research methods. The following methods and tools were used to perform the tasks:

- methods of thermal conductivity theory;
- solving a system of linear algebraic equations by the finite difference method;
- Visual Studio, C#, .NET Framework, OpenGL, Tao Framework.

Originality of research is in the improvement of methods and algorithms for modeling and visualization of thermal fields.

Practical value of the results. The results of the research can be used both in the development of programs for the visualization of thermal fields, and in the practice of teaching disciplines related to the modeling of thermal processes.

In the Economics section we calculated the complexity of the developed information system, the cost of work to create a program and the time for its creation.

Keywords: program software, heat transfer, thermal field, termogram, heat conductivity, visualization, C#, OpenGL, Tao Framework, .NET Framework.

ЗМІСТ

СПИСОК УМОВНИХ ПОЗНАЧЕНЬ

UML – Unified Modeling Language;

ПК – персональний комп’ютер;

ЕОМ – електронно-обчислювальна машина;

API – Application Programming Interface (програмний інтерфейс програми);

ПЗ – програмне забеспечення;

HDD – Hard Disk Drive (жорский диск);

МКР – метод кінцевих різниць;

ОС – операційна система;

XML – Extensible Markup Language;

LINQ – Language Integrated Query;

OpenGL – Open Graphics Library;

СЛАР – система лінійних алгебраїчних рівнянь;

grad F – градієнт функції F;

ОК – обчислювальний комплекс;

ІД – індекс прибутковості;

NPU – Net Present Value (чиста поточна вартість).

ВСТУП

Актуальність дослідження. За останні десятиліття сфера інтенсивного дослідження та застосування явищ теплообміну надзвичайно розширилася. В даний час спостерігається значний прогрес в розвитку обчислювальних методів вирішення завдань для рівнянь в приватних похідних і збільшення потужності сучасних обчислювальних машин. Виходячи з цього, теоретичне дослідження процесів теплообміну в даний час в значній мірі базується на їх чисельному моделюванні з використанням ЕОМ.

Моделювання теплових полів є однією з найбільш актуальних проблем при проектуванні електронних компонентів. Візуалізація теплових полів, тобто отримання термограми, являє собою заключну частину моделювання теплових процесів. Тобто при створенні спеціальних засобів розрахунку теплових параметрів і моделювання теплових процесів, розробник зіткнеться з завданням візуалізації теплових полів. Тому розробка в даній сфері буде актуальною ще довгий час.

Мета дослідження полягає в підвищенні швидкості та точності виконання програм з моделювання та візуалізації температурних полів.

Завдання дослідження. Для досягнення поставленої мети в роботі сформульовані і вирішені такі завдання:

1. Викласти принципи процесів теплообміну;
2. Дослідити особливості реалізації математичної моделі теплових процесів;
3. Проаналізувати інструменти моделювання процесів теплообміну, які доступні в MS Visual Studio;
4. Спроектувати та розробити відповідне програмне забезпечення, яке можна застосовувати на різних конфігураціях ОС;
5. Зробити висновки щодо доцільності створення подібної системи.

Об’єкт дослідження: процес візуалізації теплових полів на мові програмування C#.

Предмет дослідження: методи, моделі та засоби моделювання та візуалізації теплових полів.

Методи дослідження. Для виконання поставлених завдань були використані наступні методи та інструменти:

- методи теорії теплопровідності;
- вирішення системи лінійних алгебраїчних рівнянь методом кінцевих різниць;
- Visual Studio, C#, .Net Framework, OpenGL, Tao Framework.

Наукова новизна полягає в удосконалення методів та алгоритмів моделювання та візуалізації теплових полів.

Практичне значення. Проведене дослідження буде корисно для науковців, розробників, аспірантів і студентів, що спеціалізуються або цікавляться проблематикою дослідження та моделювання процесів теплообміну. Результати дослідження можуть застосовуватися як в практиці викладання, так і для організації самостійного навчання.

Особистий внесок автора:

1. Наукові результати роботи отримані автором самостійно.

2. Вибір методів досліджень і технологій реалізації;

3. Реалізація моделювання теплових полів засобами Microsoft .NET Framework;

4. Розробка теоретичної частини роботи, в якій досліджені і систематизовані знання про існуючі підходи до розрахунку математичної моделі процесів теплообміну.

5. Оцінка отриманих результатів.

Структура і обсяг роботи

Робота складається з вступу, трьох розділів і висновків. Містить 84 сторінки, в тому числі 69 сторінок тексту основної частини з 10 рисунками, списку використаних джерел з 52 найменуваннями на 5 сторінках, 3 додатка на 10 сторінках.

РОЗДІЛ 1
МАТЕМАТИЧНЕ МОДУЛЮВАННЯ ТЕПЛОВИХ ПОЛІВ

1.1. Математичне модулювання

Сучасний етап розвитку науково-технічного прогресу характеризується широким використанням засобів обчислювальної техніки і чисельного моделювання. Ці обчислювальні засоби істотно розширюють можливості теоретичних досліджень. Нинішні можливості дозволяють зробити математичне моделювання інструментом об'єднання теоретичних і експериментальних досліджень в одне ціле.

1.1.1. Основи побудови математичної моделі

Моделювання широко застосовується в наукових дослідженнях і при вирішенні прикладних проблем в різних областях техніки. Ця методологія заснована на вивченні властивостей і характеристик об'єктів різної природи за допомогою дослідження їх природних або штучних аналогів. Моделювання в такому плані являє собою двоєдиний процес створення моделей і дослідження моделей після того, як вони побудовані. Використання моделей завжди і неминуче пов'язане зі спрощенням, ідеалізацією модельованого об'єкта.

Сама модель, природно, не охоплює об'єкт у всій повноті його властивостей, а відображає лише деякі його досліджувані характеристики - вона схожа з пізнаваним об'єктом тільки за певною сукупністю ознак. Модель будується для відображення лише частини властивостей досліджуваного об'єкта і тому зазвичай простіше оригіналу. І найголовніше, модель більш доступна і зручна для дослідження, ніж модельований об'єкт.

Для більш повного дослідження об'єкта залучається ряд моделей, кожна з яких моделює ті чи інші характеристики об'єкта. У прикладному дослідженні навіть для відображення одних і тих же властивостей об'єкта завжди є

можливість залучення різних моделей. Моделі розрізняються за ступенем якісної та кількісної адекватності досліджуваного об'єкта щодо обраних характеристик, за можливості їх дослідження. Успіх моделювання визначається саме вдалим вибором моделей і їх набору. Очевидно, що цей вибір суб'єктивний здебільшого і базується на всіх наявних експериментальних і теоретичних уявленнях про об'єкт, на всьому придбаному раніше досвіді моделювання.

Більшість з різних моделей можна розділити на фізичні і математичні. Фізичні моделі відносяться до матеріальних моделям, які, імітуючи частина властивостей досліджуваного об'єкта, мають ту ж природу, що і модельований об'єкт. Математичні моделі є найбільш характерними в природничо-наукових дослідженнях в якості ідеальних моделей.

При математичному моделюванні дослідження властивостей і характеристик вихідного об'єкта замінюється дослідженням його математичних моделей. Зараз такий вид моделювання характеризується широким залученням комп'ютерів, методів обчислювальної математики.

Евристична роль математичного моделювання проявляється в тому, що замість натурного експерименту проводиться математичний експеримент. Замість дослідження прояви того чи іншого параметра. Такий експеримент, доповнюючи натурний, дозволяє значно глибше дослідити явище або процес.

1.1.2. Застосування ЕОМ для математичного моделювання

Дослідження математичних моделей має на увазі перш за все якісне вивчення математичних моделей і отримання точного або наближеного рішення. ЕОМ надають нові можливості не тільки для знаходження наближеного рішення чисельними методами, але і для якісного дослідження математичної моделі.

Якісне дослідження починається з размерностного аналізу завдання. Приведення завдання до безрозмірного вигляду дозволяє скоротити число визначальних параметрів завдання. Виділення малих або великих безрозмірних параметрів дає можливість в ряді випадків істотно спростити вихідну

математичну модель, врахувати особливості завдання при розробці чисельних методів її рішення.

Сама математична модель може бути досить складною і нелінійної. Це часто робить неможливим якісне дослідження традиційними методами прикладної математики. Тому в більшості випадків проводиться якісне дослідження на більш простих і змістовних по відношенню до вихідної моделі завданнях.

Важливим елементом при якісному дослідженні математичних моделей є коректність моделі. Перш за все потрібно розглядати проблему існування рішення. Відповідні строгі результати дають впевненість в коректності математичної моделі. Крім того, конструктивні доведення теорем існування можуть бути покладені в основу наближених методів вирішення поставленого завдання.

У застосуванні ЕОМ при математичному моделюванні можна виділити два етапи. Перший з них характеризується дослідженням досить простих математичних моделей. На цьому етапі обчислювальні засоби використовуються поряд і нарівні з іншими методами прикладної математики. Можна сказати, що тоді відбувається рішення конкретної і досить вузької завдання з певним набором вхідних даних.

Другий етап характерний дослідженням складних математичних моделей. В такому випадку обчислювальні засоби стають основними, абсолютно переважаючими, а традиційні засоби прикладного математичного моделювання виконують більш допоміжну роль.

1.2. Вивчення теплових процесів

При вивченні теплових процесів виділяють три основних способи перенесення тепла:

1. Теплопровідність – передача тепла від однієї частини тіла до іншої або від одного тіла до іншого, що знаходиться в зіткненні з першим.

2. Конвекція - перенесення, обумовлений просторовим переміщенням речовини і спостерігається в рухомих середовищах, таких як рідини і гази.

3. Випромінювання - перенесення енергії у вигляді електромагнітних хвиль.

В даній роботі буде досліджено саме процес теплопровідності.

1.2.1. Температурне поле та теплопровідність

Як і будь-яке інше фізичне явище, процес теплопередачі відбувається в просторі і часі, тому вивчення зміни температури, основною характеристикою даного явища, зводиться до знаходження залежності

$$T = f(x, y, z, \tau), \tag{1.1}$$

де x, y, z - просторові координати в декартовій системі, τ - час.

Сукупність миттєвих значень температури в усіх точках досліджуваного простору називається температурним полем. Розрізняють стаціонарне і нестаціонарне температурні поля.

Стаціонарним температурним полем називається таке поле, температура якого в будь-який його точці не змінюється в часі, тобто є функцією тільки координат (сталий стан):

$$T = \Phi(x, y, z), \quad \frac{\partial T}{\partial \tau} = 0. \tag{1.2}$$

Нестаціонарним температурним полем називається таке поле, температура якого змінюється не тільки в просторі, але і з плином часу (несталий стан). Рівняння (1.1) є математичною записом нестаціонарного температурного поля, яке і буде вивчено в даній роботі.

1.2.2. Основний закон теплопровідності Фур'є

Наявність температурного градієнта є необхідною умовою поширення тепла. Досвід показує, що передача тепла теплопровідністю відбувається по нормалі до ізометричної поверхні від місць з більшою температурою до місць з меншою температурою.

Кількість тепла, що проходить в одиницю часу і віднесена до одиниці площі ізотермічної поверхні, називається щільністю теплового потоку; відповідний вектор визначається співвідношенням

$$q = (-1_n)\frac{dQ}{d\tau}\frac{1}{S}, \tag{1.3}$$

де $\frac{dQ}{d\tau}$ – кількість тепла, що проходить в одиницю часу, або швидкість теплового потоку, S – площа ізотермічної поверхні, (-1_n) – одиничний вектор, спрямований по нормалі до поверхні в сторону зменшення температури.

Основний закон теплопровідності можна сформулювати так: щільність теплового потоку прямо пропорційна напруженості температурного поля, або щільність теплового потоку прямо пропорційна градієнту температури, тобто

$$q = \lambda E = -\lambda\, grad\, T = -\lambda \nabla T = -\lambda 1_n \frac{\partial T}{\partial n}, \tag{1.4}$$

де λ – коефіцієнт пропорційності, що називається коефіцієнтом теплопровідності.

Коефіцієнт теплопровідності є фізичною характеристикою тіла щодо його здатності до теплопровідності та дорівнює кількості тепла, що протікає в одиницю часу через одиницю поверхні, при перепаді температури на одиницю довжини нормалі, що дорівнює одному градусу.

Теплопровідність в газах і парах значною мірою обумовлена переносом кінетичної енергії руху молекул, тому коефіцієнти λ для газів і парів малі.

У рідинах перенесення тепла теплопровідністю відбувається за типом поширення поздовжніх коливань аналогічно поширенню звуку. Тому тут коефіцієнти теплопровідності рідин більше, ніж у газів, так як їх молекулярна структура сприяє кращому переносу тепла.

В металах перенесення тепла теплопровідністю в значній мірі визначається переносом енергії вільними електронами. Коефіцієнт теплопровідності залежить від температури; для багатьох металів він зменшується з підвищенням температури за лінійним законом.

Виходячи із загального стану та спираючись на сукупність всіх раніше вище перелічених та згаданих фактів, для визначення кількості тепла, протікаючого через будь-яку поверхню твердого тіла, необхідно знати температурне поле всередині тіла.

1.2.3. Диференційне рівняння Фур'є-Кірхгофа

Перенесення тепла в рухомому середовищі описується диференціальним рівнянням Фур'є-Кірхгофа. Якщо знехтувати дифузійної теплопровідністю і перенесенням теплоти за рахунок дифузії, то за відсутності поля зовнішніх сил рівняння прийме такий вигляд:

$$\mathrm{c}_p \rho \frac{dT}{d\tau} = div(\lambda\, gradT) + I_q + \eta\Phi_v, \qquad (1.5)$$

де I_q - джерело теплоти, Φ_v - диссипативна функція Релея, c_p - питома ізобарна теплоємність. Для твердого тіла матеріальна похідна $dT/d\tau$ дорівнює похідної, оскільки відсутність тиску тіла ($\vec{v} = 0$), а питома ізобарна теплоємність c_p дорівнює ізохорної теплоємності c_v ($\mathrm{c}_p = \mathrm{c}_v = c$).

Тоді рівняння набуде такого вигляду:

$$c\rho \frac{\partial T}{\partial \tau} = div(\lambda \, gradT) + I_q. \tag{1.6}$$

У загальному випадку коефіцієнт теплопровідності λ і питома теплоємність c, а також джерело теплоти I_q залежать від температури. Зазвичай λ та $c\rho$ вважають постійними, а тоді рівняння теплопровідності (1.6) приймає вид класичного рівняння теплопровідності Фур'є

$$\frac{\partial T}{\partial \tau} = \alpha \nabla^2 T + \frac{I_q}{c\rho}, \tag{1.7}$$

де α - коефіцієнт температуропровідності ($\alpha = \lambda / c\rho$). Для твердого тіла закон теплопровідності Фур'є можна написати так:

$$\vec{q} = -\vec{n}^1 \lambda \frac{\partial T}{\partial n} = -\vec{n}^1 \lambda \left(\frac{\partial T}{\partial u_V}\right) \frac{\partial u_V}{\partial n} = -\alpha_V \, grad \, u_V, \tag{1.8}$$

де u_V - об'ємна концентрація внутрішньої енергії; α_V - коеффіцієнт температуропроводності при постійному обсязі, $\alpha_V = \lambda / \mathrm{c}_p \rho$, так як

$$\left(\frac{\partial u_V}{\partial T}\right) = c_V \rho. \tag{1.9}$$

Отже, щільність потоку теплоти згідно з формулою (1.8) прямо пропорційна градієнту концентрації внутрішньої енергії тіла. Коефіцієнт пропорційності α_V є коефіцієнтом дифузії внутрішньої енергії тіла.

При постійному тиску коефіцієнт температуропровідності визначається співвідношенням $\alpha_p = \lambda/\mathrm{c}_p\rho$, де c_p - питома ізобарна теплоємність, дорівнює:

$$\mathrm{c}_p = \frac{1}{\rho}\frac{\partial H_V}{\partial T}. \tag{1.10}$$

Тут H_V - об'ємна концентрація ентальпії.

У цьому випадку закон теплопровідності Фур'є можна написати у вигляді

$$\vec{q} = -\vec{n}^1\lambda\left(\frac{\partial T}{\partial H_V}\right)\frac{\partial H_V}{\partial n} = -\alpha_p\ grad\ H_V, \tag{1.11}$$

де коефіцієнт пропорційності α_p між тепловим потоком і градієнтом об'ємної концентрації ентальпії є коефіцієнтом дифузії ентальпії.

Таким чином, коефіцієнт температуропровідності є коефіцієнтом дифузії внутрішньої енергії α_V або ентальпії α_p в залежності від умов сполучення тіла з навколишнім середовищем (V=const чи p=const).

1.2.4. Крайові умови

Диференціальне рівняння теплопровідності встановлює зв'язок між тимчасовими і просторовими змінами температури тіла; воно математично описує перенос теплоти всередині тіла. Для того, щоб знайти температурне поле всередині тіла в будь-який момент часу, тобто вирішити диференціальне рівняння, треба знати розподіл температури всередині тіла в початковий момент часу (початкова умова), геометричну форму тіла і закон взаємодії між навколишнім середовищем і поверхнею тіла (гранична умова).

Сукупність насального і граничного умов називається крайовими умовами; початкова умова називається тимчасовою крайовою умовою, а гранична умова – просторовою крайовою умовою.

Тимчасові (початкові) умови містять розподіл температури в тілі початковий момент часу - $t = 0: T = f(x, y, z)$ - в загальному вигляді.

При рівномірному розподілі температури в тілі початкова умова спрощується: $t = 0: T = T_0 = const$. Граничні умови визначають особливості протікання процесу на поверхні тіла і можуть бути задані декількома способами.

• Граничні умови першого роду – задається розподіл температури на поверхні (або кордоні) тіла для кожного моменту часу: $T = T_w(x, y, z, t)$, де T_w – температура на поверхні тіла. У багатьох практично значущих варіантах $T_w = const$.

• Граничні умови другого роду – задається значення теплового потоку для кожної точки поверхні (або межі) тіла в будь-який момент часу:

$$-\lambda\left(\frac{\partial T}{\partial \vec{n}}\right)_w = q_w(x, y, z, t), \tag{1.12}$$

де $\vec{n}$ – нормаль до поверхні тіла. Найбільш часто використовується умова $q_w = const$. Такий варіант теплообміну має місце, наприклад, при нагріванні різних виробів в високотемпературних печах.

• Граничні умови третього роду – задається взаємозв'язок між потоком тепла за рахунок теплопровідності від твердої стінки і тепловим потоком з навколишнього середовища за рахунок температурного напору (закон Ньютона-Ріхмана):

$$-\lambda\left(\frac{\partial T}{\partial \vec{n}}\right)_w = \alpha(T_w - T^e), \tag{1.13}$$

де α – коефіцієнт теплообміну. Це найбільш широко застосовується умова в задачах теплотехніки.

• Граничні умови четвертого роду - для визначення теплового взаємодії між елементами, що мають різні теплофізичні характеристики, задають умови

рівності температур і теплових потоків по обидва боки від кордону розділу:

$$\begin{cases} -\lambda_1 \left(\frac{\partial T_1}{\partial \vec{n}}\right)_\Gamma = -\lambda_2 \left(\frac{\partial T_2}{\partial \vec{n}}\right)_\Gamma \\ T_1(x_\Gamma, y_\Gamma, z_\Gamma, t) = T_2(x_\Gamma, y_\Gamma, z_\Gamma, t) \end{cases} \quad (1.14)$$

де $x_\Gamma, y_\Gamma, z_\Gamma$ – координати кордону розділу середовищ; T_1, T_2 – температури дотичних середовищ. Ця умова застосовується, наприклад, при рішенні задач теплопровідності для багатошарових пластин.

Диференціальне рівняння (1.5) разом з умовами однозначності дає повну математичну формулювання крайової задачі теплопровідності.

При вирішенні конкретних крайових задач нестаціонарної теплопровідності можна, застосовуючи методи математичного моделювання, домогтися істотного спрощення загальної математичної постановки. Так, якщо для даного процесу:

$$\frac{\partial^2 T}{\partial x^2} \gg \frac{\partial^2 T}{\partial y^2} \text{ та } \frac{\partial^2 T}{\partial x^2} \gg \frac{\partial^2 T}{\partial z^2}, \quad (1.15)$$

то можна замість рівняння (1.5) обмежитися одномірним нестаціонарним рівнянням кондуктивного теплопереносу

$$\rho c \frac{\partial T}{\partial t} = \frac{\partial}{\partial x}\left(\lambda \frac{\partial T}{\partial x}\right) + Q_w(x, t, T), \quad (1.16)$$

яке разом з умовами однозначності дає більш просту математичне формулювання крайової задачі. Є багато практично значущих випадків, коли рішення рівняння (1.16) досить для повного опису даного процесу.

У практиці теплотехнічних розрахунків часто виникають одномірні завдання з циліндричної або сферичної симетрією. Наприклад, циліндрична симетрія є в задачах про охолодженні довгого циліндра або при аналізі теплового

стану в трубчастих каналах.

Природною системою координат в таких завданнях є, відповідно, циліндрична (r, φ) або сферична (r, θ, φ). Внаслідок одномірності все величини не залежатимуть від кутів θ, φ. Тоді параболічне рівняння (1.2) зі змінними коефіцієнтами в відповідних координатах прийме вигляд:

$$\rho c \frac{\partial T}{\partial t} = \frac{1}{r^{\nu}} \frac{\partial}{\partial r} \left(\lambda r^{\nu} \frac{\partial T}{\partial r} \right) + Q_w(r, t, T), \tag{1.17}$$

де r - радіальна координата, ν - показник симетрії, рівний 0, 1, 2 відповідно для плоского, циліндричного і сферичного випадків.

На рис 1.1 дана графічна інтерпретація чотирьох видів граничних умов. Скалярна величина вектора теплового потоку пропорційна абсолютній величині градієнта температури, який чисельно дорівнює тангенсу кута нахилу дотичної до кривої розподілу температури уздовж нормалі до ізотермічної поверхні, тобто

$$\left(\frac{\partial T}{\partial n} \right)_{\text{п}} = tg\psi_{\text{п}}. \tag{1.18}$$

На рис. 1.1 зображені на поверхні тіла чотири елементи поверхні ΔS з нормаллю до неї n (нормаль вважається позитивною, якщо вона спрямована назовні). По осі ординат відкладена температура.

Гранична умова першого роду полягає в тому, що задана $T_{\text{п}}(\tau)$; в найпростішому випадку $T_{\text{п}}(\tau) = const$. Відшукується нахил дотичної до температурної кривої у поверхні тіла, а тим самим і кількість тепла, що віддається поверхнею (див. рис. 1.1, а).

Завдання з граничними умовами другого роду мають зворотний характер; задається тангенс кута нахилу дотичної до температурної кривої у поверхні тіла (див. рис. 1.1, б); знаходиться температура поверхні тіла.

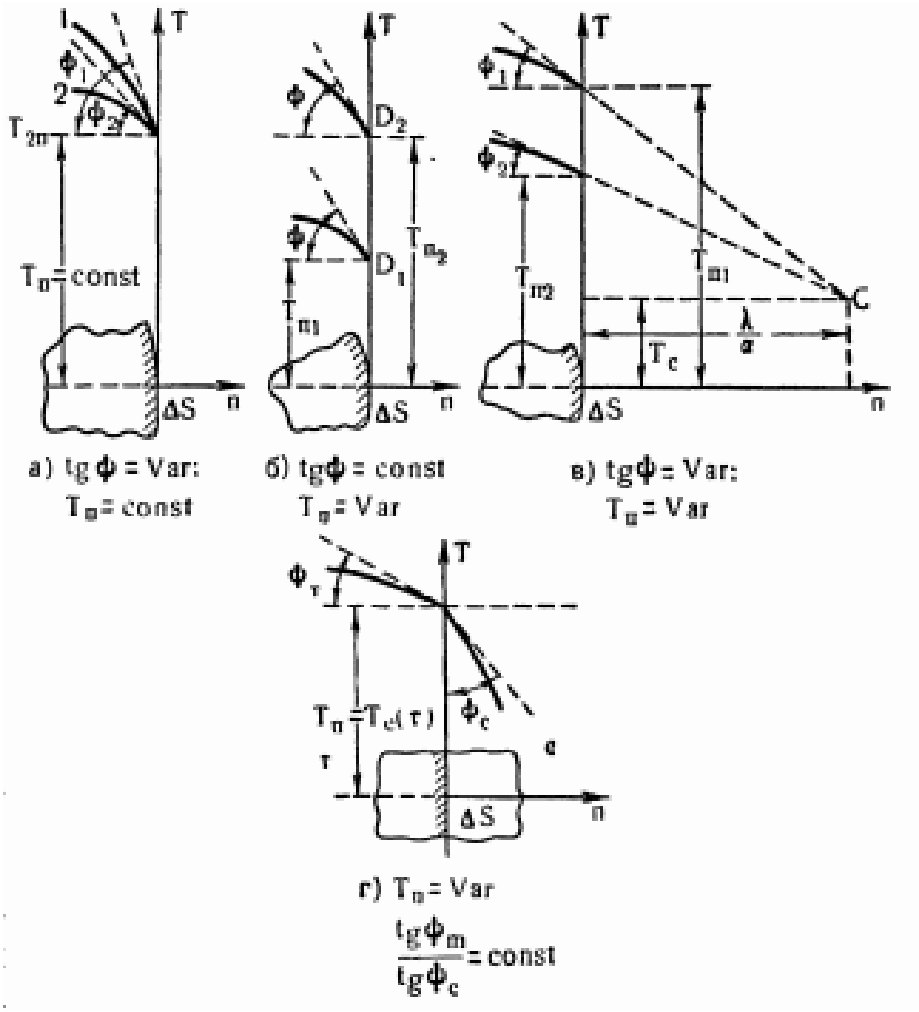

Рис. 1.1. Графічна інтерпретація граничних умов

У завданнях з граничними умовами третього роду температура поверхні тіла і тангенс кута нахилу дотичної до температурної кривої - величини змінні, але задається на зовнішньої нормалі точка С, через яку повинні проходити всі дотичні до температурної кривої (див. рис. 1.1, в).

Тангенс кута нахилу дотичної до температурної кривої у поверхні тіла дорівнює відношенню протилежного катета $[T_{\text{п}}(\tau) - T_c]$ до прилеглого катета $\frac{\lambda}{\alpha}$ відповідного прямокутного трикутника. Прилеглий катет $\frac{\lambda}{\alpha}$ є величиною постійною, а протилежний катет $[T_{\text{п}}(\tau) - T_c]$ безперервно змінюється в процесі теплообміну прямо пропорційно $tg\psi_{\text{п}}$. Звідси випливає, що напрямна точка С є незмінною.

У завданнях з граничними умовами четвертого роду задається відношення

тангенсів кута нахилу дотичних до температурних кривих в тілі і в середовищі на кордонах їх розділу (див. рис. 1.1, г):

$$\frac{tg\psi_m}{tg\psi_c} = \frac{\lambda_c}{\lambda} = const \tag{1.19}$$

з урахуванням вчиненого теплового контакту (дотичні біля поверхні розділу проходять через одну і ту ж точку).

Диференціальне рівняння спільно з початковими і граничними умовами повністю пределяет завдання, тобто, знаючи геометричну форму тіла, початкові і граничні умови, можна диференціальне рівняння вирішити до кінця і, отже, знайти функцію розподілу температури в будь-який момент часу. Таким чином, в результаті рішення повинна бути знайдена функція $T(x, y, z, \tau) = f(x, y, z, \tau)$.

Функція $f(x, y, z, \tau)$ повинна задовольняти диференціальному рівнянню (при підстановці її замість Т в диференціальне рівняння теплопровідності воно повинно звертатися в тотожність), а також початковій і граничній умов.

По теоремі єдиності рішення, якщо деяка функція $T(x, y, z, \tau)$ задовольняє диференціальних рівнянь теплопровідності, початковим і граничним умовам, то вона є єдиним рішенням даного завдання.

1.2.5. Методи розрахунку тепла

В процесі нагрівання або охолодження тіло сприймає або віддає певну кількість тепла. Існують три способи визначення витрати тепла в процесі теплообміну.

1. До елементу поверхні dS за час $d\tau$ підводиться тепло, рівне

$$-\lambda \left(\frac{\partial T}{\partial n}\right) dS\, d\tau. \tag{1.20}$$

Для знаходження кількості тепла ΔQ, сприйманого тілом за проміжок часу $\Delta\tau = \tau_2 - \tau_1$, потрібно співвідношення (1.20) проінтегрувати по всій поверхні S і інтервалу часу $\Delta\tau$:

$$\Delta Q = -\int_{\tau_1}^{\tau_2} \int_{(S)} \lambda \left(\frac{\partial T}{\partial n}\right) dS\, d\tau. \tag{1.21}$$

Зазвичай температура і температурний градієнт однакові уздовж поверхні; тоді розрахункова формула (1.21) спрощується:

$$\Delta Q = Q_2 - Q_1 = -\lambda S \int_{\tau_1}^{\tau_2} \left(\frac{\partial T}{\partial n}\right) d\tau. \tag{1.22}$$

2. Елемент обсягу $dv = dx\, dy\, dz$ за час $\Delta\tau = \tau_2 - \tau_1$ нагрівається від T_1 до T_2; він сприймає кількість тепла, яке дорівнює

$$c\,\gamma\,(T_2 - T_1)dv. \tag{1.23}$$

Загальна кількість тепла ΔQ, яке пішло на нагрівання за час, знайдемо, якщо проинтегрируем по всьому об'єму V, тобто

$$\Delta Q = Q_2 - Q_1 = c\,\gamma \int_{(V)} (T_2 - T_1)dv = c\,\gamma\, V \frac{1}{V} \int_{(V)} (T_2 - T_1)dv. \tag{1.24}$$

Позначимо середню (інтегральну) температуру по всьому об'єму тіла через $\overline{T}$, тобто

$$\overline{T} = \frac{1}{V} \int_{(V)} T\, dv, \tag{1.25}$$

тоді можна написати:

$$\Delta Q = Q_2 - Q_1 = c\,\gamma\,V\,(\overline{T}_2 - \overline{T}_1), \tag{1.26}$$

так як в процесі нагрівання $\overline{T}_2 > \overline{T}_1$.

Витрата тепла $(Q - Q_0)$ на нагрівання за час τ від початку процесу $\tau_1 = 0$ буде дорівнювати

$$Q - Q_0 = c\,\gamma\,V\,(\overline{T} - \overline{T}_0), \tag{1.27}$$

де $\overline{T}_0$ - середня (інтегральна) початкова температура. Якщо початкова температура однакова у всіх точках тіла, тобто $\overline{T}_0 = T_0 = const$, то питома витрата тепла дорівнює

$$\Delta Q_v = c\,\gamma\,(\overline{T} - \overline{T}_0). \tag{1.28}$$

Отже, основне завдання в цьому методі розрахунку зводиться до визначення $\overline{T}(\tau)$.

3. Елемент поверхні dS за час $d\tau$ сприймає з навколишнього середовища кількість тепла, яке дорівнює

$$\alpha(\overline{T}_c - \overline{T}_{\text{п}})dSd\tau. \tag{1.29}$$

Для знаходження загальної кількості тепла, сприйманого по всій поверхні тіла, потрібно проінтегрувати по всій поверхні і проміжку часу $\Delta\tau = \tau_2 - \tau_1$.

Якщо температура поверхні тіла однакова у всіх точках і коефіцієнт не залежить від температури, то матимемо:

$$\Delta Q = Q_2 - Q_1 = \alpha S \int_{\tau_1}^{\tau_2} [T_{\text{с}} - T_{\text{п}}(\tau)] d\tau. \quad (1.30)$$

1.3. Висновки до першого розділу

Теоретичне дослідження процесів теплообміну в даний час в значній мірі базується на їх чисельному моделюванні з використанням ЕОМ. Це стало можливим завдяки значному прогресу в розвитку обчислювальних методів вирішення завдань для рівнянь в приватних похідних і збільшення потужності сучасних обчислювальних машин.

Чисельне моделювання процесів теплообміну в даний час набуває все більш значну роль в зв'язку з тим, що для сучасної науки і техніки необхідний достовірний прогноз таких процесів, експериментальне вивчення яких в лабораторних або натурних умовах дуже складно і дорого чи просто неможливо. Чисельне моделювання процесів теплообміну все успішніше входить в практику роботи різних науково-дослідних, проектно-конструкторських і виробничих установ.

У сучасних умовах виробництва математичне моделювання грає дуже важливу роль, виступаючи як засіб оптимізації та прогнозування. Моделювання теплових полів є, зокрема, однією з найбільш актуальних проблем при проектуванні електронних компонентів. Візуалізація теплових полів, тобто отримання термограми, являє собою заключну частину моделювання теплових процесів. Іншими словами, при створенні спеціальних засобів розрахунку теплових параметрів і моделювання теплових процесів, розробник зіткнеться з завданням візуалізації теплових полів [5].

Тому функціональне призначення даної системи полягає в наданні користувачу зручного та надійного шляху подивитися та проаналізувати розподіл температурних полів за допомогою термограми.

РОЗДІЛ 2
ТЕХНОЛОГІЯ МОДЕЛЮВАННЯ ТА ВІЗУАЛІЗАЦІЇ ТЕМПЕРАТУРНИХ ПОЛІВ

2.1. Опис використаних технологій та мов програмування

При розробці даної програми застосувалась мова програмування C# з використанням бібліотек OpenGL та TaoFramework, а також середа розробки Microsoft Visual Studio на платформі .NET Framework.

2.1.1. Мова програмування C# та платформа .NET Framework

.NET Framework (читається дот-нет) — програмна технологія, запропонована фірмою Microsoft як платформа для створення як звичайних програм, так і веб-застосунків. Багато в чому є продовженням ідей та принципів, покладених в технологію Java. Однією з головних ідей .NET є сумісність служб, написаних різними мовами. Кожна бібліотека (збірка) в .NET має свідчення про свою версію, що дозволяє усунути можливі конфлікти між різними версіями збірок.

.NET поділяється на дві основні частини — середовище виконання (по суті віртуальна машина) та інструментарій розробки. середовище розробки .NET створює байт-код, призначений для виконання віртуальною машиною. Вхідна мова цієї машини в .NET називається CIL (Common Intermediate Language), також відома як MSIL (Microsoft Intermediate Language), або просто IL. Застосування байт-коду дозволяє отримати крос-платформність на рівні скомпільованого проєкту (в термінах .NET: збірка), а не на рівні сирцевого тексту. Перед запуском збірки в середовищі виконання (CLR) байт-код перетворюється вбудованим в середовище JIT-компілятором (just in time, компіляція на льоту) в машинні коди цільового процесора. На рис. 2.1 показані основні компоненти .NET Framework.

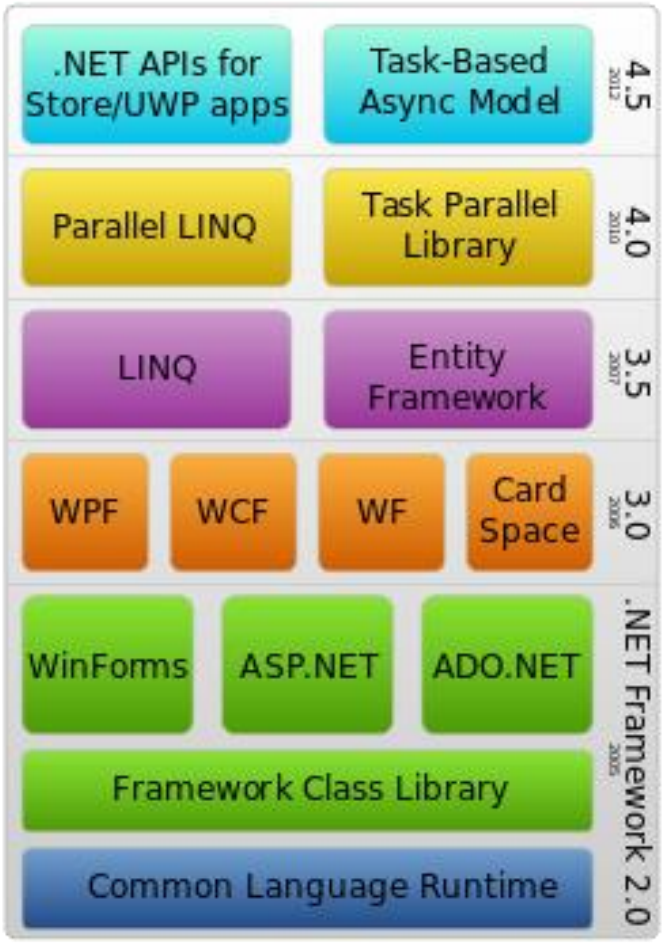

Рис. 2.1. Складові компоненти .NET Framework

.NET представляє потужну платформу для створення додатків. Можна виділити наступні її основні риси:

- Підтримка декількох мов. Основою платформи є загальномовне середовище виконання Common Language Runtime (CLR), завдяки чому .NET Framework підтримує кілька мов, у тому числі C#. При компіляції код на будь-якому з цих мов компілюється в збірку спільною мовою CIL (Common Intermediate Language) - свого роду асемблер платформи .NET Framework. Тому за певних умов можна зробити окремі модулі однієї програми на окремих мовах.
- Кросплатформеність. .NET Framework є переносною платформою (з деякими обмеженнями). Використовуючи різні технології на цій платформі, можна розробляти програми на мові C# для самих різних платформ.
- Потужна бібліотека класів. .NET Framework представляє єдину для всіх

підтримуваних мов бібліотеку класів. І який б додаток не був написаний на C#, так чи інакше задіється бібліотека класів .NET Framework.

- Різноманітність технологій. Загальномовне середовище виконання CLR і базова бібліотека класів є основою для цілого стека технологій, які розробники можуть задіяти при побудові тих чи інших додатків. Наприклад, для роботи з базами даних в цьому стеку технологій призначена технологія ADO.NET і Entity Framework Core. Для побудови графічних додатків з багатим насиченим інтерфейсом - технологія WPF і UWP, для створення більш простих графічних додатків - Windows Forms. Для розробки мобільних додатків - Xamarin. Для створення веб-сайтів і веб-додатків - ASP.NET і т.д.
- Продуктивність. Згідно ряду тестів додатки на .NET Framework в ряді категорій сильно випереджають додатки, побудовані за допомогою інших технологій.

У цій кваліфікаційній роботі програма буде написана на мові програмування C#. C# – це об'єктно-орієнтована мова програмування, розроблена у 1998-2001 роках групою інженерів компанії Microsoft під керівництвом Андерса Хейлсберга і Скотта Вільтаумота як мова розробки додатків для платформи Microsoft .NET Framework.

C# використовується для розробки застосунків, призначених для широкого спектру систем – від споживчих пристроїв до неоднорідної інфраструктури підприємства.

Переваги C# для вибору у якості мови програмування при розробці:

- портативність та багатоплатформність відкриває широкі можливості для розробки та використання програм, написаних на цій мові програмування;
- велика популярність робить розробку більш простою;
- дуже гнучка сумісність версій покращує процес розробки;
- широкий попит на цю мову програмування.

На сьогодні C# є флагманським мовою корпорації Microsoft, тому що вона найбільш повно використовує нові можливості .NET. Решта мов програмування,

хоча і підтримуються, визнані тими, що мають спадкові прогалини щодо використання .NET[14].

Особливості

C# відноситься до сім'ї мов з C-подібним синтаксисом, з них його синтаксис найбільш близький до C++ і Java.

Окрім всього:

- точки з комою використовуються для позначення кінця рядка коду;
- фігурні дужки використовуються для угруповання операторів;
- рядки коду зазвичай групуються в методи (функції), методи в класи і класи в простір імен;
- змінні присвоюються з використанням знаку рівності, а також порівнюються з використанням двох послідовних знаків рівності;
- квадратні дужки використовуються для роботи з масивами, як для їх оприлюднення, так і для отримання значення за заданим індексом в одному з них.

Мова має статичну типізацію, підтримує поліморфізм, перевантаження операторів (в тому числі операторів явного і неявного приведення типу), делегати, атрибути, події, властивості, узагальнені типи і методи, ітератори, анонімні функції з підтримкою замикань, LINQ, виключення, коментарі у форматі XML [15].

2.1.2. Visual Studio, бібліотеки OpenGL та Tao Framework

Microsoft Visual Studio

Microsoft Visual Studio - лінійка продуктів компанії Microsoft, що включають інтегроване середовище розробки програмного забезпечення і ряд інших інструментальних засобів. Дані продукти дозволяють розробляти як консольні додатки, так і додатки з графічним інтерфейсом, в тому числі з підтримкою технології Windows Forms, а також веб-сайти, веб-додатки, веб-

служби як в рідному, так і в керованому кодах для всіх платформ, підтримуваних Windows, Windows Mobile, Windows CE, .NET Framework, Xbox, Windows Phone .NET Compact Framework і Silverlight.

Microsoft Visual Studio включає в себе інтегроване середовище розробки програмного забезпечення і ряд інших інструментальних засобів, за допомогою яких було розроблено програмне забеспечення. MS Visual Studio дозволяє просто, зручно та швидко створювати великі та масштабні проекти на багатьох мовах програмування, швидко аналізувати код та робити його відладку. До того ж Microsoft Visual Studio найкраще з усіх варіантів підтримує мову програмування C#, що було найголовнішим фактором при виборі [20].

Visual Studio включає в себе редактор вихідного коду з підтримкою технології IntelliSense і можливістю найпростішого рефакторінга коду. Вбудований відладчик може працювати як відладчик рівня вихідного коду, так і відладчик машинного рівня. Решта вбудовуються інструменти включають в себе редактор форм для спрощення створення графічного інтерфейсу додатку, веб-редактор, дизайнер класів і дизайнер схеми бази даних.

Visual Studio дозволяє створювати і підключати сторонні додатки (плагіни) для розширення функціональності практично на кожному рівні, включаючи додавання підтримки систем контролю версій вихідного коду (як, наприклад, Subversion і Visual SourceSafe), додавання нових наборів інструментів (наприклад, для редагування і візуального проектування коду на предметно-орієнтованих мовах програмування) або інструментів для інших аспектів процесу розробки програмного забезпечення (наприклад, клієнт Team Explorer для роботи з Team Foundation Server) [21].

Для виконання даної роботи також були використані бібліотеки Tao Framework і OpenGL для C#.

OpenGL – це середовище для розробки портативних, інтерактивних 2D та 3D графічних додатків. З моменту своєї появи в 1992 році OpenGL став найбільш широко використовуваним і підтримуваним в галузі інтерфейсом 2D і 3D графічного програмування програм (API).

Основним принципом роботи OpenGL є отримання наборів векторних графічних примітивів у вигляді точок, ліній і трикутників з наступною математичною обробкою отриманих даних і побудовою растрової картинки на екрані та/або в пам'яті. Векторні трансформації і растеризация виконуються графічним конвеєром, який по суті являє собою дискретний автомат. Абсолютна більшість команд OpenGL потрапляє в одну з двох груп: або вони додають графічні примітиви на вхід в конвеєр, або конфігурують конвеєр на різне виконання трансформацій. На рис. 2.2 показані графічні примитиви, що використовуються в OpenGL.

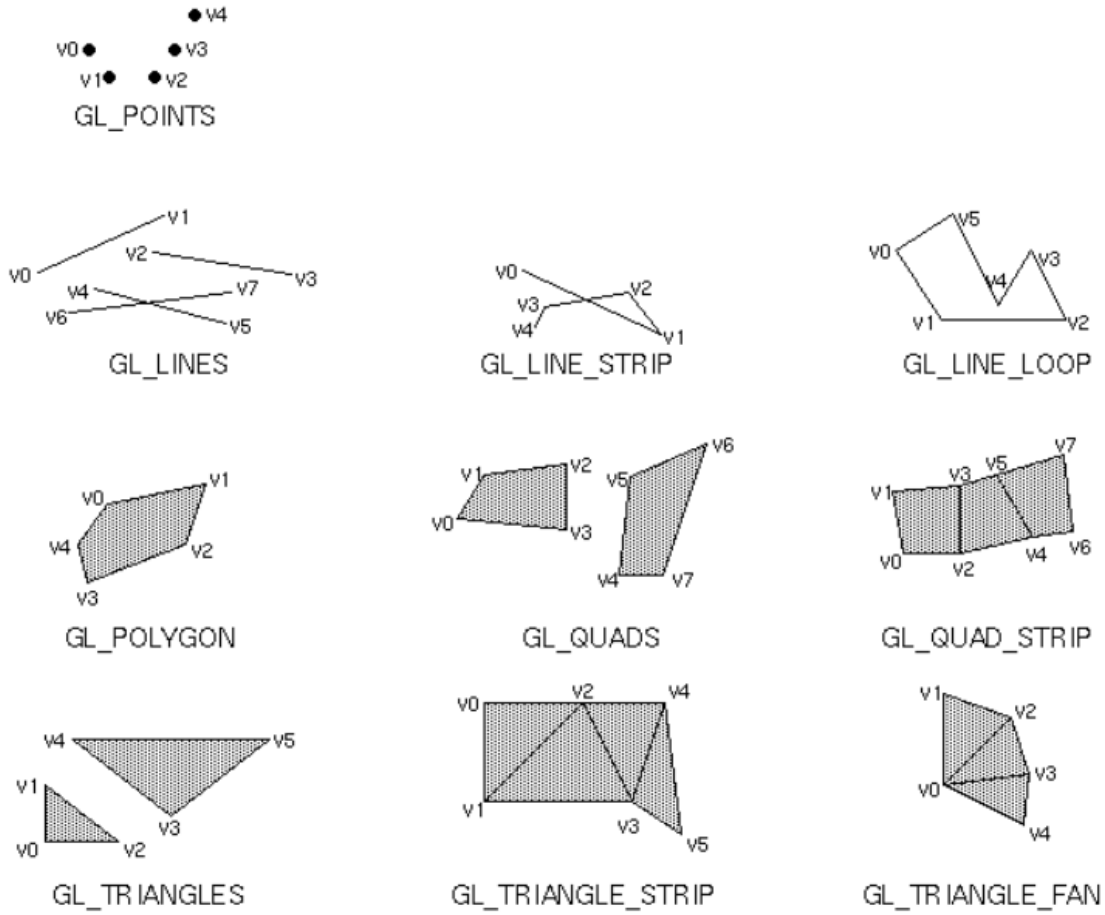

Рис. 2.2. Графічні примитиви OpenGL

На рис. 2.3 показано, як OpenGL обробляє дані. Як показано, команди входять зліва і проходять по конвеєру обробки. Деякі команди визначають геометричні об’єкти, які слід намалювати, а інші контролюють, як обробляються

об’єкти на різних етапах обробки.

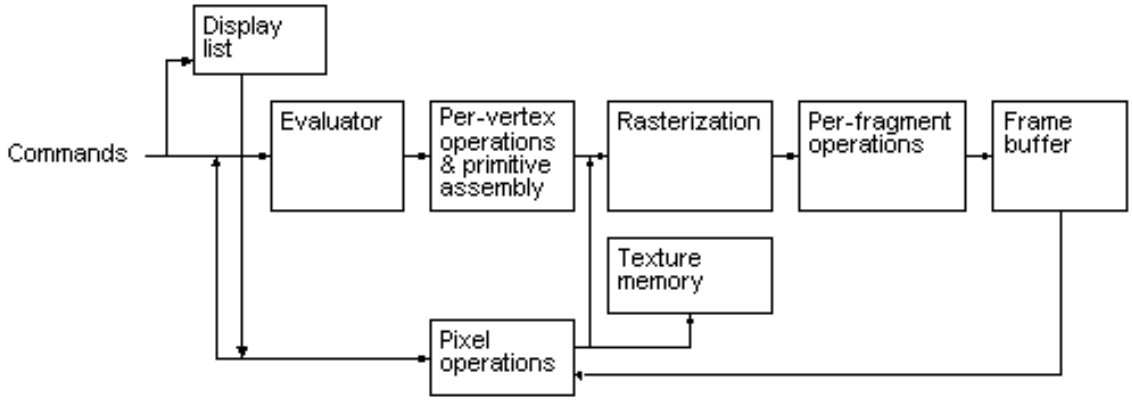

Рис. 2.3. Схема обробки даних в OpenGL

Етапи обробки в базовій операції OpenGL такі:

- Відображення списку. Замість того, щоб усі команди негайно проходили по конвеєру, ви можете накопичити деякі з них у списку відображення для подальшої обробки.

- Оцінювач. Етап обробки оцінювача забезпечує ефективний спосіб наближення кривої та геометрії поверхні шляхом обчислення поліноміальних команд вхідних значень.

- Операції над вершинами та примітивна збірка. OpenGL обробляє геометричні примітивні точки, відрізки ліній та багатокутники, усі з яких описуються вершинами. Вершини перетворюються та освітлюються, а примітиви відсікаються до області перегляду під час підготовки до растеризації.

- Растеризація. Етап растеризації створює серію адрес буфера кадру та пов'язаних значень за допомогою двовимірного опису точки, відрізка лінії або багатокутника. Кожен утворений таким чином фрагмент подається на останню стадію за операціями за фрагментом.

- Операції за фрагментом. Це остаточні операції, що виконуються над даними, перш ніж вони будуть збережені у вигляді пікселів у буфері кадрів. Операції з фрагментами включають умовне оновлення буфера кадрів на основі

вхідних та раніше збережених значень z (для буферизації z) та поєднання кольорів вхідних пікселів із збереженими кольорами, а також маскування та інші логічні операції над значеннями пікселів.

Дані можна вводити у формі пікселів, а не вершин. Дані у формі пікселів, такі як може описувати зображення для використання у відображенні текстур, пропускає перший етап обробки, описаний вище, і замість цього обробляється як пікселі на етапі операцій з пікселями. Після піксельних операцій дані пікселів зберігаються як пам’ять текстур для використання на стадії растеризації або растеризуються, при цьому отримані фрагменти об’єднуються в буфер кадрів так само, як ніби вони були створені з геометричних даних.

Серед інших засобів для вирішення поставленої задачі візуалізації температурних полів OpenGL виділяється такими ознаками:

- один із найбільш популярних відкритих мультиплатформених графічних стандартів;
- усі програми OpenGL дають послідовні результати візуального відображення на будь-якому обладнанні, сумісному з API OpenGL, незалежно від операційної системи;
- програми на основі OpenGL API програми можуть масштабуватися до будь-якого класу машини, на яку розробник вирішив націлити, починаючи від побутової електроніки, закінчуючи ПК, робочими станціями та суперкомп’ютерами;
- OpenGL дає розробнику широкий асортимент можливостей для використання комп'ютерної графіки, в тому числі для візуалізації температурних полів.

Tao Framework

Tao Framework – це вільно розповсюджена бібліотека з відкритим вихідним кодом, призначена для побудови та зручної розробки крос-платформного мультимедійного програмного забезпечення у середі .NET Framewrok і Mono. На сьогоднішній день Tao Framework – це один із найкращих

шляхів для використання бібліотек OpenGL при розробці у середі .NET на мові C#. У складі бібліотеки на сьогоднішній момент надходять усі сучасні засоби, які можуть бути надані в ході розробки мультимедіа програмного забезпечення, у тому числі реалізації бібліотеки OpenGL та FreeGlut.

2.2. Опис структури системи та алгоритмів її функціонування

На рис. 2.4 наведена структура розробленої системи та алгоритмів її функціювання у вигляді UML-діаграми:

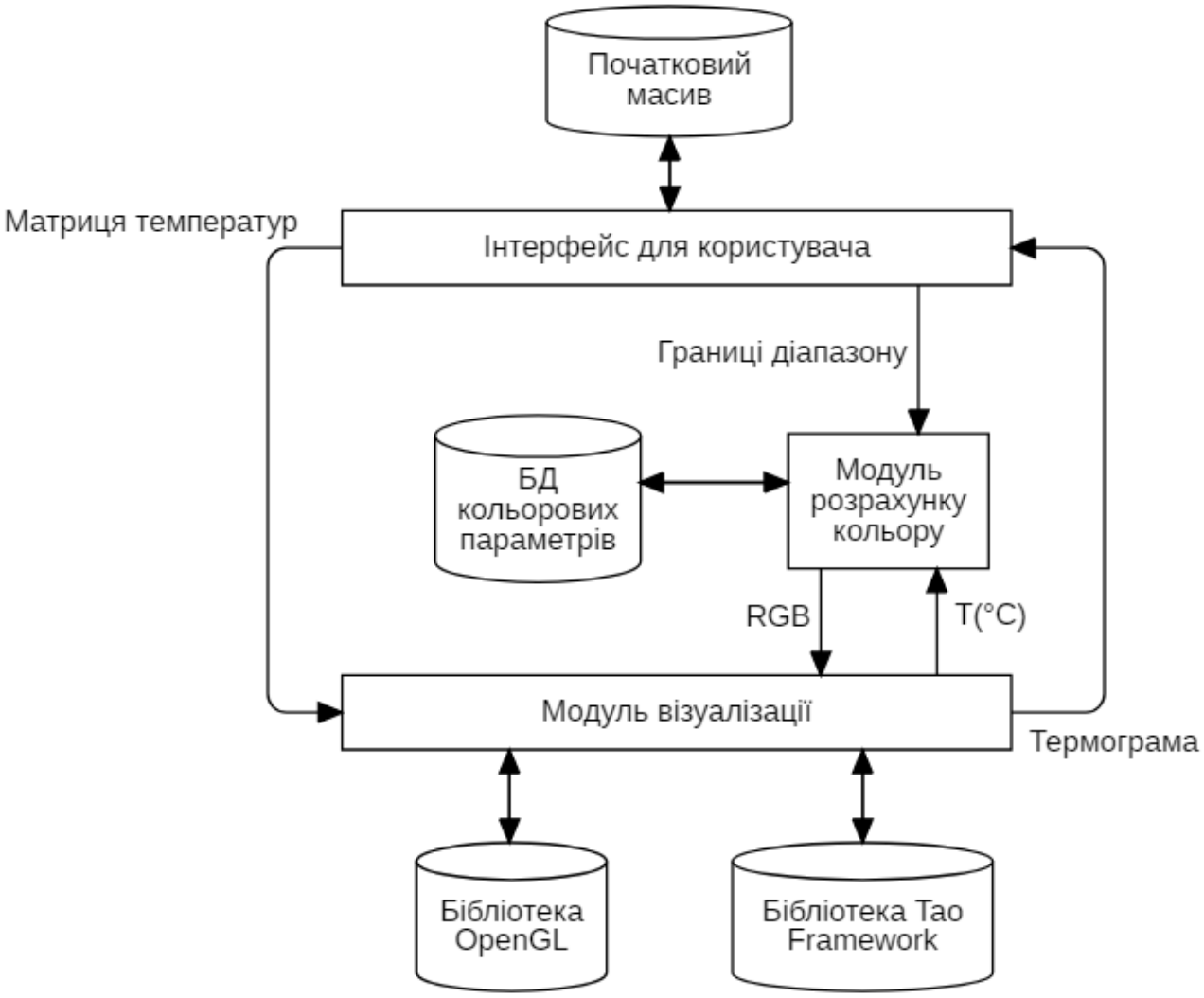

Рис. 2.4. UML-діаграма структури системи.

Розроблений програмний компонент складається з чотирьох основних

частин - файлу з вихідним масивом, призначеного для користувача інтерфейсу, модуля візуалізації та модуля розрахунку кольору.

Вихідний масив - це матриця температур моделіруємого кристала, розрахована за допомогою спеціальних програмних засобів і зберігається в окремому файлі. О методах розрахунку цієї матриці на основі програмної моделі пристрою детально розказується в роботах [22-26] та в розділі 3 кваліфікаційної роботи.

Модуль призначеного для користувача інтерфейсу відображає вихідний масив як у вигляді тексту, так і у вигляді термограмми. У цьому модулі передбачені функції збереження термограмми в графічному форматі.

У модулі візуалізації відбуваються ініціалізація компонентів бібліотеки OpenGL та відображення вихідного масиву в графічному вигляді. Цей модуль зчитує вміст комірки вихідного масиву і передає це значення в модуль розрахунку кольору, а у відповідь отримує RGB-параметри кольору, відповідного температурі даного осередку. На основі колірних параметрів кожного осередку будується колірної градієнт, який і є термограммою.

Модуль розрахунку кольору визначає RGB-складові кольору, які відповідають вхідному параметру температури. Верхня і нижня межі температури визначають інтервал, на якому буде відбуватися візуалізація. Температурі, що виходить за нижню межу, відповідає чорний колір. Температурі, що виходить за верхню межу, відповідає білий колір. Усередині інтервалу колірної градієнт змінюється по зростанню температури в такій послідовності: чорний, синій, фіолетовий, червоний, помаранчевий, жовтий, білий.

Ініціалізація графічних компонентів

Для роботи використовуються простору імен Tao.OpenGl, Tao.FreeGlut і Tao.Platform.Windows. Висновок графічного зображення здійснений через елемент simpleOpenGlControl простору імен Tao.Platform.Windows. Примірник цього елемента має ім'я Ant.

Бібліотека функцій FreeGlut дозволяє програмісту прикладних програм

створювати, керувати та керувати вікнами незалежно від операційної системи, на якій працює програма. Приховуючи від програміста залежність від операційної системи, це дозволяє писати справді портативні програми OpenGL.

Ініціалізація GLUT робиться за допомогою команди void glutInit(int *argcp, char **argv).

Перший параметр представляє з себе покажчик на кількість аргументів у командному рядку, а другий - покажчик на масив аргументів. Зазвичай ці значення беруться з головної функції програми: int main (int argc, char *argv []).

Необхідно також встановити для вікна режим відображення інформації. Тобто встановити для вікна такі параметри як: використовувана колірна модель, кількість різних буферів, і т.д. Для цього в FreeGlut існує команда void glutInitDisplayMode(unsigned int mode). У команди є єдиний параметр, який може бути представлений однією з наступних констант (таблиця 1.1) або комбінацією цих констант за допомогою побітового АБО.

Таблиця 1.1

Константа	Значення
GLUT_RGB	Для відображення графічної інформації використовуються 3 компоненти кольору RGB.
GLUT_RGBA	Аналогічно RGB, але використовується також четвертий компонент ALPHA (прозорість).
GLUT_INDEX	Колір задається не за допомогою RGB компонентів, а за допомогою палітри. Використовується для старих дисплеїв, де кількість квітів наприклад 256.
GLUT_SINGLE	Висновок у вікно здійснюється з використанням 1 буфера. Зазвичай використовується для статичного виведення інформації.
GLUT_DOUBLE	Висновок у вікно здійснюється з використанням 2 буферів. Застосовується для анімації, щоб виключити ефект мерехтіння.
GLUT_ACCUM	Використовувати також буфер накопичення (Accumulation Buffer). Цей буфер застосовується для створення спеціальних ефектів, наприклад відображення і тіні.

Продовження таблиці 1.1

Константа	Значення
GLUT_ALPHA	Використовувати буфер ALPHA. Цей буфер, як уже говорилося використовується для завдання 4-го компонента кольору - ALPHA. Зазвичай застосовується для таких ефектів як прозорість об'єктів і антиалиасинг.
GLUT_DEPTH	Створити буфер глибини. Цей буфер використовується для відсікання невидимих ліній в 3D просторі при виведенні на плоский екран монітора.
GLUT_STENCIL	Буфер трафарету використовується для таких ефектів як вирізання частини фігури, роблячи цей шматок прозорим. Наприклад, наклавши прямокутний трафарет на стіну будинку, ви отримаєте вікно, через яке можна побачити що знаходиться всередині будинку.
GLUT_STEREO	Цей прапор використовується для створення стереозображень. Використовується рідко, так як для перегляду такого зображення потрібна спеціальна апаратура.

Після ініціалізації вікна встановлюється колір очищення вікна за допомогою функції glClearColor(R, G, B, A), де R, G, B – кольорові компоненти схеми RGB, A – прозорість.

Після цього відбувається настройка проекції. Для цього ми спочатку викликаємо функцію:

Gl.glMatrixMode (Gl.GL_PROJECTION);

Функція glMatrixMode призначена для того, щоб ставити матричний режим: буде визначена матриця, над якою ми надалі будемо проводити операції. У цій роботі це GL_PROJECTION - матриця проекцій. Наступною командою очищається матриця за допомогою функції glLoadIdentity (функція замінює поточну матрицю на одиничну). Далі ми встановлюємо тип поточної проекції за допомогою функції gluOrtho2D.

Тепер, коли проекція визначена, встановлюється в якості поточної матриці об'єктно-видова матриця й очищається:

Gl.glMatrixMode (Gl.GL_MODELVIEW); Gl.glLoadIdentity ();

Нижче наведено фрагмент функції graphic_init, що виконує ініціалізацію

OpenGL в C#. При цьому n та m – це відповідно довжина та ширина матриці температур MatrT.

```
private void graphic_init(int n, int m)
{
        // ініціалізація Glut
         Glut.glutInit();
         Glut.glutInitDisplayMode(Glut.GLUT_RGB |
         Glut.GLUT_DOUBLE | Glut.GLUT_DEPTH);
         // очистка вікна
         Gl.glClearColor(255, 255, 255, 1);
         // настройка проекції
         Gl.glMatrixMode(Gl.GL_PROJECTION);
         Gl.glLoadIdentity();
         // настройка 2D ортогональной проекції в
         // відповідності з розмірами відображенної матриці
         Glu.gluOrtho2D(0, m, 0, n);
         //установка об’ектно-видової матриці та очистка
         Gl.glMatrixMode(Gl.GL_MODELVIEW);
         Gl.glLoadIdentity();
}
```

Розрахунок кольору комірки.

При візуалізації теплового поля, кожному значенню температури ставиться у відповідність свій колір в залежності від значень максимальної та мінімальної температури в матриці температур. Нижче представлений метод TempToRGB, який відповідає за розрахунок RGB-параметрів кольору комірки із заданою температурою.

```
private void TempToRGB(double T, double Tmax, double Tmin)
      {
         double Colour_lim = (Tmax - Tmin) / 6;
         double Tblue = Tmin + Colour_lim;
```

```
        double Tviolet = Tmin + Colour_lim * 2;
        double Tred = Tmin + Colour_lim * 3;
        double Torange = Tmin + Colour_lim * 4;
        double Tyellow = Tmin + Colour_lim * 5;
        double t = ((T - Tmin) % Colour_lim) / Colour_lim;
        double R = 0, G = 0, B = 0;
        R = (T <= Tblue) ? 0 :
        (T > Tviolet) ? 1 :
        t;
        G = (T <= Tred) ? 0 :
        (T > Tyellow) ? 1 :
        (T < Torange) ? t / 2 :
        0.5 + t / 2;
        B = (T <= Tmin) ? 0 :
        (T >= Tmin && T < Tblue) ? t :
        (T >= Tblue && T < Tviolet) ? 1 :
        (T >= Tviolet && T < Tred) ? 1 - t :
        (T >= Tred && T < Tyellow) ? 0 :
        (T >= Tyellow && T < Tmax) ? t :
        1;
        Gl.glColor3d(R, G, B);
    }
```

T – температура відображаємого елемента; Tmin и Tmax – границі діапазона відображаємих температур; R, G, B – складові колорьового спектра.

Функція Gl.glColor3d встановлює відповідний колір відтворення геометрії і грунтується на RGB складових.

Візуалізація матриці температур

Для включення можливості створення кількох екранів в одному контроллері використовується функція Gl.glEnable(Gl.GL_SCISSOR_TEST). Для виключення цієї можливості використовується зворотня функція

Gl.glEnable(Gl.GL_SCISSOR_TEST).

Тепер необхідно визначити значення порту виведення за допомогою функції Gl.glViewport(x, y, width, height), де x та y – початкові координати крайньої нижньої лівої точки порту, width та height – відповідно довжина та ширина порту.

Перед будь-якою візуалізацією проводиться очищення вікна (тому що до цього вже міг бути реалізований будь-який висновок). Для цього використовується функція glClear. Як параметр функція отримує дані, значення яких буферів їй необхідно очистити, тобто буфер кольору і буфер глибини.

Температурне поле в цій роботі складається з маленьких квадратів певного кольору. Спочатку вказується, що буде малюватися за допомогою glBegin з відповідним параметром. Далі вказуються вершини, що визначають об'єкти зазначеного типу. І нарешті, викликається glEnd, щоб вказати, що це кінець малювання об'єкта типу, зазначеного в glBegin.

В OpenGL побудування квадрату буде мати наступний вигляд:

```
glBegin(GL_QUADS); // початок рисування примитиву (квадрата)
    glVertex2d(double x, double y); // перша вершина (x та y – координати вершини квадрата)
    glVertex2d(double x, double y); // друга вершина
    glVertex2d(double x, double y); // третя вершина
    glVertex2d(double x, double y); // остання вершина
glEnd(); // кінець рисування квадрата
```

Щоб бібліотека OpenGL завершила візуалізацію цього кадру, використовується наступна функція:

```
Gl.glFlush();
```

Потім посилається елементу Ant, в якому відбувається візуалізація сцени, сигнал про те, що необхідно оновити відображається кадр, тобто іншими словами викликається його перемальовування.

```
Ant.Invalidate();
```

Описаний нижче метод out_PpenGL візуалізує як саму матрицю

температур, так і температурну шкалу в одному елементі Ant. Цей метод викликається за події, пов'язаної з таймером. Таким чином, зображення на екрані оновлюється постійно, що дозволяє швидше підібрати оптимальні значення граничних температур.

```
private void out_PpenGL(double[,] MatrT, int n, int m, double Tmax, double Tmin)
        {
           Gl.glEnable(Gl.GL_SCISSOR_TEST);
          //блок візуалізації матриці температур
           Gl.glViewport(0, 0, Ant.Width-70, Ant.Height);
           Gl.glScissor(0, 0, Ant.Width-70, Ant.Height);
           Gl.glClear(Gl.GL_COLOR_BUFFER_BIT);
           for (int i = 0; i < n - 1; i++)
           {
              for (int j = 0; j < m - 1; j++)
              {
                 Gl.glBegin(Gl.GL_QUADS);
                 TempToRGB(MatrT[i, j], Tmax, Tmin);
                 Gl.glVertex2d(j + 1, m - i);
                 TempToRGB(MatrT[i + 1, j], Tmax, Tmin);
                 Gl.glVertex2d(j + 1, m - i - 1);
                 TempToRGB(MatrT[i + 1, j + 1], Tmax, Tmin);
                 Gl.glVertex2d(j + 2, m - i - 1);
                 TempToRGB(MatrT[i, j + 1], Tmax, Tmin);
                 Gl.glVertex2d(j + 2, m - i);
                 Gl.glEnd();
              }
           }
           //створення границі між блоками візуалізації матриці температур та температурної шкали
```

```
Gl.glViewport(Ant.Width - 70, 0, Ant.Width, Ant.Height);
Gl.glScissor(Ant.Width - 70, 0, Ant.Width, Ant.Height);
Gl.glClear(Gl.GL_COLOR_BUFFER_BIT);
Gl.glBegin(Gl.GL_QUADS);
Gl.glColor3d(255, 255, 255);
Gl.glVertex3f(Ant.Width - 70, 0, 0);
Gl.glVertex3f(Ant.Width - 50, 0, 0);
Gl.glVertex3f(Ant.Width - 50, Ant.Height, 0);
Gl.glVertex3f(Ant.Width - 70, Ant.Height, 0);
Gl.glEnd();
//блок візуалізації температурної шкали
Gl.glViewport(Ant.Width - 50, 0, Ant.Width, Ant.Height);
Gl.glScissor(Ant.Width - 50, 0, Ant.Width, Ant.Height);
Gl.glClear(Gl.GL_COLOR_BUFFER_BIT);
double step = (Tmax - Tmin) / n;
for (int i = 0; i < n - 1; i++)
{
   Gl.glBegin(Gl.GL_QUADS);
   TempToRGB(Tmax - (i + 1) * step, Tmax, Tmin);
   Gl.glVertex2d(0, m - i);
   TempToRGB(Tmax - i * step, Tmax, Tmin);
   Gl.glVertex2d(0, m - i - 1);
   TempToRGB(Tmax - i * step, Tmax, Tmin);
   Gl.glVertex2d(50, m - i - 1);
   TempToRGB(Tmax - (i + 1) * step, Tmax, Tmin);
   Gl.glVertex2d(50, m - i);
   Gl.glEnd();
}
//відрисовування всього елементу Ant на екран
Gl.glFlush();
```

```
        Gl.glDisable(Gl.GL_SCISSOR_TEST);
        Ant.Invalidate();
    }
```

Розрахунок температурного поля

Метод Matrix_Create розраховує температурне поле за допомогою метода кінцевих різниць. Детальніше про це розповідається у розділі 3 кваліфікаційної роботи. Алгоритм цього методу виглядає так:

1. Ввод $N_x, N_y, t_{\text{конечное}}, L, H, \lambda, \rho, c, T_h, T_c, T_0$.

2. Визначаємо розрахункові шаги сітки по просторовим координатам

$$h_x = \frac{L}{N_x - 1}, h_y = \frac{H}{N_y - 1}.$$

3. Визначаємо розрахунковий шаг сітки по часу $\tau = \frac{t_{\text{конечное}}}{100}$.

4. Вводимо початкове поле температури $T_{i,j}^0 = T_0, \begin{matrix} i = 1, \dots, N_x; \\ j = 1, \dots, N_y \end{matrix}$.

5. time=0.

6. ЯКЩО $time \geq t_{\text{конечное}}$

ТО

7. Збільшуємо змінну часу на шаг τ:

time=time+ τ

8. Розраховуємо прогоні коефіцієнти в 1-ому вузлі різницевої сітки по осі x α_1 та β_1, використовуючи ліву граничну умову.

9. Розраховуємо остальні прогоночні коефіцієнти α_i та β_i за формулою 3.15.

10. Визначаємо температуру на правій границі $T_{N_x,j}^{n+\frac{1}{2}}$, використовуючи праву граничну умову.

11. Розраховуємо поле температури на проміжному часовому слої $T_{i,j}^{n+\frac{1}{2}}, i = N_x - 1, \dots, 1$ за формулою

ІНАКШЕ

17. Вивід результату

$T_i^{n+1} = \alpha_i \cdot T_{i+1}^{n+1} + \beta_i$.

12. Розраховуємо прогоні коефіцієнти в 1-ому вузлі різницевої сітки по осі y α_1 та β_1 на основі нижньої граничної умови.

13. Розраховуємо остальні прогоночні коефіцієнти α_j та β_j за формулою 3.15.

14. Визначаємо температуру на правій границі T_{i,N_y}^{n+1} на основы верхньої граничної умови.

15. Розраховуємо поле температури на цілому часовому слої $T_{i,j}^{n+1}, i = N_{\text{н}} - 1, \dots, 1$ за формулою $T_i^{n+1} = \alpha_i \cdot T_{i+1}^{n+1} + \beta_i$.

16. Повернутися до шагу 6.

2.3. Опис інтерфейсу користувача

Робота починається з початкового екрану (рис. 2.5).

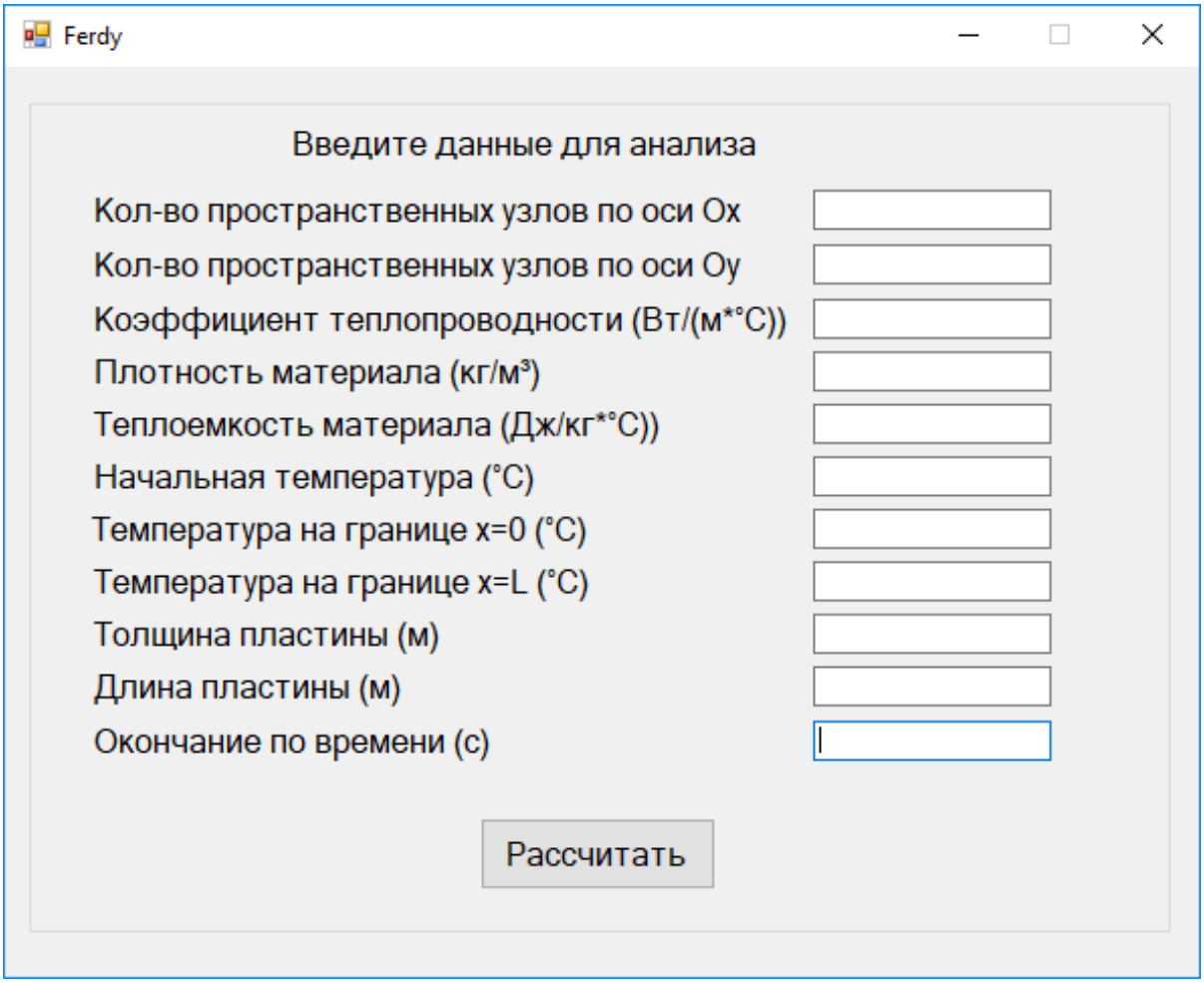

Рис. 2.5. Початковий екран програми

Для подальшої роботи з програмою необхідно ввести початкові дані, як, наприклад, на рис. 2.6.

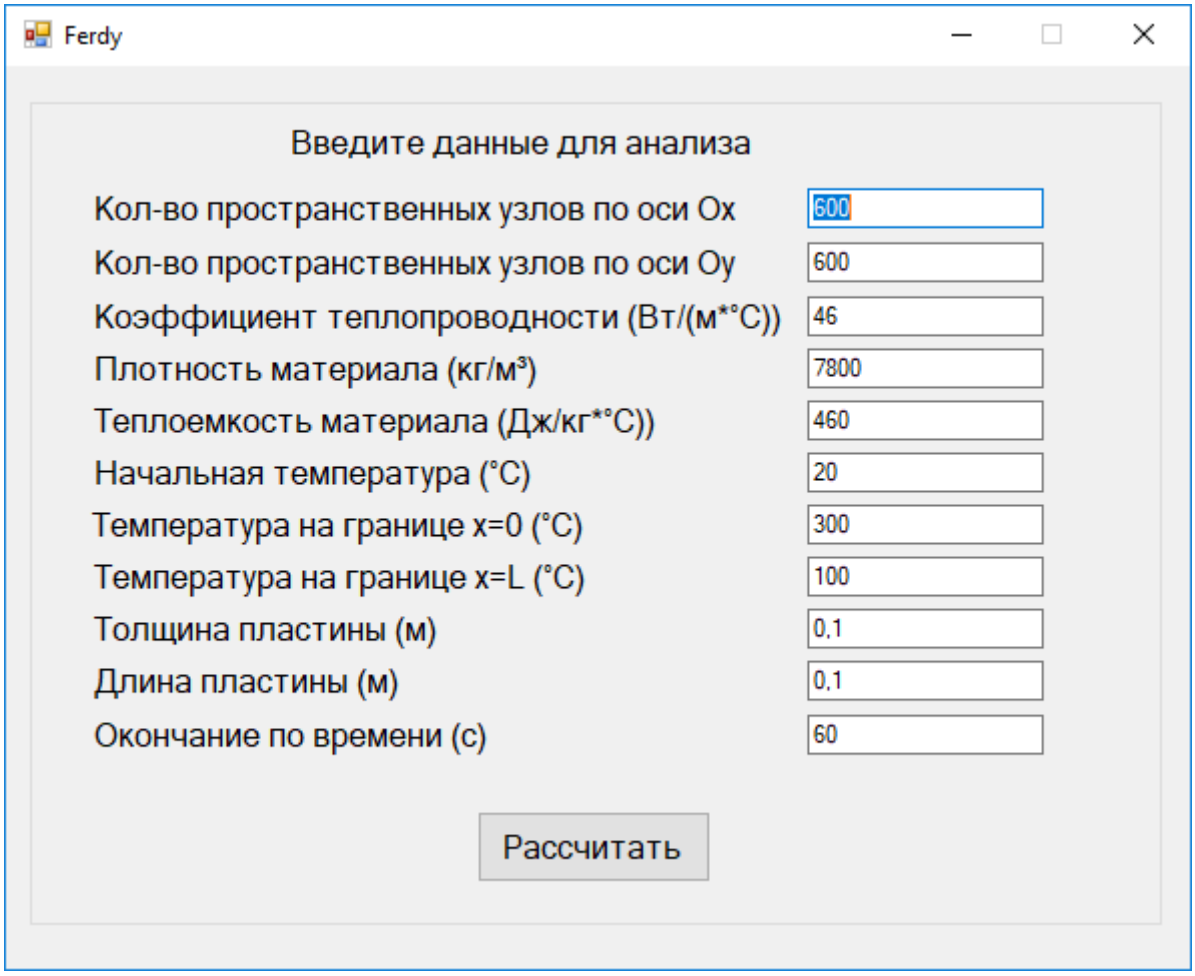

Рис. 2.6. Введення початкових даних

Після натискання кнопки «Рассчитать» з’явиться результат на екрані, прикладом якого є рис. 2.7.

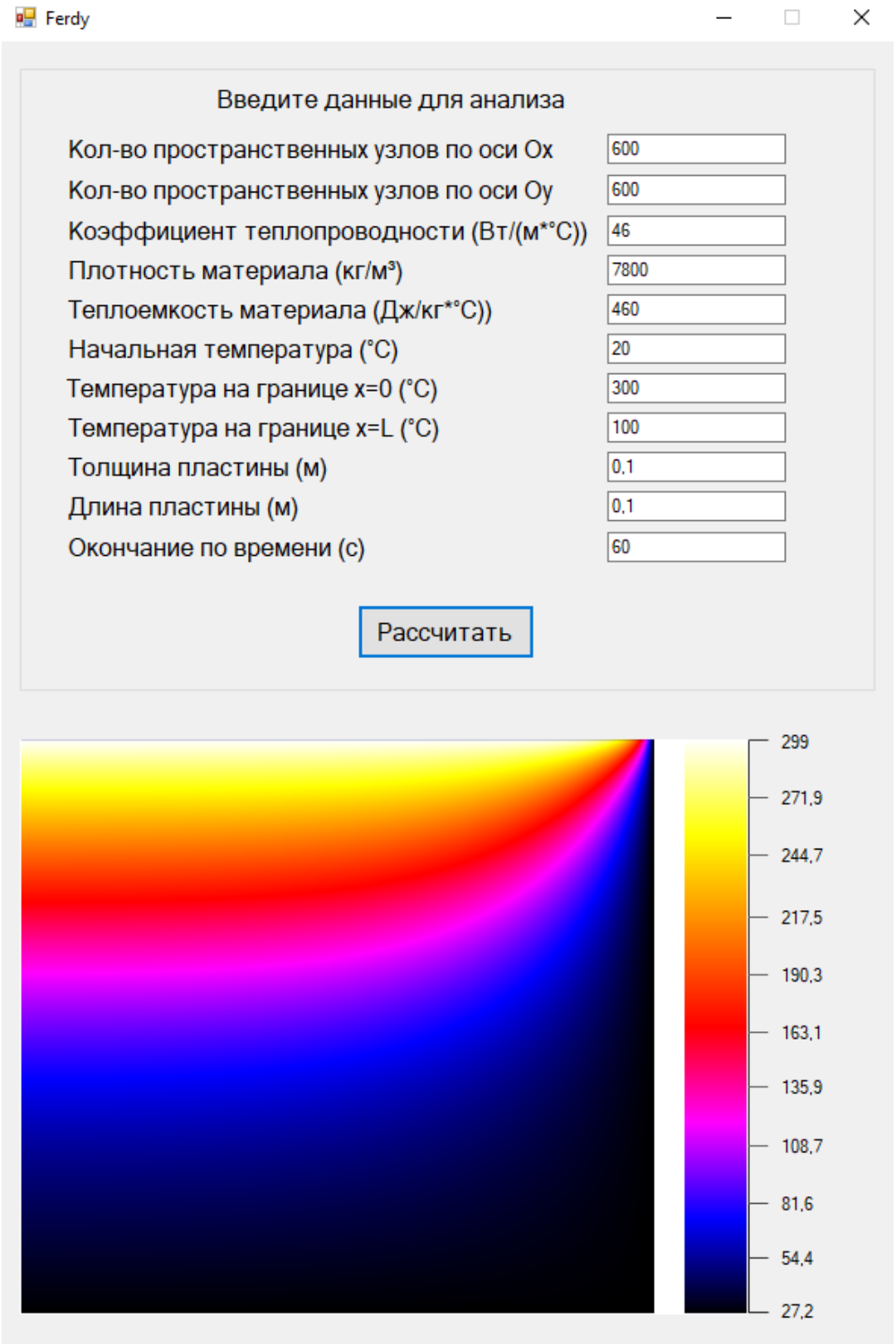

Рис. 2.7. Результат роботи програми

2.4. Висновки до другого розділу

Метою даного розділу було дослідження та аналіз практичної частини з розробки програмного забезпечення для візуалізації та моделювання температурних полів. В результаті досліджень було виявлено, що для практичної реалізації поставленої задачі найкраще підійдуть відкриті бібліотеки OpenGL та Tao Framework на мові програмування C# на платформі .NET Framework.

Через кросплатформеність, продуктивність та широке коло можливостей .NET Famework на даний момент є потужною та однією із найкращих платформ для створення додатків. Саме мова программування C#, яка використовується для розробки застосунків, призначених для широкого спектру систем, найбільш всього розкриває потенціал .NET Framework.

OpenGL є найбільш широко використовуваним і підтримуваним в галузі середовищем для розробки портативних, інтерактивних 2D та 3D графічних додатків. Tao Framework є одним із найкращих шляхів для використання бібліотеки OpenGL в незалежності від платформи. У складі цих двох бібліотек на сьогоднішній момент надходять усі сучасні засоби, які можуть бути надані в ході розробки мультимедіа програмного забезпечення, що і потрібно нам для візуалізації температурних полів.

В результаті був описаний алгоритм візуалізації температурної моделі напівпровідникового кристала, реалізований на мові програмування C#. Розроблений програмний модуль дозволяє зручно та надійно подивитися та проаналізувати розподіл температурних полів за допомогою термограми. Таким чином, це дозволяє ефективно та швидко аналізувати, розраховувати та візуалізовувати температурні поля у вигляді термограми на ПК. Розроблений програмний модуль може бути частиною спеціального програмного забезпечення для моделювання теплових полів електронних компонентів.

РОЗДІЛ 3
ОПТИМІЗАЦІЯ ОБЧИСЛЕНЬ ПРИ ВИРІШЕННІ ПРАКТИЧНИХ ЗАДАЧ

3.1. Математична постановка задачі

У даному випадку будемо розглядувати двухточечні крайові задачі для звичайних диференціальних рівнянь другого порядку. Загальний вигляд таких задач:

$$\begin{aligned} &F(x, y, y', y'') = 0, \qquad x \in [a, b], \\ &\varphi_1(y(a), y'(a)) = 0, \quad \varphi_2(y(b), y'(b)) = 0. \end{aligned} \tag{3.1}$$

де F, φ_1, φ_2 – задані функції певної гладкості.

Найбільш споживані і найкраще вивчені лінійні крайові задачі, тобто завдання виду (3.1), в яких F, φ_1, φ_2 − лінійні функції. Для визначеності будемо вважати основним об'єктом подальшого вивчення лінійну крайову задачу

$$L[y] \coloneqq y'' + p(x)y' + q(x)y = f(x), \quad x \in [a, b], \tag{3.2}$$

$$l_a[y] \coloneqq a_0 y(a) + a_1 y'(a) = A, \tag{3.3}$$

$$l_a[y] \coloneqq \beta_0 y(b) + \beta_1 y'(b) = B, \tag{3.4}$$

де до коефіцієнта крайових умов (3.3), (3.4) ставиться вимога

$$|\alpha_0| + |\alpha_1| \neq 0, \quad |\beta_0| + |\beta_1| \neq 0, \tag{3.5}$$

а функції $p = p(x), q = q(x)$ та $f = f(x)$ в рівнянні (3.2) повинні бути такими, щоб дана задача мала єдине рішення $y = y(x)$ в заданому фунціональном просторі.

Крайові умови (3.3), (3.4) визначають так звану третю або, інакше, змішану крайову задачу для рівняння (3.2), що містить в собі першу (коли $\alpha_1 == \beta_1 = 0$)

або другу (при $\alpha_0 = \beta_0 = 0$) крайові задачі. Залишивши в стороні важливий випадок періодичної крайової задачі, коли замість (3.3), (3.4) виставляються умови $y(a) = y(b)$, $y'(a) = y'(b)$, будемо конструювати методи наближеного рішення змішаної задачі (3.2) − (3.4), лише зрідка виділяючи випадки першої та (або) другого завдань, якщо це виявиться істотним.

Точне рішення крайових задач викликає дуже великі труднощі. Звідси існує підвищений інтерес і велику різноманітність наближених методів вирішення таких завдань. Метод кінцевих різниць є одним з таких методів і був обраний в цій кваліфікаційної роботі для вирішення поставленого завдання розрахунку температурних полів.

3.2. Метод кінцевих різниць

Метод кінцевих різниць заснований на заміні похідних їх наближеним значенням, вираженим через різниці значень функції в окремих дискретних точках - вузлах сітки. Диференціальне рівняння в результаті таких перетворень замінюється еквівалентним співвідношенням в кінцевих різницях, рішення якого зводиться до виконання нескладних алгебраїчних операцій. Остаточний результат рішення дається виразом, за яким значення «майбутнього» потенціалу (температури) в даній точці (вузлі) є функцією часу, її «справжнього» потенціалу та «справжнього» потенціалу суміжних вузлових точок. Повторюваність однакових операцій при розрахунку температурних полів створює великі зручності для застосування сучасної виислітельной техніки, що підвищує у багато разів ефективність.

Наближену заміну першої і другої похідних через різницеві відносини можна провести елементарно наступним чином. Нехай дана функція $y = f(x)$, графік якої представлений на рис. 3.1.

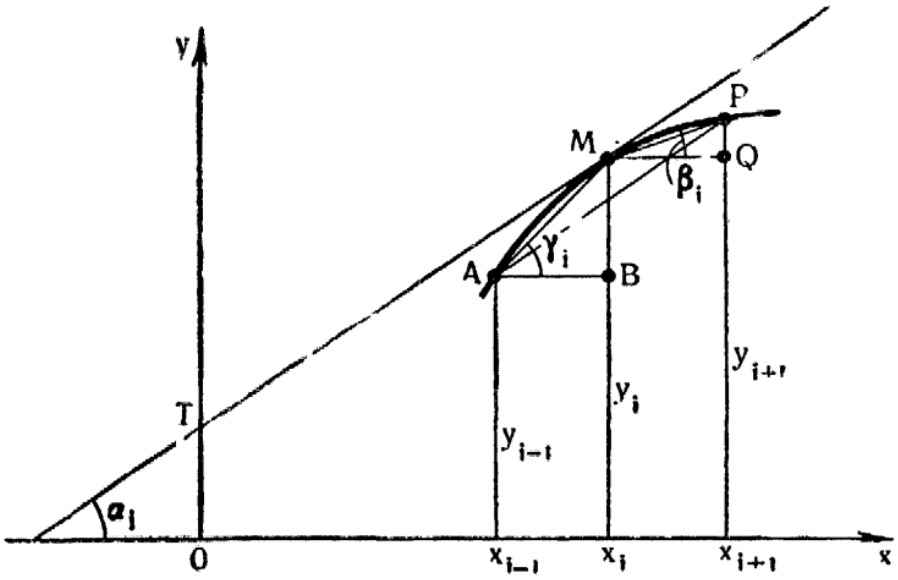

Рис. 3.1. Графік функції $y = f(x)$ та її похідної.

Якщо через α_i позначити кут, утворений з позитивним напрямком осі абсцис дотичної до кривої, проведеної в точці $M(x_i, y_i)$, то похідна функція при $x = x_i$ визначається за формулою

$$y_i' = \tan \alpha_i. \tag{3.6}$$

Візьмемо на кривій дві сусідні точки $A(x_{i-1}, y_{i-1})$ та $P(x_{i+1}, y_{i+1})$ так, щоб різниці $x_i - x_{i-1} = x_{i+1} - x_i = h$ були б досить малі, і наближено замінимо α_i на β_i чи γ_i (або розглянемо замість дотичної MT одну з січних MP чи AM). З цього виходить, що

$$y_i' \approx tg\beta_i = \frac{QP}{MQ} = \frac{y_{i+1} - y_i}{h}, \tag{3.7}$$

або

$$y_i' \approx tg\gamma_i = \frac{BM}{AB} = \frac{y_i - y_{i-1}}{h}. \tag{3.8}$$

Якщо ж кутовий коефіцієнт дотичної МТ наближено замінити кутовим коефіцієнтом січної АР, то

$$y_i' \approx \frac{y_{i+1} - y_{i-1}}{2h}. \tag{3.9}$$

Праві частини формул (3.7) − (3.9) називаються відповідно: різницевим відношенням вперед, різницевим відношенням назад і симетричним різницевим відношенням.

Наближене значення другої похідної y_i'' функції $y = f(x)$ при $x = x_i$ може бути отримано, якщо замінити криву на ділянці АР ламаною лінією АМР, що має в точці М два нахилу, тобто

$$y_i' \approx \frac{1}{h}\left(\frac{y_{i+1} - y_i}{h} - \frac{y_i - y_{i-1}}{h}\right) = \frac{y_{i+1} - 2y_i + y_{i-1}}{h^2}. \tag{3.10}$$

Варто сказати, що наведені формули (3.7) − (3.10) для заміни похідних різницевих відносинами не є єдино можливими. Часом потрібно проводити інші заміни, проте при чисельному інтегруванні рівнянь теплопровідності найбільш часто застосовують саме ці формули.

Метод кінцевих різниць повзоляет вирішувати системи диференціальних рівнянь теплопровідності як при постійних, так і при змінних коефіцієнтах, а також одновимірні, двох- і тривимірні задачі.

3.2.1. Практична реалізація МКР

Проаналізуємо процес теплопереносу у пластині (рис. 3.2).

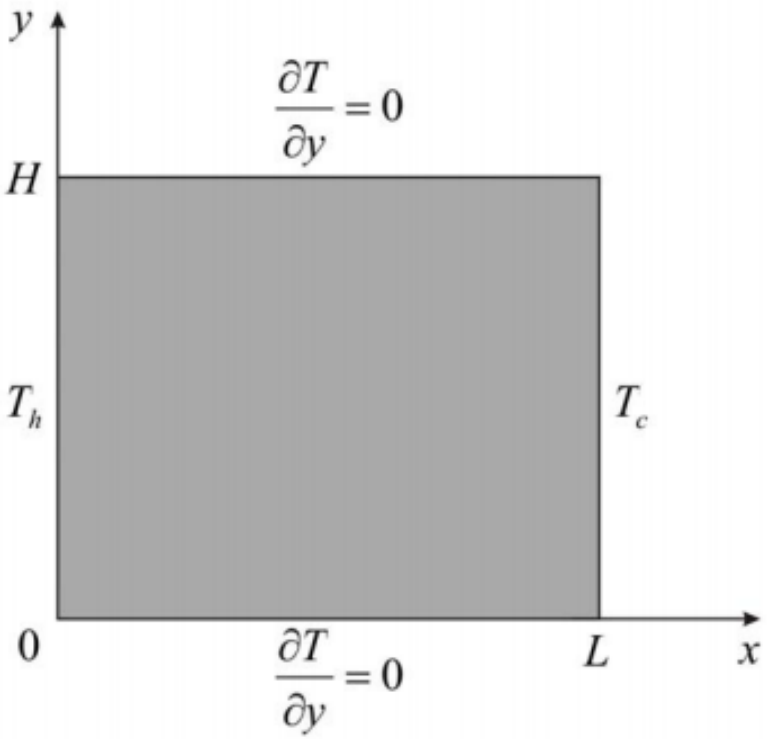

Рис. 3.2. Область вирішення.

Для рішення даної задачі маємо пластину з довжиною L та висотою H. Горизонтальні границі є адіабатичними, а на вертикальних границях підтримуються постійні температури T_h та T_c, при цьому початкова температура області вирішення – T_0.

Математична постановка задачі буде мати вигляд:

$$\rho c \frac{\partial T}{\partial t} = \lambda \left(\frac{\partial^2 T}{\partial x^2} + \frac{\partial^2 T}{\partial y^2} \right), \begin{vmatrix} 0 < x < L \\ 0 < y < H \end{vmatrix} \tag{3.11}$$

Початкові і граничні умови запишуться в такий спосіб:

$$
\begin{aligned}
&t = 0: T = T_0, 0 \le x \le L, 0 \le y \le H \\
&x = 0: T = T_h, t > 0 \\
&x = L: T = T_c, t > 0 \\
&y = 0: \frac{\partial T}{\partial y} = 0, t > 0 \\
&y = H: \frac{\partial T}{\partial y} = 0, t > 0
\end{aligned}
\tag{3.12}
$$

Для апроксимації диференціального рівняння (3.11) різницевим введемо просторово-часову сітку з координатами $x_i = (i-1) \cdot h_x, \quad y_j = (j-1) \cdot h_y, \quad t_n = n \cdot \tau$, де h_x, h_y - кроки сітки за координатами x, y відповідно; τ - крок за часом; $i = \overline{1, N_x}$; $j = \overline{1, N_y}$; $n = \overline{0, K}$. Тобто вся розрахункова область (рис. 3.2) покривається сіткою (рис. 3.3).

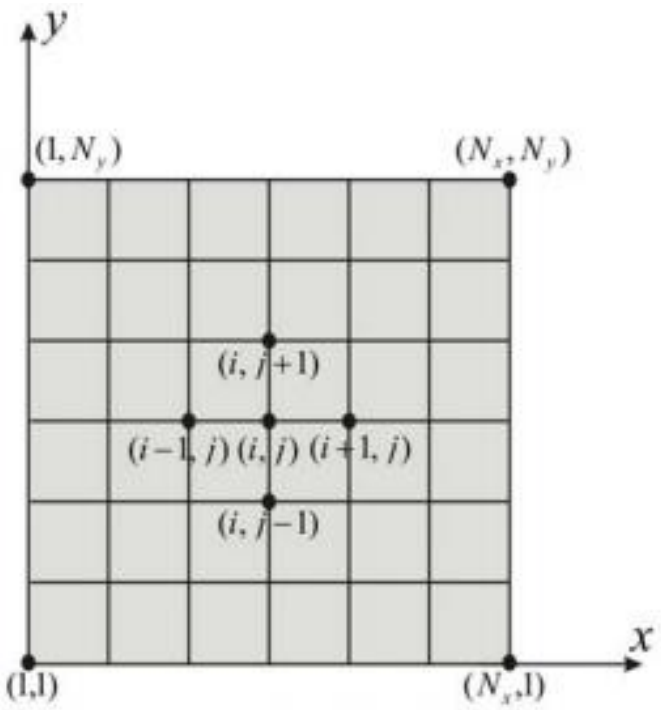

Рис. 3.3. Різницева сітка області рішення

Введемо наступне позначення: $T(x_i, y_j, t_n) = T_{i,j}^n$.

Дискретизацию рівняння (3.11) будемо проводити на основі локально одновимірної схеми А.А. Самарського, яка є абсолютно стійкою і має властивість сумарною апроксимації. Сутність цього підходу полягає в тому, що

крок за часом реалізується в два етапи - на проміжному часовому кроці проводимо дискретизацию двовимірного рівняння (3.11) тільки в напрямку осі *x* і отримуємо одновимірний рівняння, після його рішення проводимо знову дискретизацию рівняння (3.11), але вже в напрямку осі *y* і, вирішуючи отримане одномірне рівняння, визначаємо поле температури на цілому кроці по часу.

Отже:

$$\rho \cdot c \cdot \frac{T_{i,j}^{n+\frac{1}{2}} - T_{i,j}^{n}}{\tau} = \lambda \cdot \left(\frac{T_{i+1,j}^{n+\frac{1}{2}} - 2 \cdot T_{i,j}^{n+\frac{1}{2}} + T_{i-1,j}^{n+\frac{1}{2}}}{h_x^2} \right), \tag{3.13}$$

$$\rho \cdot c \cdot \frac{T_{i,j}^{n+1} - T_{i,j}^{n+\frac{1}{2}}}{\tau} = \lambda \cdot \left(\frac{T_{i,j+1}^{n+1} - 2 \cdot T_{i,j}^{n+1} + T_{i,j-1}^{n+1}}{h_y^2} \right). \tag{3.14}$$

Різницеві рівняння (3.13), (3.14) зводяться до стандартного трехдіагональной увазі і вирішуються послідовно методом прогонки. Спочатку для всієї області вирішується рівняння (3.13), після того як його рішення буде знайдено, переходять до вирішення рівняння (3.14).

Розглянемо рішення рівняння (3.13) методом прогонки. Наведемо це рівняння до виду $A_i T_{i+1,j}^{n+\frac{1}{2}} - B_i T_{i,j}^{n+\frac{1}{2}} + C_i T_{i-1,j}^{n+\frac{1}{2}}$. Тоді коефіцієнти A_i, B_i, C_i візьмуть вигляд:

$$A_i = C_i = \frac{\lambda}{h_x^2}, B_i = \frac{2 \cdot \lambda}{h_x^2} + \frac{\rho \cdot c}{\tau}, F_i = -\frac{\rho \cdot c \cdot T_{i,j}^{n}}{\tau}. \tag{3.15}$$

Для визначення прогоночних коефіцієнтів по співвідношенню

$$\alpha_i = \frac{A_i}{B_i - C_i \cdot \alpha_{i-1}}, \beta_i = \frac{C_i \cdot \beta_{i-1} - F_i}{B_i - C_i \cdot \alpha_{i-1}}. \tag{3.16}$$

необхідно знайти α_1 і β_1 з лівої граничної умови. Далі визначаючи значення $T_{N,j}^{n+\frac{1}{2}}$ з правої граничної умови, знаходять поле температури $T_{N,j}^{n+\frac{1}{2}}$ на проміжному часовому шарі за формулою $T_i^{n+1} = \alpha_i \cdot T_{i+1}^{n+1} + \beta_i$. Після цього приступають до вирішення рівняння (3.14). Етапи рішення рівняння (3.14) аналогічні рішенню рівняння (3.13).

3.3. Висновки до третього розділу

Метою даного розділу було дослідити на практиці ефективність вирішення поставленої задачі методом кінцевих різниць. Як було вже досліджено в цьому розділі, метод кінцевих різниць заснований на заміні похідних їх наближеним значенням, вираженим через різниці значень функції в окремих дискретних точках - вузлах сітки. В результаті таких перетворень диференціальне рівняння замінюється еквівалентним співвідношенням в кінцевих різницях, рішення якого зводиться до виконання нескладних алгебраїчних операцій. Таким чином, кінцевий результат рішення дається виразом, за яким значення «майбутнього» потенціалу (температури) в даній точці (вузлі) є функцією часу, її «справжнього» потенціалу та «справжнього» потенціалу суміжних вузлових точок.

В результаті виходить повторюваність однакових операцій при розрахунку полів температури, що в свою чергу створює великі зручності для застосування в сучасних ЕОМ, що підвищує у багато разів ефективність розрахунків. При цьому в результаті багатьох досліджень і розрахунків стало відомо, що з усіх наявні зараз методів чисельного рішення задач теплопровідності та моделювання, метод кінцевих різниць є найбільш цінним і ефективним. В умовах як і все більшого попиту на рішення подібних задач, так і збільшення складності та розмірів відповідних систем, вищевказані переваги даного методу ще більше підкреслюють правильність та актуальність вибору.

РОЗДІЛ 4
ЕКОНОМІКА

4.1. Визначення трудомісткості розробки програмного забезпечення

Одним з головних етапів при розробці ПЗ є визначення трудомісткості та розрахунок витрат на створення програмного продукту. В даному розділі буде продемонстровано приклад розрахунку витрат на розробку програми з паралельними обчисленнями.

Початкові дані:

1. передбачуване число операторів програми – 750;

2. коефіцієнт складності програми – 1,8;

3. коефіцієнт корекції програми в ході її розробки – 0,1;

4. годинна заробітна плата програміста– 75 грн/год;

5. коефіцієнт збільшення витрат праці в наслідок недостатнього опису задачі – 1,2;

6. коефіцієнт кваліфікації програміста, обумовлений від стажу роботи з даної спеціальності – 1,0;

7. вартість машино-години ЕОМ – 18 грн/год.

Нормування праці в процесі створення ПЗ істотно ускладнено в силу творчого характеру праці програміста. Тому трудомісткість розробки ПЗ може бути розрахована на основі системи моделей з різною точністю оцінки.

Трудомісткість розробки ПЗ можна розрахувати за формулою:

$$t = t_o + t_u + t_a + t_n + t_{отл} + t_д \text{ , людино-годин,} \qquad (4.1)$$

де t_o- витрати праці на підготовку й опис поставленої задачі (приймається 70 людино-годин);

t_u - витрати праці на дослідження алгоритму рішення задачі;

t_a - витрати праці на розробку блок-схеми алгоритму;

t_n - витрати праці на програмування по готовій блок-схемі;

$t_{отл}$ -витрати праці на налагодження програми на ЕОМ;

$t_д$ - витрати праці на підготовку документації.

Складові витрати праці визначаються через умовне число операторів у програмногму забезпеченні, яке розробляється.

Умовне число операторів (підпрограм):

$$Q = q \cdot C \cdot (1 + p), \tag{4.2}$$

де q - передбачуване число операторів (750);

C - коефіцієнт складності програми (1,8);

p - коефіцієнт корекції програми в ході її розробки (0,1).

Звідси умовне число операторів в програмі:

$Q = 1{,}8 \cdot 750 \cdot (1 + 0{,}1) = 1485$ людино-години,

Витрати праці на вивчення опису задачі t_u визначається з урахуванням уточнення опису і кваліфікації програміста:

$$t_u = \frac{Q \cdot B}{(75..85) \cdot k}, \text{ людино-годин,} \tag{4.3}$$

де B - коефіцієнт збільшення витрат праці внаслідок недостатнього опису задачі;

k - коефіцієнт кваліфікації програміста, обумовлений від стажу роботи з даної спеціальності. При стажі роботи від 2 до 3 років він складає 1,0.

Приймемо збільшення витрат праці внаслідок недостатнього опису завдання не більше 50% (B = 1,2). З урахуванням коефіцієнта кваліфікації k = 1,0, отримуємо витрати праці на вивчення опису завдання:

$$t_u = \frac{1485 * 1,2}{80 * 1,0} = 22,27 \text{ людино-годин}$$

Витрати праці на розробку алгоритму рішення задачі визначаються за формулою:

$$t_a = \frac{Q}{(20 \ldots 25) \cdot k} \text{ , людино-годин,} \qquad (4.4)$$

де Q – умовне число операторів програми;

k – коефіцієнт кваліфікації програміста.

Підставивши відповідні значення в формулу (4.4), отримаємо:

$$t_a = \frac{1485}{22 * 1,0} = 67,5 \text{ людино-годин.}$$

Витрати на складання програми по готовій блок-схемі:

$$t_n = \frac{Q}{(20 \ldots 25) \cdot k} \text{ , людино-годин.} \qquad (4.5)$$

$$t_n = \frac{1485}{22 * 1,0} = 67,5 \text{ людино-годин.}$$

Витрати праці на налагодження програми на ЕОМ:

$$t_{отл} = \frac{Q}{(4..5) \cdot k} \text{ , людино-годин.} \quad (4.6)$$

$$t_{отл} = \frac{1485}{4 * 1,0} = 371,25 \text{ людино-годин.}$$

Витрати праці на підготовку документації визначаються за формулою:

$$t_{д} = t_{др} + t_{до} \text{ , людино-годин,} \quad (4.7)$$

де $t_{др}$-трудомісткість підготовки матеріалів і рукопису:

$$t_{др} = \frac{Q}{(15..20) \cdot k} \text{ , людино-годин,} \quad (4.8)$$

$t_{до}$ - трудомісткість редагування, печатки й оформлення документації:

$$t_{до} = 0,75 \cdot t_{др} \text{ , людино-годин.} \quad (4.9)$$

Підставляючи відповідні значення, отримаємо:

$$t_{др} = \frac{1485}{17 * 1,0} = 72,35 \text{ людино-годин.}$$

$$t_{до} = 0,75 \cdot 72,35 = 65,51 \text{ людино-годин.}$$

$$t_{д} = 72,35 + 65,51 = 137,86 \text{ людино-годин.}$$

Повертаючись до формули (3.1), отримаємо повну оцінку трудомісткості розробки програмного забезпечення:

t = 70 + 22,27 + 67,5 + 67,5 + 371,25 + 137,86 = 736,38 людино-годин.

4.2. Розрахунок витрат на створення програми

Витрати на створення ПЗ $К_{ПО}$ включають витрати на заробітну плату виконавця програми $З_{ЗП}$ і витрат машинного часу, необхідного на налагодження програми на ЕОМ ($З_{МВ}$):

$$К_{ПО} = З_{ЗП} + З_{МВ}, \text{ грн.} \quad (4.10)$$

Заробітна плата виконавців визначається за формулою:

$$З_{ЗП} = t \cdot С_{ПР}, \text{ грн,} \quad (4.11)$$

де t - загальна трудомісткість, людино-годин;

$С_{ПР}$ - середня годинна заробітна плата програміста, грн/година

З урахуванням того, що середня годинна зарплата програміста становить 75 грн / год, отримуємо:

$З_{ЗП}$ = 736,38 · 75 = 55 228,5 грн.

Вартість машинного часу, необхідного для налагодження програми на ЕОМ, визначається за формулою:

$$З_{мв} = t_{отл} \cdot С_{мч}, \text{ грн,} \quad (4.12)$$

де $t_{отл}$ - трудомісткість налагодження програми на ЕОМ, год;
$C_{мч}$ - вартість машино-години ЕОМ, грн/год (18 грн/год).

Підставивши в формулу (4.12) відповідні значення, визначимо вартість необхідного для налагодження машинного часу:

$$З_{мв} = 371{,}25 \cdot 18 = 6682{,}5 \text{ грн.}$$

Звідси витрати на створення програмного продукту:

$$К_{ПО} = 55\ 228{,}5 + 6682{,}5 = 61\ 911 \text{ грн.}$$

Очікуваний період створення ПЗ:

$$T = \frac{t}{B_k \cdot F_p}, \text{ міс.} \qquad (4.13)$$

де B_k - число виконавців (дорівнює 1);
F_p - місячний фонд робочого часу (при 40 годинному робочому тижні F_p=176 годин).

$$T = \frac{736{,}38}{1 * 176} \approx 4{,}18 \text{ міс.}$$

4.3. Маркетингові дослідження ринку збуту розробленого програмного продукту

В результаті дослідження області моделювання та візуалізації теплових процесів була представлена концепція програмного забезпечення для таких цілей, яка б надавала користувачу зручний та надійний шлях подивитися та

проаналізувати розподіл температурних полів за допомогою термограми. При цьому розроблене ПЗ має бути зрозумілим при використанні і не вимагати багато часу для ознайомлення з ним, а також можна адаптувати під любі цілі, пов’язані з візуалізацією температурних полів.

Таким чином, розроблена програма дозволяє ефективно та швидко аналізувати, розраховувати та візуалізовувати температурні поля у вигляді термограми на ПК. Головна різниця від схожих продуктів полягає в удосконаленні моделювання теплових процесів в незалежності від різних конфігурацій обчислювальних систем, що дозволяє робити розрахунки та моделювання з більшою точністю та швидкістю та меньшою вартістю.

Задовольнити запити споживачів - непроста задача. Насамперед, потрібно добре вивчити споживача, тобто відповісти на питання: хто купує, яку кількість, за якою ціною, з якою метою, для задоволення яких потреб, де купує, забезпечити, якщо це необхідно, сервіс. Зважаючи на те, що ринок програмного забезпечення досить широкий, необхідно максимально наблизити розроблений програмний продукт до вимог замовника.

Основними покупцями ПЗ є промислові підприємства тих галузей промисловості, де потрібен строгий контроль за температурою при виробництві чи експлуатації продукції (виробництво електроніки та електроних приборів, сталелитійний процес тощо). Так, наприклад, промисловим підприємствам необхідно вести облік та стежити за температурним станом і параметрами виробничого обладнання, робочих агрегатів, виробів, що виділяють велику кількість тепла, тощо. З метою зменшення грошових витрат, підприємства широко використовують вітчизняні програмні продукти в зв'язку з їх дешевиною, адаптованістю до місцевих умов господарювання і простотою сервісного обслуговування.

Дослідження ринку на даному напрямку показали, що аналогічні розробки присутні на ринку, але використовуються обмежено і локально. Цінову політику конкурентів складно оцінити, тому що при реалізації аналогічного програмного продукту на ринку використовується прямий маркетинг між виробником і

споживачем, тобто в такому випадку у кожного замовника буде разроблено конкретне ПЗ на своє замовлення. З огляду на стрімке зростання високотехнологічного виробництва, особливо виробництва електроніки, все більше збільшується попит на моделювання та візуалізацію теплових процесів в виробленої продукції або виробничих процесах. Разом з тим пропозиція в цій області дуже обмежена, тому в найближчі роки такі розробки є перспективними і затребуваними.

Так як розробляється складна програмна продукція, яка потребує спеціального налагодження та обслуговування, то будемо використовувати прямий маркетинг між виробником і споживачем. Конкретне ПЗ в такому випадку розробляється за замовленням споживача. У цьому випадку споживач знаходиться у більш вигідній ситуації, оскільки відпадає необхідність у послугах посередника, а також є можливість у разі потреби доробки й обслуговування ПЗ.

4.4. Оцінка економічної ефективності впровадження програмного забезпечення

Впровадження програмного забезпечення може дозволити підприємству:

- скоротити кількість працівників (економія по заробітній платі);
- скоротити час та вартість виробництва продукції за рахунок покращення виробничого процесу;
- зменшити шанс виникнення браку на виробництві;
- підвищити продуктивність праці;
- поліпшити якість графічного відображення даних моделювання теплових процесів.

Так як є необхідність реальних інвестицій та одержання економії коштів при впровадженні розробленого програмного продукту, то потрібно зробити розрахунок чистих грошових надходжень.

Економічна оцінка ефективності пропонованого впровадження оцінена за системою показників, які використовуються у міжнародній і вітчизняній практиці. При оцінці економічної ефективності використані наступні показники:

- чиста поточна вартість (NPU);
- строк окупності капітальних вкладень;
- індекс прибутковості;
- коефіцієнт ефективності інвестицій.

При ухваленні рішення стосовно доцільності впровадження проекту необхідно враховувати значення всіх показників, тому що кожен показник несе свій обсяг інформації, і тільки всі вони в сукупності можуть дати повне уявлення про реальну ефективність.

Такий розрахунок показаний у таблиці 4.1.

Таблиця 4.1.

Показники, грн	За роками						Усього за 5 років	Середнє за 5 років
	0	1	2	3	4	5		
1. Інвестиції на ПЗ	61911	-	-	-	-	-	61911	12382
2. Витрати до впровадження ПЗ	-	130000	112500	112500	112500	130000	597500	119500
- на щорічну перевірку на погрішність	-	28500	28500	28500	28500	28500	142500	28500
- оплату праці бригади КВП	-	75000	75000	75000	75000	75000	375000	75000
- на щорічну перевірку держстандарту	-	6750	6750	6750	6750	6750	33750	6750
- на придбання пірометра	-	17500	-	-	-	17500	35000	7000
- на електроенергію	-	2250	2250	2250	2250	2250	11250	2250
3. Витрати після впровадження ПО	-	285500	30500	30500	30500	30500	407500	81500

Продовження таблиці 4.1

Показники, грн	За роками						Усього за 5 років	Середнє за 5 років
	0	1	2	3	4	5		
- на придбання та облаштування спеціального робочого місця	-	255000	-	-	-	-	255000	51000
-на щорічну перевірку	-	4000	4000	4000	4000	4000	20000	4000
-на оплату праці оператора	-	25000	25000	25000	25000	25000	125000	25000
-на електроенергію	-	1500	1500	1500	1500	1500	7500	1500
4. Економія	-	-155500	82000	82000	82000	99500	190000	38000
5. Амортизація	-	2150	1410	-	-	-	3560	712
6. Чисті грошові надходження	-	-153350	83410	82000	82000	99500	193560	38712
7. Коефіцієнт дисконтування	-	0,901	0,888	0,773	0,691	0,636	-	-
Дисконтні грошові надходження	-	-138168	74068	63386	56662	63282	119230	23846

Чиста поточна вартість доходів: NPU = 119230 − 61911 = 57319 грн. (>0)

Строк окупності: Т = 61911/23846 = 2,6 рока.

Індекс прибутковості: ІП = 119230/61911 = 1,92.

Показник економічної ефективності (NPU - чиста поточна вартість доходів за роки реалізації впровадження (3-5 років)) складе 57319 грн, тобто відповідає умовам ефективності, тому що NPU > 0.

Середній строк окупності капвкладень складе 2,6 року.

Індекс прибутковості за 5 років складе 1,92, тобто ІП > 1, проект варто прийняти.

Таким чином, показник ефективності свідчить про те, що дане впровадження є економічно вигідним.

ВИСНОВКИ

У процесі дослідження були отримані наступні результати. При вивченні основ процесів теплообміну було з’ясовано, що виділяють три основних способи перенесення тепла:

1. Теплопровідність – передача тепла від однієї частини тіла до іншої або від одного тіла до іншого, що знаходиться в зіткненні з першим.

2. Конвекція - перенесення, обумовлений просторовим переміщенням речовини і спостерігається в рухомих середовищах, таких як рідини і гази.

3. Випромінювання - перенесення енергії у вигляді електромагнітних хвиль.

Так як основним шляхом переносу тепла для головного предмету дослідження є саме теплопровідність, то в цій роботі використовувалися методи рішень задач з теплообміну. Головним із них є ідея методу кінцевих різниць, яка була детально розглянута разом із її реалізацією у розділі 3. Сама ідея має такий вигляд: замість похідних в диференціальному рівнянні використовуються їх кінцеворізницеві апроксимації. При цьому при побудові дискретних апроксимацій крайових диференціальних завдань потрібно прагнути пов'язати дві майже суперечливі цілі: гарна якість апроксимації і ефективне стійке рішення утворюються у своїй алгебраїчних систем.

В роботі також було досліджено математичне моделювання теплових процесів. Теоретичне дослідження процесів теплообміну в даний час в значній мірі базується на їх чисельному моделюванні з використанням ЕОМ, що стало можливим завдяки значному прогресу в розвитку обчислювальних методів вирішення завдань для рівнянь в приватних похідних і збільшення потужності сучасних обчислювальних машин.

У сучасних умовах виробництва математичне моделювання грає дуже важливу роль, виступаючи як засіб оптимізації та прогнозування. Моделювання теплових полів є, зокрема, однією з найбільш актуальних проблем при проектуванні електронних компонентів. При створенні спеціальних засобів

розрахунку теплових параметрів і моделювання теплових процесів, розробник зіткнеться з завданням візуалізації теплових полів.

В результаті була представлена концепція програмного забезпечення для таких цілей, яка б надавала користувачу зручний та надійний шлях подивитися та проаналізувати розподіл температурних полів за допомогою термограми. При цьому розроблене ПЗ має бути зрозумілим при використанні і не вимагати багато часу для ознайомлення з ним, а також можна адаптувати під любі цілі, пов’язані з візуалізацією температурних полів. Таким чином, розроблена програма дозволила би ефективно та швидко аналізувати, розраховувати та візуалізовувати температурні поля у вигляді термограми на ПК. Результатом стала розробка відповідного програмного забезпечення, яка задовольняє вище написаним вимогам.

Отриманого практичного досвіду буде корисно для більш глибокого розуміння суті моделювання та візуалізації теплових процесів, їх необхідності і можливостей, а також полегшить перехід до вирішення складніших завдань.

Проведене дослідження буде корисно для науковців, розробників, аспірантів і студентів, що спеціалізуються або цікавляться проблематикою моделювання процесів теплообміну.

ПЕРЕЛІК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Кузнецов Г.В., Шеремет М.А. Разностные методы решения задач теплопроводности: учебное пособие. / Г.В. Кузнецов, М.А. Шеремет. – Томск: Изд-во ТПУ, 2007. – 172 с.

2. Аралов М.Н., Барабанов А.В., Гребенникова Н.И.. Визуализация тепловых полей с использованием библиотек OpenGL и Tao Framework – 2008 – С. 35-39.

3. Барабанов В.Ф. Математические методы моделирования тепловых полей в трехмерной сборке интегральных схем [Текст] / В.Ф. Барабанов, М.Н. Аралов, Н.И. Гребенникова // Вестник Воронежского государственного технического университета. - 2013. - Т. 9, № 6-3. – С. 55-57.

4. Петухов Б.С., Генин Л.Г., Ковалев С.А., Соловьев С.Л. Теплообмен в ядерных энергетических установках. – М.: Изд-во МЭИ, 2003. – 548 с.

5. Лыков А.В. Теория теплопроводности. – М.: Высшая школа, 1967. – 600 с.

6. Самарский А.А., Вабищевич П.Н. Вычислительная теплопередача. – М.: Едиториал УРСС, 2003. – 782 с.

7. Постанова Кабінету Міністрів України від 26.04.2015 №266 «Перелік галузей знань і спеціальностей, за якими здійснюється підготовка здобувачів вищої освіти».

8. Демидович Б. П., Марон И. А. Основы вычислительной математики. – М.: Наука, 1966. – 695 с.

9. Березин И.С., Жидков Н.П. Методы вычислений. – М.: Наука, 1962. – Т. 1. – 464 с.

10. Берковский Б.М., Ноготов Е.Ф. Разностные методы исследования задач теплообмена. – Минск: Наука и техника, 1976. – 141 с.

11. Копченова Н. В., Марон И. А. Вычислительная математика в примерах и задачах. – М.: Наука, 1972. – 367 с.

12. Березин И.С., Жидков Н.П. Методы вычислений. – М.: Наука, 1962. – Т. 2. – 639 с.

13. Крылов В.И., Бобков В.В., Монастырный П.И. Вычислительные методы. – М.: Наука, 1976. – Т. 1. – 302 с.

14. C# Programming Guide. [Электронный ресурс] - Режим доступа: https://docs.microsoft.com/en-us/dotnet/csharp/programming-guide/index

15. Хейлсберг А., Торгерсен М., Вилтамут С., Голд П. Язык программирования C#. Классика Computers Science. 4-е издание = C# Programming Language (Covering C# 4.0), 4th Ed. — СПб.: «Питер», 2012. — 784 с. — ISBN 978-5-459-00283-6.

16. Грэхем И. Объектно-ориентированные методы. Принципы и практика = Object-Oriented Methods: Principles & Practice. — 3-е изд. — М.: «Вильямс», 2004. — С. 880. — ISBN 0-201-61913-X.

17. Weisfeld M. The Object-Oriented Thought Process. — Fourth Edition. — Addison-Wesley Professional, 2013. — 336 с. — ISBN 978-0-321-86127-6.

18. Уроки OpenGL. Программирование 3D графики. [Электронный ресурс] - Режим доступа: http://esate.ru

19. Херн Д., Бейкер М.П. Компьютерная графика и стандарт OpenGL = Computer Graphics with OpenGL. — 3-е изд. — М.: Вильямс, 2005. — 1168 с. — ISBN 5-8459-0772-1.

20. Рендольф Н., Гарднер Д., Минутилло М., Андерсон К. Visual Studio 2010 для профессионалов = Professional Visual Studio 2010. — М.: «Диалектика», 2011. — С. 1184. — ISBN 978-5-8459-1683-9.

21. Макки А. Введение в .NET 4.0 и Visual Studio 2010 для профессионалов = Introducing .NET 4.0: with Visual Studio 2010. — М.: «Вильямс», 2010. — С. 416. — ISBN 978-5-8459-1639-6.

22. Методы и алгоритмы моделирования тепловых полей в трехмерной сборке интегральных схем [Текст] / В.Ф. Барабанов, С.Л. Подвальный, А.В. Ачкасов, М.Н. Аралов // Радиотехника. – 2014. - №6 – С. 82-87.

23. Барабанов В.Ф. Структура программной модели цифрового устройства [Текст] / В.Ф. Барабанов, М.Н. Аралов // Информационные технологии моделирования и управления. – 2013. - № 6 (84). - С. 583-589.

24. Брагин Д.М. Математическое моделирование процесса размещения разногабаритных элементов на монтажном пространстве с учетом тепловых полей [Текст] / Д.М. Брагин, В.Ф. Барабанов, А.М. Нужный // Вестник Воронежского государственного технического университета. - 2007.- Т. 3, № 5. – С. 76 –80.

25. Барабанов В.Ф. Интерактивные средства моделирования сложных технологических процессов [Текст] / В.Ф. Барабанов, С.Л. Подвальный. - Воронеж, 2000.

26. Брагин Д.М. Интегрированный программный комплекс моделирования и проектирования электронных средств с учетом тепловых полей [Текст] / Д.М.Брагин, В.Ф. Барабанов, А.М. Нужный // Системы управления и информационные технологии. - 2007. - Т. 29. - № 3. - С.63- 66.

27. Вержбицкий В.М. Основы численных методов – М.: Высш. шк., 2002.

28. Лыков А.В. Теплообмен. Справочник - М.: «Энергия», 1972 - 560 с.: илл.

29. Снеддон И.Н. Преобразование Фурье - М.: Изд-во иностр. лит., 1955.

30. Курант Р., Гильберт Д. Методы математической физики - М.: Изд-во иностр. лит., 1951.

31. Apraci V. Conduction heat transfer - Addison Wesley, 1966 - 550 p.

32. Михайлов М.Д. Нестационарный тепло- и массоперенос в одномерных телах - Минск, ИТМО, 1969

33. Швейдер П. Инженерные проблемы теплопроводности – М.: Изд-во иностр. лит., 1960

34. Лыков А.В. Некоторые аналитические методы решения задач нестационарной теплопроводности. Методы решения нелинейных уравнений нестационарной теплопроводности. - «Изв. АН СССР. Энергетика и транспорт», 1969- №2 – с. 3-27; №5 – с. 109-150.

35. Карслоу Х.С., Егер Д.К. Телпопроводность твердых тел – М.: Изд. «Наука», 1964.

36. Райт мл. Р.С., Липчак Б. OpenGL. Суперкнига = OpenGL SuperBible. — 3 изд. — М.: Вильямс, 2006. — С. 1040. — ISBN 5-8459-0998-8.

37. Энджел Э. Интерактивная компьютерная графика. Вводный курс на базе OpenGL = Interactive Computer Graphics. A Top-Down Approach with Open GL. — 2-е изд. — М.: Вильямс, 2001. — 592 с. — ISBN 5-8459-0209-6.

38. Нейгел К. C# 5.0 и платформа .NET 4.5 для профессионалов = Professional C# 5.0 and .NET 4.5. — М.: «Диалектика», 2013. — 1440 с. — ISBN 978-5-8459-1850-5.

39. Скит Д. C# для профессионалов: тонкости программирования, 3-е издание, новый перевод = C# in Depth, 3rd ed.. — М.: «Вильямс», 2014. — 608 с. — ISBN 978-5-8459-1909-0.

40. Просиз Д. Программирование для Microsoft .NET = Programming Microsoft .NET. — М.: Русская редакция, 2003. — С. 704. — ISBN 5-7502-0217-8.

41. Арсенин В.Я. Математическая физика – М.: «Наука», 1966.

42. Диткин В.А., Прудников А.П. Теплопроводность твердых тел – М.: «Нвука», 1964.

43. Polyanin A. D., Zaitsev V. F. Handbook of Exact Solutions for Ordinary Differential Equations (2nd edition) - Chapman & Hall/CRC Press, Boca Raton, 2003 - ISBN 1-58488-297-2.

44. Алифанов О.М. Обратные задачи теплообмена – М.: Машиностроение, 1988.

45. Карташов Э.М. Аналитические методы в теплопроводности твердых тел – М.: Высшая школв, 1979.

46. Беляев Н.М., Рядно А.А. Методы теории теплопроводности – М.: Высшая школа, 1982.

47. Штеттер Х. Анализ методов дискретизации для обыкновенных дифференциальных уравнений. — Москва: Мир, 1978. — 461 с.

48. Воеводин В.В. Вычислительные основы линейной алгебры – М.: Наука, 1977.

49. Самарский А.А. Введение в численные методы – М.: Наука, 1987.

50. Исаченко В.П., Осипова В.А., Сукомел А.С. Теплопередача – М.: Энергоиздат, 1981.

51. Ладыженская О.А. Краевые задачи математической физики – М.: Наука, 1973.

52. Марчук Г.И. Методы вычислительной математики – Новосибирск, 1973.

ДОДАТОК А

ЛІСТИНГ ПРОГРАМИ

```
using System;
using System.IO;
using System.Threading.Tasks;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Linq;
using System.Text;
using System.Windows.Forms;
using Tao.FreeGlut;
using Tao.OpenGl;
using Tao.Platform.Windows;

namespace DiplomWork
{
    public partial class Form1 : Form
    {
        public Form1()
        {
            InitializeComponent();
            Ant.InitializeContexts();
        }

        private void graphic_init(int n, int m)
        {
            Glut.glutInit();
            Glut.glutInitDisplayMode(Glut.GLUT_RGB |
            Glut.GLUT_DOUBLE | Glut.GLUT_DEPTH);
            // очистка окна
            Gl.glClearColor(255, 255, 255, 1);
            // настройка проекции
            Gl.glMatrixMode(Gl.GL_PROJECTION);
            Gl.glLoadIdentity();
            // настройка 2D ортогональной проекции в
            // соответствии с размерами отображаемой
матрицы
            Glu.gluOrtho2D(0, m, 0, n);
```

```
            //установка объектно-видовой матрицы и
очистка
            Gl.glMatrixMode(Gl.GL_MODELVIEW);
            Gl.glLoadIdentity();
        }

        private void TempToRGB(double T, double Tmax,
double Tmin)
        {
            double Colour_lim = (Tmax - Tmin) / 6;
            double Tblue = Tmin + Colour_lim;
            double Tviolet = Tmin + Colour_lim * 2;
            double Tred = Tmin + Colour_lim * 3;
            double Torange = Tmin + Colour_lim * 4;
            double Tyellow = Tmin + Colour_lim * 5;
            double t = ((T - Tmin) % Colour_lim) /
Colour_lim;
            double R = 0, G = 0, B = 0;
            R = (T <= Tblue) ? 0 :
            (T > Tviolet) ? 1 :
            t;
            G = (T <= Tred) ? 0 :
            (T > Tyellow) ? 1 :
            (T < Torange) ? t / 2 :
            0.5 + t / 2;
            B = (T <= Tmin) ? 0 :
            (T >= Tmin && T < Tblue) ? t :
            (T >= Tblue && T < Tviolet) ? 1 :
            (T >= Tviolet && T < Tred) ? 1 - t :
            (T >= Tred && T < Tyellow) ? 0 :
            (T >= Tyellow && T < Tmax) ? t :
            1;
            Gl.glColor3d(R, G, B);
        }

        private void out_PpenGL(double[,] MatrT, int n,
int m, double Tmax, double Tmin)
        {
            Gl.glEnable(Gl.GL_SCISSOR_TEST);
            Gl.glViewport(0, 0, Ant.Width-70,
Ant.Height);
            Gl.glScissor(0, 0, Ant.Width-70, Ant.Height);
            Gl.glClear(Gl.GL_COLOR_BUFFER_BIT);
            for (int i = 0; i < n - 1; i++)
            {
```

```
                for (int j = 0; j < m - 1; j++)
                {
                    Gl.glBegin(Gl.GL_QUADS);
                    TempToRGB(MatrT[i, j], Tmax, Tmin);
                    Gl.glVertex2d(j + 1, m - i);
                    TempToRGB(MatrT[i + 1, j], Tmax,
Tmin);
                    Gl.glVertex2d(j + 1, m - i - 1);
                    TempToRGB(MatrT[i + 1, j + 1], Tmax,
Tmin);
                    Gl.glVertex2d(j + 2, m - i - 1);
                    TempToRGB(MatrT[i, j + 1], Tmax,
Tmin);
                    Gl.glVertex2d(j + 2, m - i);
                    Gl.glEnd();
                }
            }

            Gl.glViewport(Ant.Width - 70, 0, Ant.Width,
Ant.Height);
            Gl.glScissor(Ant.Width - 70, 0, Ant.Width,
Ant.Height);
            Gl.glClear(Gl.GL_COLOR_BUFFER_BIT);

            Gl.glBegin(Gl.GL_QUADS);
            Gl.glColor3d(255, 255, 255);
            Gl.glVertex3f(Ant.Width - 70, 0, 0);
            Gl.glVertex3f(Ant.Width - 50, 0, 0);
            Gl.glVertex3f(Ant.Width - 50, Ant.Height, 0);
            Gl.glVertex3f(Ant.Width - 70, Ant.Height, 0);
            Gl.glEnd();

            Gl.glViewport(Ant.Width - 50, 0, Ant.Width,
Ant.Height);
            Gl.glScissor(Ant.Width - 50, 0, Ant.Width,
Ant.Height);
            Gl.glClear(Gl.GL_COLOR_BUFFER_BIT);
            double step = (Tmax - Tmin) / n;
            for (int i = 0; i < n - 1; i++)
            {
                Gl.glBegin(Gl.GL_QUADS);
                TempToRGB(Tmax - (i + 1) * step, Tmax,
Tmin);
                Gl.glVertex2d(0, m - i);
                TempToRGB(Tmax - i * step, Tmax, Tmin);
```

```
                Gl.glVertex2d(0, m - i - 1);
                TempToRGB(Tmax - i * step, Tmax, Tmin);
                Gl.glVertex2d(50, m - i - 1);
                TempToRGB(Tmax - (i + 1) * step, Tmax,
Tmin);
                Gl.glVertex2d(50, m - i);
                Gl.glEnd();
            }

            Gl.glFlush();
            Gl.glDisable(Gl.GL_SCISSOR_TEST);
            Ant.Invalidate();
        }

        private double[,] Matrix_Create(int Nx, int Ny,
double lamda, double ro, double c, double T0, double Th,
double Tc, double H, double L, double t_end)
        {
            int i, j;
            double ai, bi, ci, fi,
                a, hx, hy, tau, time;
            double[] alfa = new double[Nx + 1];
            double[] beta = new double[Nx + 1];
            double[,] T = new double[Nx+1, Ny+1];

            hx = L / (Nx - 1);
            hy = H / (Ny - 1);
            a = lamda / (ro * c);
            tau = t_end / 100.0;

            for (i = 0; i < Nx; i++)
                for (j = 0; j < Ny; j++)
                    T[i, j] = T0;

            time = 0;
            while (time < t_end)
            {
                time += tau;
                for (j = 0; j < Ny; j++)
                {
                    alfa[0] = 0.0;
                    beta[0] = Th;
                    for (i = 1; i < Nx - 1; i++)
                    {
                        ai = lamda / (hx * hx);
```

```
                        bi = 2.0 * lamda / (hx * hx) + ro
* c / tau;
                        ci = lamda / (hx * hx);
                        fi = -ro * c * T[i, j] / tau;
                        alfa[i] = ai / (bi - ci * alfa[i
- 1]);
                        beta[i] = (ci * beta[i - 1] - fi)
/ (bi - ci * alfa[i - 1]);
                    }
                    T[Nx-1, j] = Tc;
                    for (i = Nx - 1; i > 0; i--) T[i, j]
= alfa[i] * T[i + 1, j] + beta[i];
                }

                for (i = 1; i < Nx - 1; i++)
                {
                    alfa[0] = 2.0 * a * tau / (2.0 * a *
tau + (hy * hy));
                    beta[0] = (hy * hy) * T[i, 0] / (2.0
* a * tau + (hy * hy));
                    for (j = 1; j < Ny - 1; j++)
                    {
                        ai = lamda / (hy * hy);
                        bi = 2.0 * lamda / (hy * hy) + ro
* c / tau;
                        ci = lamda / (hy * hy);
                        fi = -ro * c * T[i, j] / tau;
                        alfa[j] = ai / (bi - ci * alfa[j
- 1]);
                        beta[j] = (ci * beta[j - 1] - fi)
/ (bi - ci * alfa[j - 1]);
                    }
                    T[i, Ny] = (2.0 * a * tau * beta[Ny -
1] + (hy * hy) * T[i, Ny]) / (2.0 * a * tau * (1.0 -
alfa[Ny - 1]) + (hy * hy));
                    for (j = Ny - 1; j > 0; j--) T[i, j]
= alfa[j] * T[i, j + 1] + beta[j];
                }
            }
            return T;
        }
        private void Matrix_Form(double[,] Matr)
        {
            string writePath =
@"D:\Work\Diplom\DiplomWork\1.txt";
```

```
            try
            {
                using (StreamWriter sw = new
StreamWriter(writePath, false,
System.Text.Encoding.Default))
                {
                    for (int i = 0; i <
Matr.GetLength(0); i++)
                    {
                        for (int j = 0; j <
Matr.GetLength(1); j++)
                        {
                            sw.Write(Matr[i,j]+" ");
                        }
                        sw.WriteLine("");
                    }
                }
            }
            catch (Exception e)
            {
                Console.WriteLine(e.Message);
            }
        }

        private void button1_Click(object sender,
EventArgs e)
        {
            int Nx, Ny, n, m;
            double lamda, ro, c, t_end,
                T0, L, H, Th, Tc,
                min,max;

            Nx = Convert.ToInt32(textBox1.Text);
            Ny = Convert.ToInt32(textBox2.Text);
            lamda = Convert.ToDouble(textBox3.Text);
            ro = Convert.ToDouble(textBox4.Text);
            c = Convert.ToDouble(textBox5.Text);
            T0 = Convert.ToDouble(textBox6.Text);
            Th = Convert.ToDouble(textBox7.Text);
            Tc = Convert.ToDouble(textBox8.Text);
            H = Convert.ToDouble(textBox9.Text);
            L = Convert.ToDouble(textBox10.Text);
            t_end = Convert.ToDouble(textBox11.Text);
```

```
            double[,] T = Matrix_Create(Nx, Ny, lamda,
ro, c, T0, Th, Tc, H, L, t_end);
            Matrix_Form(T);

            n = T.GetLength(0);
            m = T.GetLength(1);
            max = T[0, 0];
            min = T[0, 1];
            for (int i = 0; i < n; i++)
            {
                for (int j = 0; j < m; j++)
                {
                    if (T[i, j] > max) max = T[i, j];
                    if (T[i, j] < min) min = T[i, j];
                }
            }
            graphic_init(n, m);
            out_PpenGL(T,n,m,max,min);

            double graph_step = (max - min) / 11;
            double curr_temp = max;

            label13.Text = Math.Round(curr_temp,
1).ToString();

            curr_temp -= graph_step;
            label14.Text = Math.Round(curr_temp,
1).ToString();

            curr_temp -= graph_step;
            label15.Text = Math.Round(curr_temp,
1).ToString();

            curr_temp -= graph_step;
            label16.Text = Math.Round(curr_temp,
1).ToString();

            curr_temp -= graph_step;
            label17.Text = Math.Round(curr_temp,
1).ToString();

            curr_temp -= graph_step;
            label18.Text = Math.Round(curr_temp,
1).ToString();
```

```
            curr_temp -= graph_step;
            label19.Text = Math.Round(curr_temp,
1).ToString();

            curr_temp -= graph_step;
            label20.Text = Math.Round(curr_temp,
1).ToString();

            curr_temp -= graph_step;
            label21.Text = Math.Round(curr_temp,
1).ToString();

            curr_temp -= graph_step;
            label22.Text = Math.Round(curr_temp,
1).ToString();

            curr_temp -= graph_step;
            label23.Text = Math.Round(curr_temp,
1).ToString();

            this.Height = 900;
        }
    }
}
```

ДОДАТОК Б

ВІДГУК

керівника економічного розділу

на кваліфікаційну роботу магістра

на тему: «Методи, алгоритми та інформаційна технологія розрахунку і візуалізації температурних полів»

студента групи 122м-19-1 Яриловця Віктора Ігоровича

Керівник економічного розділу
доцент каф. ПЕП та ПУ, к.е.н. **Л. В. Касьяненко**

ДОДАТОК В

ПЕРЕЛІК ДОКУМЕНТІВ НА ОПТИЧНОМУ НОСІЇ

Ім’я файлу	Опис
Пояснювальні документи	
yarilovets_122m_19_1_diploma.docx	Пояснювальна записка до кваліфікаційної роботи. Документ Word.
yarilovets_122m_19_1_diploma.pdf	Пояснювальна записка до кваліфікаційної роботи в форматі PDF
yarilovets_122m_19_1_vidguk.docx	Відгук керівника. Документ Word.
yarilovets_122m_19_1_retsenzia.docx	Рецензія. Документ Word.
yarilovets_122m_19_1_tezy.docx	Тези. Документ Word.
Презентація	
yarilovets_122m_19_1_presentation. pptx	Презентація кваліфікаційної роботи