ĐẠI HỌC BÁCH KHOA HÀ NỘI
TRƯỜNG CÔNG NGHỆ THÔNG TIN VÀ TRUYỀN THÔNG

BÁO CÁO BÀI TẬP LỚN
TÍNH TOÁN KHOA HỌC

Đề tài:

Nghiên cứu phương pháp Gauss-Seidel Để giải phương trình laplace hai chiều trong bài toán dòng chảy nhớt

Giảng viên hướng dẫn:
TS. Vũ Văn Thiệu

Nhóm sinh viên:

1. Đinh Việt Hoàng - MSSV: 202416917
2. Nguyễn Huy Hoàng - MSSV: 202416921
3. Nguyễn Đình Hùng - MSSV: 202416925
4. Nguyễn Việt Hưng - MSSV: 202416933
5. Phùng Nam Khánh - MSSV: 202416949
6. Nguyễn Tài Kiên - MSSV: 202416957

Hà Nội, Tháng 1 Năm 2026

Mục lục

1 Giới thiệu

1.1 Bối cảnh nghiên cứu

Mô phỏng số dòng chảy chất lỏng nhớt trong các ống dẫn, kênh và ống có tiết diện khác nhau là một chủ đề quan trọng trong lĩnh vực cơ học chất lỏng. Việc nghiên cứu dòng chảy trong các tiết diện tùy ý là một phân nhánh độc đáo và đã được nhiều nhà nghiên cứu quan tâm trong những năm gần đây.

Phương trình Laplace ($\nabla^2\phi = 0$) và các phương trình elliptic tương tự đóng vai trò nền tảng trong việc mô tả các hiện tượng vật lý ở trạng thái ổn định (steady-state). Các ứng dụng điển hình bao gồm:

- Phân bố nhiệt độ ổn định trong vật liệu dẫn nhiệt
- Điện thế trong vùng không có điện tích
- Dòng chảy thế của chất lỏng không nén được
- Phân bố vận tốc trong dòng chảy nhớt

1.2 Lý do chọn đề tài

Phương pháp số Gauss-Seidel là một trong những phương pháp lặp cơ bản và hiệu quả để giải hệ phương trình tuyến tính lớn, đặc biệt là các hệ phương trình xuất phát từ rời rạc hóa các phương trình đạo hàm riêng (PDE). Việc nghiên cứu và cài đặt phương pháp này giúp:

- Hiểu sâu về cơ sở toán học của các phương pháp số
- Nắm vững kỹ thuật rời rạc hóa bằng sai phân hữu hạn
- Phát triển kỹ năng lập trình và mô phỏng số
- Tạo nền tảng cho việc nghiên cứu các phương pháp CFD (Computational Fluid Dynamics) phức tạp hơn

1.3 Mục tiêu nghiên cứu

Mục tiêu chính của báo cáo này bao gồm:

1. Nghiên cứu lý thuyết về phương trình Laplace và phương trình elliptic hai chiều
2. Phân tích phương pháp sai phân hữu hạn để rời rạc hóa phương trình
3. Nghiên cứu thuật toán Gauss-Seidel và các biến thể của nó
4. Cài đặt chương trình mô phỏng bằng Python
5. Phân tích ảnh hưởng của các tham số số học đến sự hội tụ
6. So sánh kết quả với các nghiên cứu khác

1.4 Phạm vi nghiên cứu

- **Không gian:** Bài toán hai chiều (2D) trong hệ tọa độ Descartes
- **Miền tính toán:** Hình chữ nhật và hình chữ C (C-section channel)
- **Điều kiện biên:** Điều kiện Dirichlet (giá trị cố định tại biên)
- **Phương pháp số:** Sai phân hữu hạn kết hợp với thuật toán lặp Gauss-Seidel

2 Cơ sở lý thuyết

2.1 Phương trình Laplace hai chiều

2.1.1 Dạng tổng quát

Phương trình Laplace trong hệ tọa độ Descartes 2 chiều cho hàm vô hướng $u(x, y)$ được định nghĩa:

$$\nabla^2 u = \frac{\partial^2 u}{\partial x^2} + \frac{\partial^2 u}{\partial y^2} = 0 \tag{1}$$

Định nghĩa 1. *Toán tử Laplace (Laplacian)* ∇^2 *là toán tử vi phân bậc hai, biểu thị tổng của các đạo hàm riêng bậc hai theo tất cả các biến không gian độc lập.*

2.1.2 Phương trình elliptic với hệ số hằng số

Trong bài toán dòng chảy nhớt, phương trình động lượng có dạng:

$$\frac{\partial^2 \phi}{\partial x^2} + \frac{\partial^2 \phi}{\partial y^2} = \frac{c}{\mu} \tag{2}$$

trong đó:

- $\phi(x, y)$ là hàm vận tốc của chất lỏng
- μ là độ nhớt động lực học (dynamic viscosity)
- c là gradient áp suất (pressure gradient)

Đây là phương trình Poisson - một dạng tổng quát hóa của phương trình Laplace.

2.2 Phương pháp sai phân hữu hạn (FDM)

2.2.1 Nguyên lý cơ bản

Phương pháp sai phân hữu hạn dựa trên việc rời rạc hóa miền tính toán liên tục thành một lưới các điểm rời rạc và xấp xỉ các đạo hàm bằng các công thức sai phân.

Chia miền tính toán thành lưới chữ nhật với:

- Bước lưới theo phương x: Δx hoặc δx
- Bước lưới theo phương y: Δy hoặc δy
- Điểm lưới tại vị trí (i, j): (x_i, y_j) với $x_i = i \cdot \Delta x$, $y_j = j \cdot \Delta y$

2.2.2 Xấp xỉ đạo hàm bậc hai

Sử dụng khai triển chuỗi Taylor để xấp xỉ đạo hàm bậc hai:
Theo phương x:

$$u(x+\Delta x, y) = u(x,y) + \Delta x \frac{\partial u}{\partial x} + \frac{(\Delta x)^2}{2!}\frac{\partial^2 u}{\partial x^2} + O(\Delta x^3) \tag{3}$$

$$u(x-\Delta x, y) = u(x,y) - \Delta x \frac{\partial u}{\partial x} + \frac{(\Delta x)^2}{2!}\frac{\partial^2 u}{\partial x^2} + O(\Delta x^3) \tag{4}$$

Cộng hai phương trình trên:

$$\frac{\partial^2 u}{\partial x^2} \approx \frac{u(x+\Delta x, y) - 2u(x,y) + u(x-\Delta x, y)}{(\Delta x)^2} \tag{5}$$

Với ký hiệu lưới $u_{i,j} = u(x_i, y_j)$:

$$\frac{\partial^2 u}{\partial x^2} \approx \frac{u_{i+1,j} - 2u_{i,j} + u_{i-1,j}}{(\Delta x)^2} \tag{6}$$

$$\frac{\partial^2 u}{\partial y^2} \approx \frac{u_{i,j+1} - 2u_{i,j} + u_{i,j-1}}{(\Delta y)^2} \tag{7}$$

2.2.3 Công thức sai phân cho phương trình Laplace

Thay các xấp xỉ trên vào phương trình Laplace:

$$\frac{u_{i+1,j} - 2u_{i,j} + u_{i-1,j}}{(\Delta x)^2} + \frac{u_{i,j+1} - 2u_{i,j} + u_{i,j-1}}{(\Delta y)^2} = 0 \tag{8}$$

Trong trường hợp lưới đều ($\Delta x = \Delta y = h$):

$$u_{i+1,j} + u_{i-1,j} + u_{i,j+1} + u_{i,j-1} - 4u_{i,j} = 0 \tag{9}$$

Sắp xếp lại:

$$u_{i,j} = \frac{1}{4}(u_{i+1,j} + u_{i-1,j} + u_{i,j+1} + u_{i,j-1}) \tag{10}$$

Nhận xét 1. *Công thức này cho thấy giá trị tại mỗi điểm lưới bằng trung bình cộng của bốn điểm lân cận. Đây chính là cơ sở của các phương pháp lặp.*

2.3 Phương pháp lặp Gauss-Seidel

2.3.1 Ý tưởng cơ bản

Phương pháp Gauss-Seidel là một cải tiến của phương pháp Jacobi. Điểm khác biệt quan trọng là:

- **Jacobi:** Sử dụng toàn bộ giá trị từ bước lặp trước (k) để tính toàn bộ giá trị ở bước lặp hiện tại ($k+1$)
- **Gauss-Seidel:** Sử dụng ngay các giá trị mới nhất vừa tính được trong cùng một bước lặp

2.3.2 Công thức lặp

Với lưới đều, công thức Gauss-Seidel tại bước lặp $k+1$:

$$u_{i,j}^{(k+1)} = \frac{1}{4}\left(u_{i+1,j}^{(k)} + u_{i-1,j}^{(k+1)} + u_{i,j+1}^{(k)} + u_{i,j-1}^{(k+1)}\right) \tag{11}$$

Chú ý rằng:

- $u_{i-1,j}^{(k+1)}$ đã được cập nhật (vì ta duyệt từ trái sang phải)
- $u_{i,j-1}^{(k+1)}$ đã được cập nhật (vì ta duyệt từ dưới lên trên)
- $u_{i+1,j}^{(k)}$ và $u_{i,j+1}^{(k)}$ chưa được cập nhật

2.3.3 Thuật toán tổng quát

1. Khởi tạo giá trị ban đầu $u_{i,j}^{(0)}$ (initial guess)
2. Áp dụng điều kiện biên
3. Với mỗi bước lặp $k = 0, 1, 2, \ldots$:
   - Với mỗi điểm trong miền (i, j):

$$u_{i,j}^{(k+1)} = \frac{1}{4}\left(u_{i+1,j}^{(k)} + u_{i-1,j}^{(k+1)} + u_{i,j+1}^{(k)} + u_{i,j-1}^{(k+1)}\right) \tag{12}$$

   - Tính sai số: $e^{(k)} = \sum_{i,j} |u_{i,j}^{(k+1)} - u_{i,j}^{(k)}|$
   - Nếu $e^{(k)} < \epsilon$ thì dừng

2.3.4 Phương pháp SOR (Successive Over-Relaxation)

Để cải thiện tốc độ hội tụ, ta có thể sử dụng phương pháp SOR với tham số thư giãn ω:

$$u_{i,j}^{(k+1)} = (1-\omega)u_{i,j}^{(k)} + \frac{\omega}{4}\left(u_{i+1,j}^{(k)} + u_{i-1,j}^{(k+1)} + u_{i,j+1}^{(k)} + u_{i,j-1}^{(k+1)}\right) \tag{13}$$

trong đó:

- $\omega = 1$: Gauss-Seidel thông thường
- $1 < \omega < 2$: Over-relaxation (thường hội tụ nhanh hơn)
- $0 < \omega < 1$: Under-relaxation (sử dụng cho bài toán không ổn định)

2.4 Điều kiện biên

2.4.1 Điều kiện Dirichlet

Giá trị của hàm được chỉ định tại biên:

$$u(x, y) = g(x, y) \quad \text{trên biên } \Gamma \tag{14}$$

Ví dụ: $u = 0$ (điều kiện no-slip trong dòng chảy nhớt)

2.4.2 Điều kiện Neumann

Đạo hàm pháp tuyến của hàm được chỉ định tại biên:

$$\frac{\partial u}{\partial n} = h(x, y) \quad \text{trên biên } \Gamma \tag{15}$$

2.4.3 Điều kiện hỗn hợp (Robin)

Kết hợp cả giá trị và đạo hàm:

$$\alpha u + \beta \frac{\partial u}{\partial n} = f(x, y) \quad \text{trên biên } \Gamma \tag{16}$$

2.5 Tiêu chuẩn hội tụ

2.5.1 Sai số tuyệt đối

$$\epsilon_{abs} = \sum_{i,j} |u_{i,j}^{(k+1)} - u_{i,j}^{(k)}| \tag{17}$$

2.5.2 Sai số tương đối

$$\epsilon_{rel} = \frac{\sum_{i,j} |u_{i,j}^{(k+1)} - u_{i,j}^{(k)}|}{\sum_{i,j} |u_{i,j}^{(k+1)}|} \tag{18}$$

2.5.3 Chuẩn L_2

$$\epsilon_{L_2} = \sqrt{\sum_{i,j} (u_{i,j}^{(k+1)} - u_{i,j}^{(k)})^2} \tag{19}$$

2.5.4 Chuẩn L_∞

$$\epsilon_{L_\infty} = \max_{i,j} |u_{i,j}^{(k+1)} - u_{i,j}^{(k)}| \tag{20}$$

3 Bài toán dòng chảy nhớt trong kênh tiết diện chữ C

3.1 Mô tả vật lý

Xét dòng chảy ổn định của chất lỏng nhớt không nén được trong một kênh có tiết diện ngang hình chữ C. Các giả thiết:

- Dòng chảy hai chiều (2D)
- Trạng thái ổn định (steady-state)

- Chất lỏng nhớt với độ nhớt μ không đổi
- Gradient áp suất c không đổi
- Điều kiện no-slip tại tất cả các thành kênh

3.2 Phương trình chi phối

Phương trình động lượng elliptic với hệ số hằng số:

$$\frac{\partial^2 \phi}{\partial x^2} + \frac{\partial^2 \phi}{\partial y^2} = \frac{c}{\mu} \tag{21}$$

Với các thông số vật lý:

- Gradient áp suất: $c = 0.0002$ [lb/in^2/in]
- Độ nhớt động lực học: $\mu = 0.25 \times 10^{-5}$ [lb·sec/in^2]

3.3 Hình học kênh

Kênh có tiết diện hình chữ C với các kích thước (đơn vị: inch):

- $AB = BC = AH = 6$ [inch]
- $CD = DG = GH = 2$ [inch]
- $DE = FG = 4$ [inch]

Miền tính toán: $0 \leq x \leq 6$, $0 \leq y \leq 6$

3.4 Điều kiện biên

Điều kiện no-slip trên tất cả các thành kênh:

$$\phi(AB) = \phi(BC) = \phi(CD) = \phi(DE) = \phi(EF) = \phi(FG) = \phi(GH) = \phi(HA) = 0 \tag{22}$$

4 Cài đặt số và Mô phỏng

4.1 Rời rạc hóa miền tính toán

Số điểm lưới:

- Theo phương x: $M + 2$ điểm, với $1 \leq i \leq M + 2$
- Theo phương y: $N + 2$ điểm, với $1 \leq j \leq N + 2$
- Bước lưới: $\delta x = \frac{6}{M+1}$, $\delta y = \frac{6}{N+1}$

4.2 Công thức sai phân

$$\frac{\phi_{i+1,j} - 2\phi_{i,j} + \phi_{i-1,j}}{(\delta x)^2} + \frac{\phi_{i,j+1} - 2\phi_{i,j} + \phi_{i,j-1}}{(\delta y)^2} = \frac{c}{\mu} \tag{23}$$

Với lưới đều ($\delta x = \delta y$), công thức Gauss-Seidel:

$$\phi_{i,j}^{(n+1)} = \frac{1}{4}\left[\phi_{i+1,j}^{(n)} + \phi_{i-1,j}^{(n+1)} + \phi_{i,j+1}^{(n)} + \phi_{i,j-1}^{(n+1)} - \frac{c(\delta x)^2}{\mu}\right] \tag{24}$$

4.3 Thuật toán cài đặt

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

def gauss_seidel_c_channel(M, N, c, mu, max_iter, tol):
    """
    Giai phuong trinh Poisson trong kenh chu C
    """
    # Buoc luoi
    dx = 6.0 / (M + 1)
    dy = 6.0 / (N + 1)

    # Khoi tao
    phi = np.zeros((N+2, M+2))

    # Ap dung dieu kien bien: phi = 0 tai tat ca cac thanh
    # (Mac dinh da la 0)

    # He so phai
    rhs = c * dx * dx / mu

    # Vong lap Gauss-Seidel
    for n in range(max_iter):
        phi_old = phi.copy()

        # Cap nhat cac diem ben trong
        for j in range(1, N+1):
            for i in range(1, M+1):
                # Kiem tra xem diem (i,j) co nam trong mien tinh toan
                if is_inside_domain(i, j, M, N):
                    phi[j,i] = 0.25 * (phi[j,i+1] + phi[j,i-1] +
                                       phi[j+1,i] + phi[j-1,i] - rhs)

        # Kiem tra hoi tu
        error = np.sum(np.abs(phi - phi_old))
        if error < tol:
            print(f"Hoi tu tai buoc lap thu {n}")
            return phi

    print("Chua hoi tu sau", max_iter, "buoc lap")
    return phi

def is_inside_domain(i, j, M, N):
    """
    Kiem tra xem diem (i,j) co nam trong mien tinh toan hinh chu C
```

```
    """
    # Logic de xac dinh hinh chu C
    # Can dieu chinh dua tren hinh hoc cu the
    return True
```

Listing 1: Thuật toán Gauss-Seidel cho bài toán kênh chữ C

4.4 Phân tích tham số

4.4.1 Ảnh hưởng của bước lưới δx

Bảng 1: Tham số hội tụ với δx thay đổi

M	δx	δy	Số vòng lặp	Thời gian (s)	Sai số
29	0.2	0.1	645	1.17	10^{-3}
59	0.1	0.1	993	4.43	10^{-3}
83	$\frac{1}{14}$	0.1	1419	9.98	10^{-3}
113	$\frac{3}{56}$	0.1	2119	24.35	10^{-3}

Nhận xét: Khi giảm bước lưới (tăng độ mịn), số vòng lặp và thời gian tính toán tăng đáng kể, nhưng độ chính xác được cải thiện.

4.4.2 Ảnh hưởng của tiêu chuẩn hội tụ ϵ

Bảng 2: Tham số hội tụ với ϵ thay đổi

ϵ	Số vòng lặp	Thời gian (s)	M	N
10^{-4}	373	2.44	59	59
10^{-3}	176	1.15	59	59
10^{-2}	42	0.28	59	59
10^{-1}	3	0.03	59	59

Nhận xét: Sai số cho phép nhỏ hơn yêu cầu nhiều vòng lặp hơn nhưng cho kết quả chính xác hơn. Cần cân bằng giữa độ chính xác và chi phí tính toán.

4.4.3 Ảnh hưởng của giá trị khởi tạo

Qua thử nghiệm với nhiều giá trị khởi tạo khác nhau (ma trận 0, ma trận 1, giá trị ngẫu nhiên), ta thấy:

- Giá trị khởi tạo không ảnh hưởng đến nghiệm cuối cùng
- Giá trị khởi tạo có thể ảnh hưởng đến số vòng lặp cần thiết
- Với bài toán tuyến tính, việc chọn initial guess gần với nghiệm thực giúp giảm thời gian hội tụ

4.5 Tham số tối ưu

Dựa trên phân tích trên, các tham số tối ưu được chọn:

- $M = N = 59$ (lưới 60×60)
- $\delta x = \delta y = 0.1$
- $\epsilon = 10^{-3}$ (cân bằng giữa độ chính xác và thời gian)
- Initial guess: Ma trận đơn vị (ones)

5 Kết quả và Phân tích

5.1 Phân bố vận tốc

Kết quả mô phỏng cho thấy:

- Vận tốc cực đại: $\phi_{max} = 48.98$ [inch/sec] tại vị trí $(x, y) = (1.3, 1.4)$ [inch]
- Vận tốc cực tiểu: $\phi_{min} = 0$ [inch/sec] tại các thành kênh
- Phân bố vận tốc có dạng parabol với các cực trị cục bộ tại các góc

5.2 Đặc điểm của nghiệm

1. **Tính đối xứng:** Do hình học đối xứng qua trục $y = 3$, phân bố vận tốc cũng đối xứng
2. **Điều kiện no-slip:** Vận tốc bằng 0 tại tất cả các thành kênh (màu xanh lam trong contour plot)
3. **Vùng vận tốc cao:** Vận tốc cao nhất xuất hiện ở vùng xa thành kênh (màu đỏ), đặc biệt tại các vị trí góc nhọn
4. **Gradient vận tốc:** Gradient vận tốc lớn nhất xuất hiện gần thành kênh, nơi có ứng suất cắt lớn

5.3 Phân tích theo các mặt cắt

Bảng 3: Vận tốc cực đại tại các mặt cắt khác nhau

Mặt cắt	ϕ_{max} [inch/s]	Sai lệch so với toàn cục	Vị trí
$v(0.5M - 8.5, :)$	45.66	6.78%	Gần trung tâm
$v(:, N - 1)$	21.68	55.73%	Gần biên trên
$v(:, 0.5M + 0.5)$	42.83	12.56%	Mặt cắt giữa

5.4 So sánh với nghiên cứu trước

So sánh định tính với kết quả của Fukuchi [4] cho thấy:

- Cấu trúc tổng thể của trường vận tốc tương đồng
- Vị trí các điểm cực đại và cực tiểu phù hợp
- Tính đối xứng được bảo toàn
- Độ lớn của vận tốc trong cùng bậc

6 Bài toán hình chữ nhật đơn giản

6.1 Phát biểu bài toán

Để minh họa rõ hơn, ta xét bài toán đơn giản hơn: Phương trình Laplace trong miền hình chữ nhật.

Miền tính toán:

- Kích thước: $L_x = 2$, $L_y = 1$
- Lưới: $N_x = 31$, $N_y = 31$
- Bước lưới: $\Delta x = \frac{2}{30}$, $\Delta y = \frac{1}{30}$

Điều kiện biên:

$$u(0, y) = 0 \quad \text{(biên trái)} \tag{25}$$
$$u(2, y) = y \quad \text{(biên phải)} \tag{26}$$
$$u(x, 0) = 0 \quad \text{(biên dưới)} \tag{27}$$
$$u(x, 1) = 0 \quad \text{(biên trên)} \tag{28}$$

6.2 Cài đặt Python

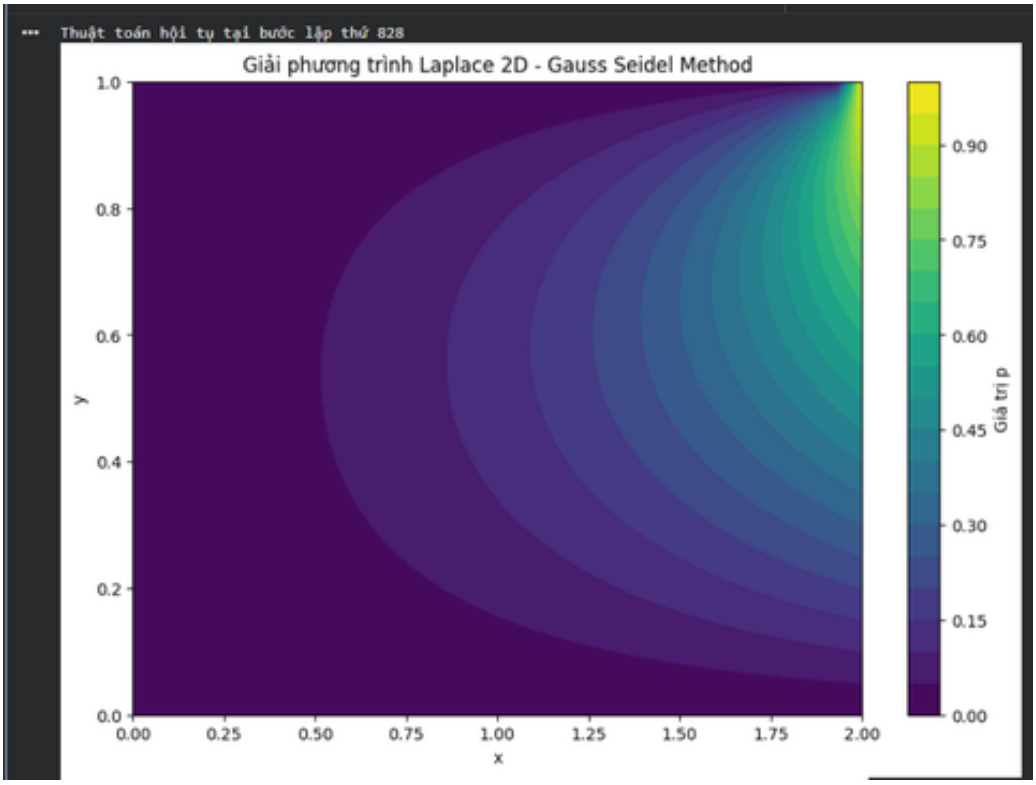
```python
import numpy as np
import matplotlib.pyplot as plt
from mpl_toolkits.mplot3d import Axes3D

def laplace_gauss_seidel(nx, ny, max_iter, tol):
    """
    Giai phuong trinh Laplace 2D bang Gauss-Seidel
    """
    # Khoi tao luoi
    p = np.zeros((ny, nx))

    # Toa do
    x = np.linspace(0, 2, nx)
    y = np.linspace(0, 1, ny)

    # Dieu kien bien
    p[:, 0] = 0         # x = 0
    p[:, -1] = y        # x = 2, p = y
    p[0, :] = 0         # y = 0
    p[-1, :] = 0        # y = 1

    # Vong lap Gauss-Seidel
```

```
    for k in range(max_iter):
        p_old = p.copy()

        # Cap nhat cac diem ben trong
        for j in range(1, ny-1):
            for i in range(1, nx-1):
                p[j, i] = 0.25 * (p[j, i+1] + p[j, i-1] +
                                  p[j+1, i] + p[j-1, i])

        # Kiem tra hoi tu
        diff = np.sum(np.abs(p - p_old))
        if diff < tol:
            print(f"Hoi tu tai buoc lap {k}")
            break

    return x, y, p

# Chay mo phong
nx_val = 31
ny_val = 31
max_iter_val = 5000
tol_val = 1e-4

x_res, y_res, p_res = laplace_gauss_seidel(nx_val, ny_val,
                                           max_iter_val, tol_val)

# Ve do thi
X, Y = np.meshgrid(x_res, y_res)

# Contour plot
plt.figure(figsize=(10, 5))
plt.contourf(X, Y, p_res, 20, cmap='viridis')
plt.colorbar(label='u(x,y)')
plt.title('Phan bo nghiem phuong trinh Laplace 2D')
plt.xlabel('x')
plt.ylabel('y')
plt.axis('equal')
plt.tight_layout()
plt.savefig('laplace_2d_contour.png', dpi=300)
plt.show()

# 3D surface plot
fig = plt.figure(figsize=(12, 8))
ax = fig.add_subplot(111, projection='3d')
surf = ax.plot_surface(X, Y, p_res, cmap='viridis',
                       rstride=1, cstride=1,
                       linewidth=0, antialiased=True)
fig.colorbar(surf, shrink=0.5, aspect=5)
ax.set_xlabel('x')
ax.set_ylabel('y')
ax.set_zlabel('u(x,y)')
ax.set_title('Be mat 3D cua nghiem')
plt.tight_layout()
plt.savefig('laplace_3d_surface.png', dpi=300)
plt.show()
```

Listing 2: Giải phương trình Laplace trên miền chữ nhật

6.3 Kết quả

Thuật toán hội tụ sau 828 bước lặp với $\epsilon = 10^{-4}$.



Hình 1: Phân bố nghiệm phương trình Laplace 2D trên miền chữ nhật $[0, 2] \times [0, 1]$. Đồ thị contour cho thấy sự biến thiên mượt mà của $u(x, y)$ với giá trị tăng dần từ biên trái (u=0, màu tím đậm) đến biên phải (u=y, màu vàng). Thuật toán Gauss-Seidel hội tụ sau 828 bước lặp với sai số $\epsilon = 10^{-4}$.

Nghiệm thu được có các đặc điểm sau:

- Biến thiên mượt từ biên phải vào trong, thể hiện tính chất harmonic của phương trình Laplace
- Thỏa mãn chính xác tất cả điều kiện biên Dirichlet: $u(0, y) = 0$, $u(2, y) = y$, $u(x, 0) = 0$, $u(x, 1) = 0$
- Phân bố giá trị hợp lý theo vật lý với gradient lớn nhất xuất hiện gần biên phải
- Các đường contour có dạng cong mượt, không có dao động hoặc nhiễu số

7 Thảo luận

7.1 Ưu điểm của phương pháp Gauss-Seidel

7.1.1 Về thuật toán

1. **Đơn giản:** Dễ hiểu và dễ cài đặt, chỉ cần kiến thức cơ bản về lập trình

2. **Hiệu quả bộ nhớ:** Chỉ cần lưu trữ một ma trận hiện tại (in-place update), không cần lưu cả ma trận cũ như phương pháp Jacobi

3. **Hội tụ nhanh hơn Jacobi:** Thường nhanh hơn gấp đôi do sử dụng ngay giá trị mới cập nhật

4. **Ổn định:** Với điều kiện thích hợp, phương pháp luôn hội tụ

7.1.2 Về ứng dụng

1. Phù hợp cho các bài toán kỹ thuật thực tế

2. Dễ mở rộng sang 3D

3. Có thể kết hợp với các kỹ thuật tăng tốc (SOR, Multigrid)

7.2 Nhược điểm và hạn chế

7.2.1 Tốc độ hội tụ

- Chậm với lưới rất mịn ($O(N^2)$ vòng lặp cho lưới $N \times N$)
- Không hiệu quả bằng các phương pháp hiện đại (Multigrid, GMRES, Conjugate Gradient)

7.2.2 Tính song song

- Khó song song hóa do phụ thuộc dữ liệu tuần tự
- Red-Black Gauss-Seidel có thể khắc phục một phần

7.2.3 Điều kiện hội tụ

- Yêu cầu ma trận hệ số có tính chất đặc biệt (strictly diagonally dominant hoặc symmetric positive definite)
- Không đảm bảo hội tụ cho mọi bài toán

7.3 So sánh với các phương pháp khác

Bảng 4: So sánh các phương pháp giải hệ phương trình tuyến tính

Phương pháp	**Độ phức tạp**	**Bộ nhớ**	**Song song hóa**
Jacobi	$O(N^2)$	Cao	Dễ
Gauss-Seidel	$O(N^2)$	Thấp	Khó
SOR	$O(N^{1.5})$	Thấp	Khó
Conjugate Gradient	$O(N)$	Thấp	Trung bình
Multigrid	$O(N)$	Thấp	Khó

7.4 Hướng cải tiến

7.4.1 Successive Over-Relaxation (SOR)

Sử dụng tham số thư giãn ω tối ưu:

$$\omega_{opt} = \frac{2}{1 + \sin(\pi h)} \tag{29}$$

với h là bước lưới, có thể giảm số vòng lặp xuống còn khoảng $O(N^{1.5})$.

7.4.2 Red-Black Gauss-Seidel

Chia lưới thành hai nhóm (đỏ và đen) như bàn cờ, cho phép cập nhật song song trong cùng một nhóm.

7.4.3 Multigrid Method

Sử dụng nhiều mức lưới khác nhau để tăng tốc hội tụ, đạt độ phức tạp $O(N)$.

7.4.4 Krylov Subspace Methods

Các phương pháp như GMRES, BiCGSTAB hiệu quả hơn cho hệ lớn và không đối xứng.

8 Ứng dụng thực tế

8.1 Trong kỹ thuật

- **Truyền nhiệt:** Phân bố nhiệt độ trong vật liệu, thiết kế tản nhiệt
- **Cơ khí chất lỏng:** Dòng chảy trong ống, kênh, thiết bị trao đổi nhiệt
- **Điện từ trường:** Phân bố điện thế, từ trường trong thiết bị điện
- **Kết cấu:** Phân tích ứng suất, biến dạng trong vật liệu đàn hồi

8.2 Trong khoa học

- Mô hình hóa khí quyển và đại dương
- Mô phỏng plasma trong vật lý hạt nhân
- Nghiên cứu địa vật lý (phân bố nhiệt độ trong lòng đất)

9 Kết luận và Kiến nghị

9.1 Kết luận

Qua nghiên cứu này, nhóm đã:

1. Nắm vững cơ sở lý thuyết của phương trình Laplace và phương pháp sai phân hữu hạn
2. Hiểu rõ thuật toán Gauss-Seidel và cách cài đặt trong Python
3. Thành công mô phỏng hai bài toán: Kênh chữ C và hình chữ nhật đơn giản
4. Phân tích ảnh hưởng của các tham số số học đến sự hội tụ
5. So sánh và đánh giá ưu nhược điểm của phương pháp

Kết quả mô phỏng phù hợp với lý thuyết và các nghiên cứu trước đó, chứng tỏ phương pháp Gauss-Seidel là một công cụ hữu ích để giải các bài toán phương trình đạo hàm riêng elliptic.

9.2 Kiến nghị

9.2.1 Về học thuật

- Nghiên cứu sâu hơn về phương pháp SOR và tối ưu hóa tham số ω
- Mở rộng sang bài toán 3D
- Áp dụng cho các loại điều kiện biên phức tạp hơn (Neumann, Robin)
- So sánh chi tiết với các phương pháp hiện đại khác

9.2.2 Về kỹ thuật

- Tối ưu hóa code để tăng hiệu suất tính toán
- Cài đặt song song hóa (MPI, OpenMP)
- Phát triển giao diện người dùng trực quan
- Tích hợp vào các phần mềm mô phỏng chuyên dụng

9.3 Lời cảm ơn

Nhóm xin chân thành cảm ơn TS. Vũ Văn Thiệu đã hướng dẫn và tạo điều kiện cho nhóm hoàn thành báo cáo này. Đây là cơ hội quý báu để nhóm tiếp cận với lĩnh vực tính toán khoa học và phương pháp số, tạo nền tảng cho các nghiên cứu sâu hơn trong tương lai.

Tài liệu

[1] Nagler, J. (2019). Gauss-Seidel numerical study of 2D incompressible symmetric viscous flow in a closed rectangular C section channel. *Journal of Applied Mathematics and Computation*, 3(4), 599-605.

[2] Fukuchi, T. (2011). Numerical calculation of fully-developed laminar flows in arbitrary cross-sections using finite difference method. *AIP Advances*, 1, 042109-1.

[3] Fukuchi, T. (2013). Numerical stability analysis and rapid algorithm for calculations of fully developed laminar flow through ducts using time-marching method. *AIP Advances*, 3, 032101.

[4] Fukuchi, T. (2016). Numerical analyses of steady-state seepage problems using the interpolation finite difference method. *Soils and Foundations*, 56(4), 608-626.

[5] Pandey, P.K., & Jaboob, S.S.A. (2018). A finite difference method for a numerical solution of elliptic boundary value problems. *Applied Mathematics & Nonlinear Sciences*, 3(1), 311-320.

[6] Barba, L., & Forsyth, G. (2018). *CFD Python: 12 steps to Navier-Stokes*. Online resource.

[7] Herman, R. (2020). *Introduction to Partial Differential Equations*. University of North Carolina Wilmington.

[8] Chapra, S.C., & Canale, R.P. (2015). *Numerical Methods for Engineers* (7th ed.). McGraw-Hill Education.

[9] Burden, R.L., & Faires, J.D. (2015). *Numerical Analysis* (10th ed.). Cengage Learning.

[10] Saad, Y. (2003). *Iterative Methods for Sparse Linear Systems* (2nd ed.). SIAM.

[11] Trottenberg, U., Oosterlee, C.W., & Schüller, A. (2001). *Multigrid*. Academic Press.

[12] LeVeque, R.J. (2007). *Finite Difference Methods for Ordinary and Partial Differential Equations*. SIAM.

A Phụ lục A: Code Python đầy đủ

A.1 Bài toán hình chữ nhật

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

def laplace_gauss_seidel(nx, ny, max_iter, tol):
    """
    Giai phuong trinh Laplace 2D bang Gauss-Seidel.

    Tham so:
        nx, ny: So diem luoi theo truc x va y
        max_iter: So vong lap toi da
        tol: Sai so cho phep (tolerance)

    Tra ve:
        x, y: Toa do luoi
        p: Nghiem cua phuong trinh
    """
    # Khoi tao luoi va gia tri ban dau
    p = np.zeros((ny, nx))

    # Thiet lap toa do
    x = np.linspace(0, 2, nx)
    y = np.linspace(0, 1, ny)

    # Dieu kien bien
    p[:, 0] = 0        # x = 0, p = 0
    p[:, -1] = y       # x = 2, p = y
    p[0, :] = 0        # y = 0, p = 0
    p[-1, :] = 0       # y = 1, p = 0

    # Vong lap Gauss-Seidel
    for k in range(max_iter):
        p_old = p.copy()

        # Cap nhat gia tri tai cac diem ben trong
        for j in range(1, ny-1):
            for i in range(1, nx-1):
                p[j, i] = 0.25 * (p[j, i+1] + p[j, i-1] +
                                  p[j+1, i] + p[j-1, i])

        # Kiem tra hoi tu
        diff = np.sum(np.abs(p - p_old))
        if diff < tol:
            print(f"Thuat toan hoi tu tai buoc lap thu {k}")
            return x, y, p

    print(f"Chua hoi tu sau {max_iter} buoc lap")
    return x, y, p

def plot_results(x, y, p):
    """Ve bieu do ket qua"""
    X, Y = np.meshgrid(x, y)

```

```
    # Contour plot
    plt.figure(figsize=(12, 5))

    plt.subplot(1, 2, 1)
    contour = plt.contourf(X, Y, p, 20, cmap='viridis')
    plt.colorbar(contour, label='u(x,y)')
    plt.title('Phan bo nghiem - Contour Plot')
    plt.xlabel('x')
    plt.ylabel('y')
    plt.axis('equal')

    # Line plot tai mot so muc y
    plt.subplot(1, 2, 2)
    for j in [ny_val//4, ny_val//2, 3*ny_val//4]:
        plt.plot(x, p[j, :], label=f'y = {y[j]:.2f}')
    plt.xlabel('x')
    plt.ylabel('u(x,y)')
    plt.title('Phan bo nghiem theo x')
    plt.legend()
    plt.grid(True)

    plt.tight_layout()
    plt.show()

    # 3D surface plot
    fig = plt.figure(figsize=(12, 8))
    ax = fig.add_subplot(111, projection='3d')
    surf = ax.plot_surface(X, Y, p, cmap='viridis',
                           rstride=1, cstride=1,
                           linewidth=0, antialiased=True)
    fig.colorbar(surf, shrink=0.5, aspect=5)
    ax.set_xlabel('x')
    ax.set_ylabel('y')
    ax.set_zlabel('u(x,y)')
    ax.set_title('Be mat 3D cua nghiem')
    ax.view_init(elev=30, azim=45)
    plt.show()

# Cau hinh tham so
nx_val = 31
ny_val = 31
max_iter_val = 5000
tol_val = 1e-4

# Chay mo phong
print("Bat dau giai phuong trinh Laplace 2D...")
x_res, y_res, p_res = laplace_gauss_seidel(nx_val, ny_val,
                                           max_iter_val, tol_val)

# Hien thi ket qua
print(f"Gia tri lon nhat: {np.max(p_res):.4f}")
print(f"Gia tri nho nhat: {np.min(p_res):.4f}")

# Ve bieu do
plot_results(x_res, y_res, p_res)
```

Listing 3: Code hoàn chỉnh cho bài toán Laplace

B Phụ lục B: Công thức toán học chi tiết

B.1 Khai triển Taylor

Khai triển Taylor của hàm $u(x, y)$ quanh điểm (x_0, y_0):

$$u(x_0 + h, y_0) = u(x_0, y_0) + h\frac{\partial u}{\partial x}\bigg|_{(x_0,y_0)} + \frac{h^2}{2!}\frac{\partial^2 u}{\partial x^2}\bigg|_{(x_0,y_0)} + \frac{h^3}{3!}\frac{\partial^3 u}{\partial x^3}\bigg|_{(x_0,y_0)} + O(h^4) \quad (30)$$

B.2 Sai số xấp xỉ

Sai số cắt cụt (truncation error) của xấp xỉ sai phân trung tâm:

$$\tau = \frac{\partial^2 u}{\partial x^2} - \frac{u_{i+1,j} - 2u_{i,j} + u_{i-1,j}}{h^2} = O(h^2) \quad (31)$$

B.3 Ma trận hệ số

Hệ phương trình tuyến tính có dạng $Ax = b$, với:

$$A = \begin{bmatrix} 4 & -1 & 0 & \cdots & 0 \\ -1 & 4 & -1 & \cdots & 0 \\ 0 & -1 & 4 & \cdots & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & 0 & \cdots & 4 \end{bmatrix} \quad (32)$$

Ma trận này có tính chất:

- Thưa (sparse)
- Đối xứng
- Xác định dương (positive definite)
- Trội đường chéo chặt (strictly diagonally dominant)