Міністерство освіти і науки України
Національний технічний університет України
«Київський політехнічний інститут імені Ігоря Сікорського»
Фізико-технічний інститут

Розрахункова робота

Розв'язання ДРЧП за допомогою МСР із використанням неявної схеми

предмет «Методи обчислень»

Роботу виконав:
Студент 3 курсу ФТІ, групи ФІ-91
Цибульник Антон Владиславович

Приймала:
Стьопочкіна Ірина Валеріївна

2022

Зміст

Завдання та теоретичні основи

Завданням розрахункової роботи є чисельний розв'язок диференціального рівняння в часткових похідних (ДРЧП) одним з методів: або МСР (методом скінченних різниць), або МСЕ (методом скінчених елементів).

Суть методу сіток (МСР) полягає в апроксимації шуканої неперервної функції сукупністю наближених значень, розрахованих в скінченому наборі точок області – так званих «вузлах сітки». Сукупність вузлів і утворює сітку. Використання цього методу дозволяє звести диференціальну граничну задачу або до системи нелінійних рівнянь (неявна схема), або до ітераційної формули обчислень відносно невідомих вузлових значень функції (явна схема).

Методи скінчених елементів (МСЕ) і скінчених різниць (МСР) по суті є досить різними. Вони відрізняються за сутністю в тому, що в методі скінчених різниць апроксимуються *похідні шуканих функцій*, а в методі скінчених елементів – *сам розв'язок*.

Також методи відрізняються в конструкціях використовуваних сіток: у випадку методу сіток область протікання процесу апроксимується прямокутниками, а у випадку методу скінченних елементів – розбиття з урахуванням геометричних особливостей області.

Варіант завдання

Задається диференціальне рівняння в часткових похідних такого виду:

$$\frac{\partial u}{\partial t} = a\,\frac{\partial^2 u}{\partial x^2} + b\,\frac{\partial^2 u}{\partial y^2} - cu \tag{1}$$

При цьому початкові умови

$u\|_{t=0} = 0$	обрамлення області
$u\|_{t=0} = 10$	усюди всередині

В той час як граничні умови задані таким чином:

$$\frac{\partial u}{\partial x} = 0 \qquad \text{на правій та лівій грані області} \tag{2}$$

$$\frac{\partial u}{\partial y} = -ku \qquad \text{на верхній межі} \tag{3}$$

$$\frac{\partial u}{\partial y} = ku \qquad \text{на нижній межі} \tag{4}$$

Підготовка моделі до програмної обробки

Опис моделі та фізична інтерпретація

Вказане рівняння (1) є рівнянням параболічного типу, яке в загальному випадку виглядає так:

$$\frac{\partial u}{\partial t} = \sum_{i}^{n} k_i \frac{\partial^2 u}{\partial z_i^2} + \sum_{i}^{n} c_i \frac{\partial u}{\partial z_i} + d_i u + f(z_i, t),$$

де n – кількість ступенів свободи, а вказані параметри k_i, c_i, d_i, функції u та f мають такий зміст:

$u(z_i, t)$	стан процесу
k_i	коефіцієнт дифузії
c_i	коефіцієнт переносу
d_i	коефіцієнт «стоку»
$f(z_i, t)$	функція джерел

ДРЧП виду (1) є типовим прикладом протікання процесу теплопровідності. Крім того, оскільки у рівнянні задана залежність від часу та координат, то остаточно робимо висновок про еволюційну просторово-розподілену природу заданої моделі розподілу значень стану u (у нашому випадку – температури) із певним значенням в початковий момент часу $t = 0$.

Дискретизація моделі

Розв'язуватимемо завдання методом скінченних різниць (МСР). Розіб'ємо площину протікання процесу на $n \times n$ вузлів. Позначимо індексами i, j номери відповідного рядка та стовпця для конкретно взятого вузла, при цьому ці індекси пробігають значення $i = \overline{1, n},\ j = \overline{1, n}$.

Позначимо відстань між сусідніми просторовими вузлами Δx. Введемо індекс дискретного часу t, де $t = \overline{1, m}$. І наостанок позначимо відстань між сусідніми моментами часу Δt.

Скористаймося скінчено-різницевими апроксимаціями для частинних похідних u_t, u_{xx}, u_{yy}, які, власне, є своєрідними модифікаціями їхніх неперервних аналогів:

$$\frac{\partial u}{\partial t} = \frac{u_{ij}^{t+1} - u_{ij}^{t}}{\Delta x} \tag{5}$$

$$\frac{\partial^2 u}{\partial x^2} = \lambda \frac{u_{ij+1}^{t+1} - 2u_{ij}^{t+1} + u_{ij-1}^{t+1}}{\Delta x^2} + (1 - \lambda) \frac{u_{ij+1}^{t} - 2u_{ij}^{t} + u_{ij-1}^{t}}{\Delta x^2} \tag{6}$$

$$\frac{\partial^2 u}{\partial y^2} = \lambda \frac{u_{i+1j}^{t+1} - 2u_{ij}^{t+1} + u_{i-1j}^{t+1}}{\Delta y^2} + (1 - \lambda) \frac{u_{i+1j}^{t} - 2u_{ij}^{t} + u_{i-1j}^{t}}{\Delta y^2} \tag{7}$$

Підставимо вирази (5), (6) та (7) у рівняння (1). Перенесемо усі компоненти {t+1} у ліву частину рівності, а компоненти {t} – у праву. Отримаємо такий результат:

$$u_{ij}^{t+1}(\tfrac{1}{\Delta t}+\tfrac{2a\lambda}{\Delta x^2}+\tfrac{2b\lambda}{\Delta y^2}+c\lambda)-\tfrac{a\lambda}{\Delta x^2}u_{ij+1}^{t+1}-\tfrac{a\lambda}{\Delta x^2}u_{ij-1}^{t+1}-\tfrac{b\lambda}{\Delta y^2}u_{i+1j}^{t+1}-\tfrac{b\lambda}{\Delta y^2}u_{i-1j}^{t+1}=$$

$$=u_{ij}^{t}\tfrac{1}{\Delta t}+(1-\lambda)\left(a\,\tfrac{u_{ij+1}^{t}-2u_{ij}^{t}+u_{ij-1}^{t}}{\Delta x^2}+b\,\tfrac{u_{i+1j}^{t}-2u_{ij}^{t}+u_{i-1j}^{t}}{\Delta y^2}-cu_{ij}^{t}\right)$$

Назагал, чисельним розв'язком диференціального рівняння (1) будуть чергові значення функції стану u через певний проміжок часу. Тож у щойно отриманому дискретизованому рівнянні роль невідомих змінних відіграють саме компоненти {t+1} функції u, в той час як значення усіх параметрів та компоненти {t} функції стану u відомі.

Тож перепозначимо праву частину як одну складену константу B_{ij}^t, а компоненти при невідомих змінних позначатимемо таким чином:

$$a_1=\tfrac{1}{\Delta t}+\tfrac{2a\lambda}{\Delta x^2}+\tfrac{2b\lambda}{\Delta y^2}+c\lambda \tag{8}$$

$$a_2=a_3=-\tfrac{a\lambda}{\Delta x^2} \tag{9}$$

$$a_4=a_5=-\tfrac{b\lambda}{\Delta y^2} \tag{10}$$

Остаточно, з урахуванням виразів (8) - (10) дискретизована модель матиме вид:

$$a_1u_{ij}^{t+1}+a_2u_{ij+1}^{t+1}+a_3u_{ij-1}^{t+1}+a_4u_{i+1j}^{t+1}+a_5u_{i-1j}^{t+1}=B_{ij}^t,\quad i,j=\overline{2,n-1} \tag{11}$$

Формула (11) задає систему лінійних алгебраїчних рівнянь (СЛАР) для кожного наступного моменту часу t. Проте, варто зауважити, що вираз дає змогу обрахувати виключно внутрішні точки області, в той час як самі граничні значення обраховуватимуться дещо інакше.

Обробка граничних та початкових умов

Перш за все з'ясуємо, з якої матриці стану варто починати дослідження згідно варіанту. Задання початкових умов виразом $u|_{t=0}=0$ для граничної області та виразом $u|_{t=0}=10$ для усієї внутрішньої частини виокремлює матрицю станів в момент часу $t=0$ приблизно такого виду:

$$u^0=\begin{pmatrix}0&0&0&\dots&0&0&0\\0&10&10&\dots&10&10&0\\0&10&10&\dots&10&10&0\\\vdots&\vdots&\vdots&&\vdots&\vdots&\vdots\\0&10&10&\dots&10&10&0\\0&10&10&\dots&10&10&0\\0&0&0&\dots&0&0&0\end{pmatrix} \tag{12}$$

Наступним кроком дискретизуємо обмеження на границю області.

Аналогічним чином, користуючись скінчено-різницевими апроксимаціями для частинних похідних u_x та u_y, отримаємо такі дискретизовані аналоги граничних умов (2) - (4):

$$\frac{u_{i1}^{t+1} - u_{i2}^{t+1}}{\Delta x} = 0, \ \frac{u_{in}^{t+1} - u_{in-1}^{t+1}}{\Delta x} = 0 \qquad \text{ліві та праві обмеження}$$

$$\frac{u_{1j}^{t+1} - u_{2j}^{t+1}}{\Delta y} = -ku_{1j}^{t+1}, \ \frac{u_{nj}^{t+1} - u_{n-1j}^{t+1}}{\Delta y} = ku_{nj}^{t+1} \qquad \text{верхні та нижні обмеження}$$

Перенісши невідомі змінні, тобто компоненти {t+1}, та коефіцієнти при них у лівий бік, матимемо:

$$u_{i1}^{t+1} - u_{i2}^{t+1} = 0, \qquad u_{1j}^{t+1} - u_{2j}^{t+1}\left(\frac{1}{1+k\Delta y}\right) = 0, \qquad i,j = 1 \qquad (13)$$

$$u_{in}^{t+1} - u_{in-1}^{t+1} = 0, \qquad u_{nj}^{t+1} - u_{n-1j}^{t+1}\left(\frac{1}{1-k\Delta y}\right) = 0, \qquad i,j = n \qquad (14)$$

Висновок: дискретизовані записи системи внутрішніх елементів (11) та граничних елементів (13) - (14) формуватимуть повну систему лінійних алгебраїчних рівнянь (СЛАР). При цьому в програмній реалізації варто враховувати правильний порядок складення цих рівнянь. Озброївшись цими формулами, а також початковою матрицею стану (12), покроково розв'язуватимемо СЛАР у пошуках чергового значення матриці стану u у кожний наступний момент часу.

Опис програмної реалізації

Порядково зазначу функціонал програми, наведеної на Лістингу 1:

`рядки 1-3`	підключення необхідних бібліотек;
`рядки 7-13`	створення початкової матриці стану;
`рядки 5,15,17`	ініціалізація параметрів системи;
`рядки 19-21`	коефіцієнти СЛАР при невідомих змінних;
`рядок 27`	основний ітераційний цикл;
`рядки 39-58`	покроковий обхід граничних умов;
`рядки 60-69`	запис коефіцієнтів СЛАР для внутрішніх елементів;
`рядок 71`	розв'язання сформованої СЛАР;
`рядки 73-83`	перетворення стрічки розв'язку у матричний вигляд;
`рядки 88-109`	візуалізація отриманих результатів.

Розв'язання СЛАР (рядок 71) виконувалося за допомогою вбудованого методу `np.linalg.solve()` бібліотеки `numpy`. «Під капотом» цей метод використовує для вирішення системи так звану `LU`-декомпозицію, яка в свою чергу є однією з різновидів прямого методу Гауса.

Основна ідея розв'язку системи рівнянь виду $Ax = B$ методом LU-декомпозиції полягає у представленні квадратної матриці A у такому вигляді:

$$A = LU,$$

де L та U – нижня й верхня трикутна матриця такої ж розмірності. Після такої схеми переупорядкування сам розв'язок рівності $Ax = LUx = B$ шукатиметься у два кроки:

1. Знаходження розв'язку y з рівняння $Ly = B$;
2. Знаходження розв'язку x з рівняння $Ux = y$.

Власне, код програми обрахунку ДРЧП, тобто станів системи u у кожен момент часу, наведений нижче:

Лістинг 1: Код програми

```
import matplotlib.pyplot as plt
import numpy as np
import copy

n = 15 # one side dimension of nodes number

u = [] # initial matrix of states
for i in range(n):
    u.append([0.0 for m in range(n)])

for i in range(1,n-1):
    for j in range(1,n-1):
        u[i][j] = 10.0

dx,dy,dt,a,b,c,k,l = 1,1,1,2,1,-10,0.8,0.5 # all the necessary values

seconds = 20 # how long will your experiment take?

a1 = 1/dt + (2*l*a/dx**2) + (2*l*b/dy**2) + c*l
a2 = a3 = -(a*l)/dx**2
a4 = a5 = -(b*l)/dy**2

U = [] # matrix to save all the states (copy of an each value)
U.append(copy.deepcopy(u))

t = 0 # let the experiment begin!
while (t < seconds):
    u_a = []
    for i in range(n**2):
        u_a.append([0.0 for m in range(n**2)])

    u_b = []
    for i in range(n**2):
        u_b.append(0.0)

```

```
    row = 0
    for i in range(n):
        for j in range(n):
            if (j == 0):
                id = n*i + j
                u_a[row][id] = 1.0
                u_a[row][id+1] = -1.0
                row = row + 1
            elif (i == 0 and j != n-1):
                id = n*i + j
                u_a[row][id] = 1.0
                u_a[row][id+n] = -1/(1+k*dy)
                row = row + 1
            elif (j == n-1):
                id = n*i + j
                u_a[row][id] = 1.0
                u_a[row][id-1] = -1.0
                row = row + 1
            elif (i == n-1 and j != 0 and j != n-1):
                id = n*i + j
                u_a[row][id] = 1.0
                u_a[row][id-n] = -1/(1-k*dy)
                row = row + 1
            else:
                id1 = n*i + j
                id2 = n*i + (j+1)
                id3 = n*i + (j-1)
                id4 = n*(i+1) + j
                id5 = n*(i-1) + j

                u_a[row][id1], u_a[row][id2], u_a[row][id3], u_a[row][id4],
                    u_a[row][id5] = a1, a2, a3, a4, a5
                u_b[row] = u[i][j]/dt +
                    (1-l)*(a*(u[i][j+1]-2*u[i][j]+u[i][j-1])/dx**2 +
                    b*(u[i+1][j]-2*u[i][j]+u[i-1][j])/dy**2 - c*u[i][j])

                row = row + 1

    u_id = np.linalg.solve(u_a, u_b)

    # converting u_id to a matrix
    i,j = 0,0
    for id in range(n**2):
        if (id % n == 0 and id != 0):
            i = i + 1
            j = 0
            u[i][j] = u_id[id]
            j = j + 1
        else:
            u[i][j] = u_id[id]
            j = j + 1

    U.append(copy.deepcopy(u))
    t = t + 1
```

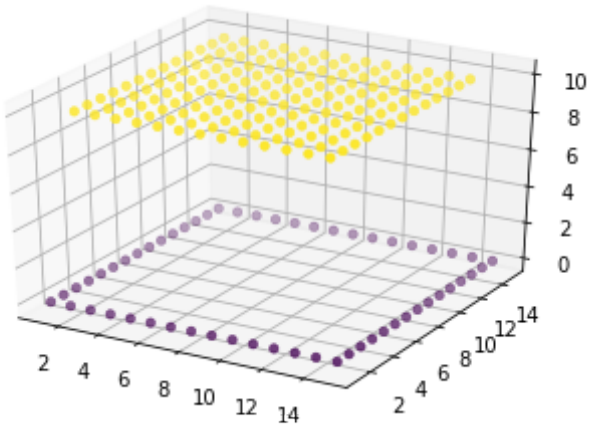
```
# drawing the results
x,y = [], []

number = 1
while (number < n+1):
    for i in range(n):
        x.append(number)
    number = number + 1

number = 1
while (number < n+1):
    for i in range(1,n+1):
        y.append(i)
    number = number + 1

for t in range(seconds):
    z = []
    for i in range(n):
        for j in range(n):
            z.append(U[t][i][j])
    ax = plt.axes(projection="3d")
    ax.scatter(x,y,z, c=z, cmap="viridis", linewidth=0.5)
    plt.show()
```

Візуалізація отриманих результатів

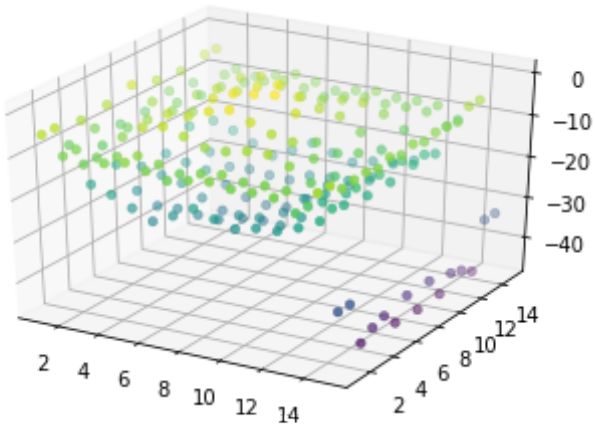
Першою буде наведена модель із 15×15 вузлів у різні проміжки часу. Величини параметрів кроків, відповідно

$$\Delta x = \Delta y = \Delta t = 1,$$

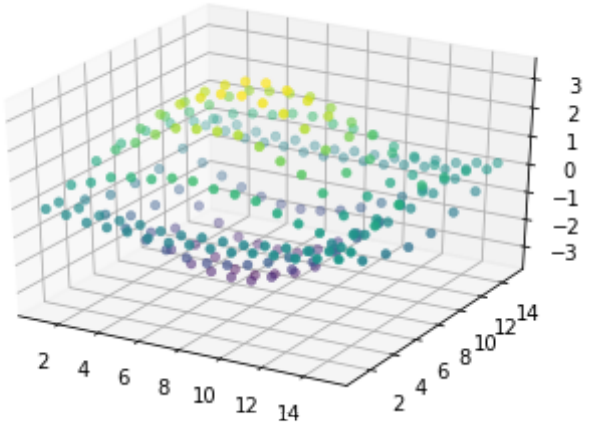
значення коефіцієнтів дифузії $a = 2,\ b = 1$, а величина «стоку» $c = -10$. Крім того, заданий параметр для граничних умов $k = 0.8$ та величина $\lambda = 0.5$:



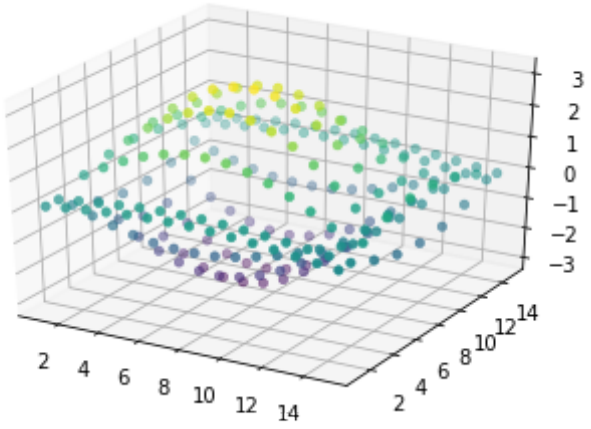
момент $t = 0$



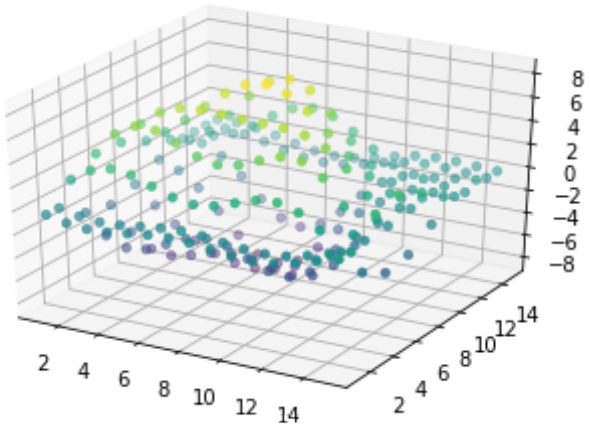
момент $t = 1$



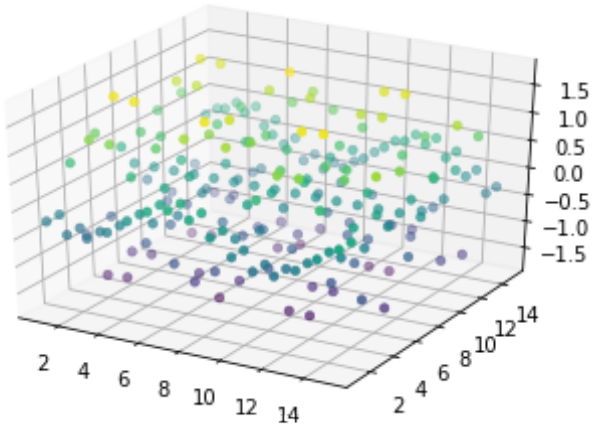
момент $t = 4$



момент $t = 10$



момент $t = 13$

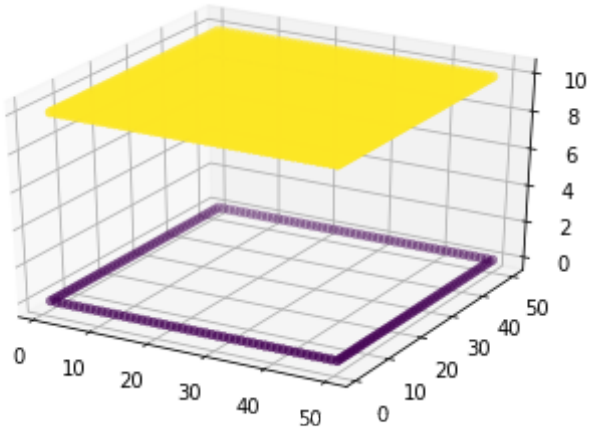


момент $t = 15$

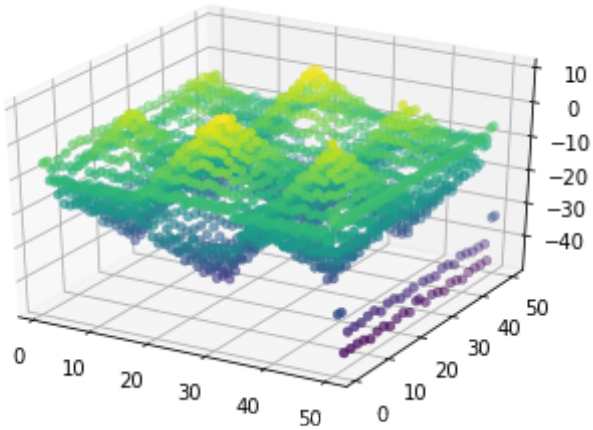
Наступною буде модель із 50×50 вузлів у різні проміжки часу. Аналогічно задані і параметри: величини кроків, відповідно

$$\Delta x = \Delta y = \Delta t = 1,$$

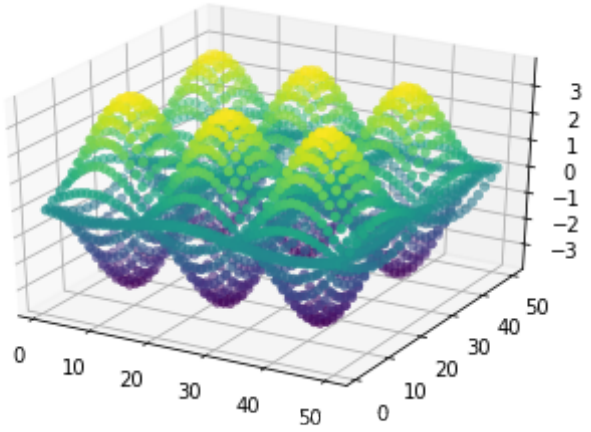
значення коефіцієнтів дифузії $a = 2,\ b = 1$, а величина «стоку» $c = -10$. Крім того, заданий параметр для граничних умов $k = 0.8$ та величина $\lambda = 0.5$:



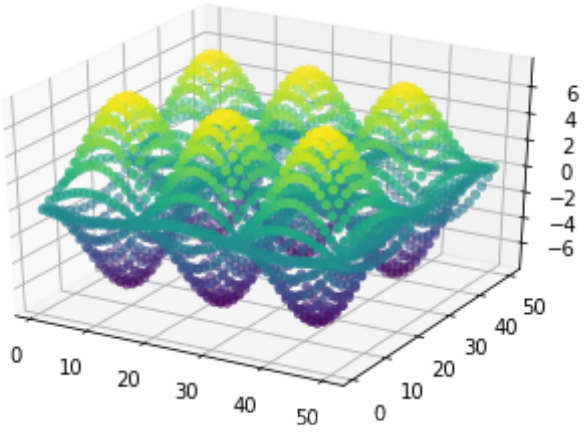
момент $t = 0$



момент $t = 1$



момент $t = 3$



момент $t = 12$

Висновки

У роботі було розглянуто розв'язання диференціального рівняння у часткових похідних (ДРЧП) методом скінченних різниць (МСР), також відомим як метод сіток. Спершу над заданою областю рівняння параболічного типу була проведена дискретизація на певну кількість вузлів. Опісля після обробки початкових та граничних умов було остаточно створене підґрунтя для формування системи алгебраїчних лінійних рівнянь (СЛАР), розв'язком якої є матриця станів у поточний момент часу.

Щодо програмного етапу, розв'зок СЛАР виконувався за допомогою вбудованого методу `np.linalg.solve()` бібліотеки `numpy`. «Під капотом» цей метод використовує так звану `LU`-декомпозицію, яка в свою чергу є однією з різновидів прямого методу Гауса. З обрахункової точки зору, при кількості вузлів 15×15 час однієї ітерація сягав 0.25 с, в той час як для 50×50 вузлів – 1.55 с.

У результатах роботи наведено приклади візуалізації нестійкої (перша модель) та стійкої (друга модель) схем. Проте, при інших значенях параметрів (наприклад, задовільнивши критерій Куранта значеннями $a = 1,\ b = 0.9$ замість $a = 2,\ b = 1$), можна досягнути стійкого аналогу для першої візуалізації.