Розрахунково-графічна робота
Методи обчислень

Виконав:
Студент 3 курсу НН ФТІ
групи ФІ-92
Поночевний Назар Юрійович
Варіант 12

Завдання:
Чисельно розв’язати рівняння:

$$\frac{\partial u}{\partial t} = a\frac{\partial^2 u}{\partial x^2} + a\frac{\partial^2 u}{\partial y^2} - f;$$
$$u(0) = u_0 \neq 0;$$
$$u(1, y, t) = u(2, y, t);$$
$$u(x,1,t) = k(u_{const} - u(x,1,t));$$

Функція f - довільна ненульова, не залежить від часу, k = 0,5; uconst = 200. Стан u повинен бути невід’ємним. Моделювання здійснювати на сітці 20x20 вузлів упродовж 10 часових кроків.

Бачимо, що це нестаціонарне рівняння дифузії (параболічне диференціальне рівняння у часткових похідних), де f - функція джерела, a - коефіцієнт дифузії та k - коефіцієнт проникності середовища. Воно може описувати, наприклад, поширення розчиненої речовини внаслідок дифузії або перерозподіл температури тіла в результаті теплопровідності.

1) Стан u можна обрахувати наближеною дискретною функцією застосовуючи метод скінченних різниць. Для розв'язку можна обрати метод Кранка-Ніколсон чи зворотний метод Ейлера. Метод Кранка-Ніколсон є безумовно чисельно стабільним для рівняння теплопровідності та дифузії. Проте, при використанні великого кроку сітки розв'язок може містити сильні коливання. У випадку, коли крок різницевої сітки змінити неможливо, при нестабільності розв'язку рекомендується використовувати менш точний, але також чисельно стабільний неявний зворотний метод Ейлера. Ми ж можемо змінити крок, тому виберемо метод Кранка-Ніколсон
2) Явно-неявна схема (Кранка-Ніколсон):

$$\frac{u_{i,j}^{k+1} - u_{i,j}^{k}}{\Delta t} = a\left(\lambda\left[\frac{u_{i+1,j}^{k+1} - 2u_{i,j}^{k+1} + u_{i-1,j}^{k+1}}{\Delta x^2} + \frac{u_{i,j+1}^{k+1} - 2u_{i,j}^{k+1} + u_{i,j-1}^{k+1}}{\Delta y^2}\right] + (1-\lambda)\left[\frac{u_{i+1,j}^{k} - 2u_{i,j}^{k} + u_{i-1,j}^{k}}{\Delta x^2} + \frac{u_{i,j+1}^{k} - 2u_{i,j}^{k} + u_{i,j-1}^{k}}{\Delta y^2}\right]\right) - f$$

$$u_{i,j}^{k+1} = u_{i,j}^{k} + a\Delta t\left(\lambda\left[\frac{u_{i+1,j}^{k+1} - 2u_{i,j}^{k+1} + u_{i-1,j}^{k+1}}{\Delta x^2} + \frac{u_{i,j+1}^{k+1} - 2u_{i,j}^{k+1} + u_{i,j-1}^{k+1}}{\Delta y^2}\right] + (1-\lambda)\left[\frac{u_{i+1,j}^{k} - 2u_{i,j}^{k} + u_{i-1,j}^{k}}{\Delta x^2} + \frac{u_{i,j+1}^{k} - 2u_{i,j}^{k} + u_{i,j-1}^{k}}{\Delta y^2}\right]\right) - f\Delta t$$

$$-a\lambda\Delta t\left[\frac{u_{i+1,j}^{k+1}}{\Delta x^2}+\frac{u_{i,j+1}^{k+1}}{\Delta y^2}\right]+\left[1+2a\lambda\Delta t\left(\frac{1}{\Delta x^2}+\frac{1}{\Delta y^2}\right)\right]u_{i,j}^{k+1}-a\lambda\Delta t\left[\frac{u_{i-1,j}^{k+1}}{\Delta x^2}+\frac{u_{i,j-1}^{k+1}}{\Delta y^2}\right]=$$

$$a\left(1-\lambda\right)\Delta t\left[\frac{u_{i+1,j}^{k}}{\Delta x^2}+\frac{u_{i,j+1}^{k}}{\Delta y^2}\right]+\left[1-2a(1-\lambda)\Delta t\left(\frac{1}{\Delta x^2}+\frac{1}{\Delta y^2}\right)\right]u_{i,j}^{k}+a(1-$$

$$\lambda)\Delta t\left[\frac{u_{i-1,j}^{k}}{\Delta x^2}+\frac{u_{i,j-1}^{k}}{\Delta y^2}\right]-f\Delta t$$

$$\mu=\frac{a\Delta t}{2}\left(\frac{1}{\Delta x^2}+\frac{1}{\Delta y^2}\right)$$

$$\Delta x=\Delta y\Rightarrow\mu=\frac{a\Delta t}{\Delta x^2}$$

$$\left(1+4\lambda\mu\right)u_{i,j}^{k+1}-\lambda\mu\left[u_{i+1,j}^{k+1}+u_{i,j+1}^{k+1}+u_{i-1,j}^{k+1}+u_{i,j-1}^{k+1}\right]=\left(1-4(1-\lambda)\mu\right)u_{i,j}^{k}+$$

$$(1-\lambda)\mu\left[u_{i+1,j}^{k}+u_{i,j+1}^{k}+u_{i-1,j}^{k}+u_{i,j-1}^{k}\right]-f\Delta t$$

Умова стійкості (умова Куранта):

$$\mu=\frac{a\Delta t}{2}\left(\frac{1}{\Delta x^2}+\frac{1}{\Delta y^2}\right)\leq\mu_{max}\Rightarrow$$

$$\Delta t\leq\frac{2\mu_{max}}{a}\frac{(\Delta x\Delta y)^2}{(\Delta x)^2+(\Delta y)^2}\Rightarrow\Delta t=\frac{1}{2a}\frac{(\Delta x\Delta y)^2}{(\Delta x)^2+(\Delta y)^2}$$

3) Визначимо константи:
   f = 10Δt
   a = 50
   λ = 0.75
   Δx = Δy = 0.05
   u(0) = u0 = 10
   u(x, 1, t) = 100/3
4) Реалізуємо програму

```
"""
Solving a Partial Differential Equation

"""

import numpy as np
from matplotlib import cm
import matplotlib.pyplot as plt
from mpl_toolkits.mplot3d import Axes3D


# ------------ Input ------------


a = 50.
w = h = 250.
dx = dy = 50.
lambda_value = 0.75


# ------------ Code ------------
```

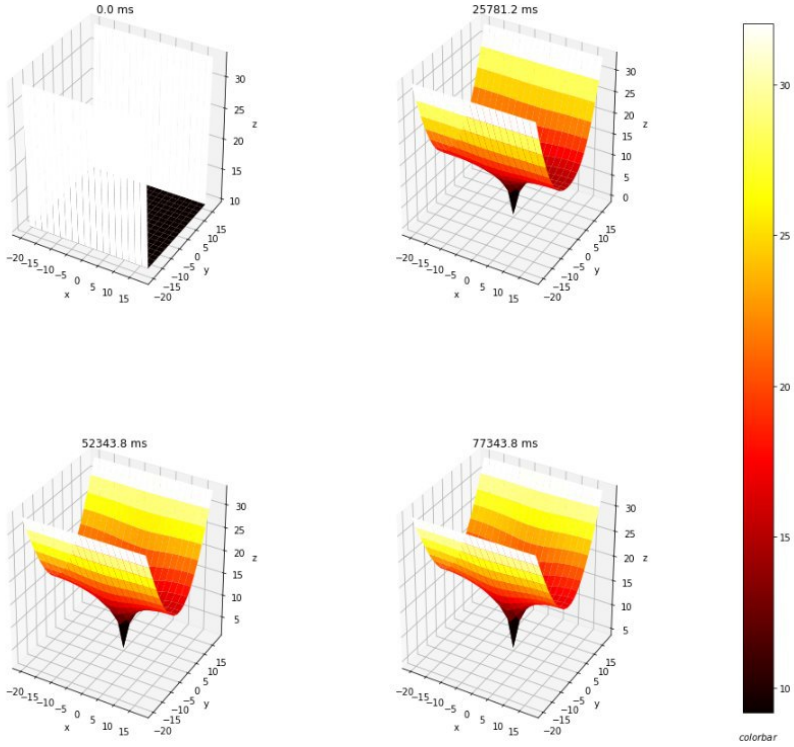
```
def create_matrix(w, h, nx, ny, mu, lambda_value, func):
    r = 0.001 * w
    r2 = r**2
    cx, cy = 0.5 * w, 0.5 * h

    C = np.identity(nx * ny)
    B = np.identity(nx * ny)
    F = np.zeros((nx, ny))
    for i in range(nx):
        for j in range(ny):
            p2 = (i * dx - cx)**2 + (j * dy - cy)**2
            if p2 < r2:
                F[i, j] = func

    F = F.reshape((nx * ny, 1))
    for i in range(nx, len(C) - nx):
        if i % nx != 0 and (i + 1) % nx != 0:
            for j in range(len(C[0])):
                if j == i - 1 or j == i + 1 or j == i - nx or j == i +
nx:
                    C[i, j] = -mu * lambda_value
                if j == i:
                    C[i, j] = 1 + 4 * mu * lambda_value

    for i in range(len(C)):
        if i % nx == 0:
            C[i, i + 1] = -1
        if (i + 1) % nx == 0:
            C[i, i-1] = -1

    for i in range(nx, len(B) - nx):
        if i % nx != 0 and (i + 1) % nx != 0:
            for j in range(len(B[0])):
                if j == i - 1 or j == i + 1 or j == i - nx or j == i +
nx:
                    B[i, j] = mu * (1 - lambda_value)
                if j == i:
                    B[i, j] = 1 - 4 * mu * (1 - lambda_value)

    for i in range(len(B)):
        if i % nx == 0:
            B[i, i + 1] = -1
        if (i + 1) % nx == 0:
            B[i, i - 1] = -1
```

```
    B = np.linalg.inv(B)
    A = np.dot(B, C)
    Func = np.dot(B, F)
    Func = Func.reshape((nx * nx,))

    return A, Func


def visualize(A, F, b, nsteps, nx, ny):
    x = np.arange(-nx, ny, 2)
    y = np.arange(-nx, ny, 2)
    x, y = np.meshgrid(x, y)

    fig = plt.figure(figsize=(15, 15))
    fignum, nfig = 0, [0, round(nsteps * 1/3), round(nsteps * 2/3),
nsteps - 1]
    for m in range(nsteps):
        if m in nfig:
            fignum += 1
            ax = fig.add_subplot(2, 2, fignum, projection='3d')
            surf = ax.plot_surface(x, y, b.reshape((nx, nx)), rstride=1,
cstride=1,
                                   cmap=cm.hot)
            ax.set_title('{:.1f} ms'.format(m * dt * 1000))
            ax.set_xlabel('x')
            ax.set_ylabel('y')
            ax.set_zlabel('z')
        b = np.linalg.solve(A, b - F)

    fig.subplots_adjust(right=0.8, wspace=0.4, hspace=0.4)
    cbar_ax = fig.add_axes([0.9, 0.15, 0.03, 0.7])
    cbar_ax.set_xlabel('$colorbar$', labelpad=20)
    fig.colorbar(surf, cax=cbar_ax)
    plt.show()


nx, ny = int(w / dx), int(h / dy)
dx2, dy2 = dx * dx, dy * dy
dt = dx2 * dy2 / (2 * a * (dx2 + dy2))
mu = a * dt / dx2
func = 1e+1 * dt * dt
print(f"Net size: {nx, ny}")
print(f"Time step: {dt}")
print(f"Mu: {mu}, Function: {func}")
```

```
A, F = create_matrix(w, h, nx, ny, mu, lambda_value, func)
b = 10 * np.ones(nx * nx)
for i in range(len(b)):
    if (i >= 0 and i <= nx - 1) or (i >= nx * (nx - 1) and i <= nx * nx
- 1):
        b[i] = 15
print(f"\nA's shape: {A.shape}")
print(f"F's shape: {F.shape}")
print(f"b's shape: {b.shape}")
visualize(A, F, b, 100, nx, ny)
```

5) Результат

```
Net size: (5, 5)
Time step: 12.5
Mu: 0.25, Function: 1562.5

A's shape: (25, 25)
F's shape: (25,)
b's shape: (25,)
```



6) Висновки

Під час числової реалізації обрахунку наближеного розв’язку диференційного рівняння у часткових похідних треба обачно обирати константи та декілька разів перевіряти тонкощі реалізації, щоб всі обрахунки виконувалися точно. Також, якщо користуватися сучасними засобами вирішення СЛАР, то можна суттєво пришвидшити збіжність і час виконання, бо сучасні бібліотеки Python

використовують паралельні обчислення і пришвидчувачі\оптимізатори методів. Загалом, у реальному житті треба користуватися готовими методами (наприклад, sympy.pde), які дозволяють ефективно проводити обчислення за допомогою безлічі сучасних способів. Проте, для подальшого покращення і модифікації методів, треба розуміти внутрішню структуру і принципи реалізації на алгоритмічному та апаратному рівнях.

7) Список використаних джерел:

   - http://csc.ucdavis.edu/~cmg/Group/readings/pythonissue_3of4.pdf
   - https://www.wikiwand.com/uk/%D0%A0%D1%96%D0%B2%D0%BD%D1%8F%D0%BD%D0%BD%D1%8F_%D0%B4%D0%B8%D1%84%D1%83%D0%B7%D1%96%D1%97
   - https://uk.wikipedia.org/wiki/%D0%9C%D0%B5%D1%82%D0%BE%D0%B4_%D0%9A%D1%80%D0%B0%D0%BD%D0%BA%D0%B0-%D0%9D%D1%96%D0%BA%D0%BE%D0%BB%D1%81%D0%BE%D0%BD
   - http://www.mitht.rssi.ru/it/pdf/cm/7_krank.pdf
   - https://en.wikipedia.org/wiki/Courant%E2%80%93Friedrichs%E2%80%93Lewy_condition
   - https://studme.org/231876/matematika_himiya_fizik/raznostnaya_shema_krankanikolsona
   - http://wiki.tomabel.org/images/c/c2/Paul_Summers_Final_Write_up.pdf
   - https://mybiblioteka.su/tom2/1-54383.html
   - https://en.wikipedia.org/wiki/Alternating_direction_implicit_method#Example:_2D_diffusion_equation