**I. Giới thiệu bài toán**

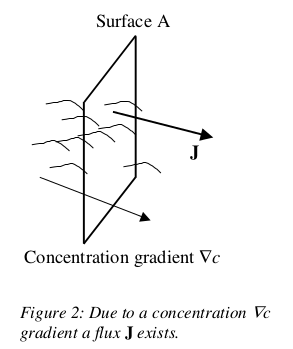
Báo cáo này xoay quanh việc giải bài toán khuếch tán. Khảo sát quá trình khếch tán của một khối chất rắn hoặc chất tan trong một dung môi. Quá trình khuếch tán có thể được mô tả bởi một phương trình vi phân bậc 2, tuyến tính từng phần.

Xét một hợp chất hóa học đặt rrong một dung môi. Hợp chất này có nồng độ c ( đơn vị là mol/m3). Chúng ta giả sử rằng hợp chất này chỉ chuyển động bởi sự khuếch tán tự do trong dung môi. Chúng ta sẽ lấy đạo hàm theo nồng độ c, bằng cách xem xét một khối cân bằng trong một thể tích nhỏ và áp dụng định luật Fick để liên hệ giữa dòng chảy khếch tán và độ biến thiên nồng độ.

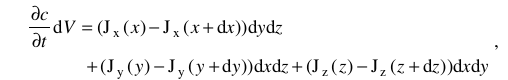
Xem xét tình huống như trong hình 2, nơi mà độ biến thiên nồng độ ∇c tồn tại, và do đó tồn tại dòng chất tan. Một dòng là một lượng chất vượt qua một đơn vị diện tích trong đơn vị thời gian. Dòng chảy này, J được tính bằng (mol/m2s). Theo định luật Fick, ta có:

J = − D∇c (1)

trong đó D là hệ số khuếch tán (m2/s).



Tiếp đến, chúng ta xem xét một khối đủ nhỏ dV = dxdydz( Hình 3) và tính toán lượng chất tan khuếch tán vào và ra khỏi khối hộp đó. Chúng ta xem xét theo 3 hướng trục tọa độ đề các và tổng hợp chúng để thu được giá trị dòng tổng hợp. Ví dụ theo hướng x, chúng ta có thể viết (J x ( x) − J x ( x + dx))dydz. Từ đó, ta có:



trong đó t biểu thị thời gian. Chia cả 2 vế biểu thức cho dV và tìm giới hạn khi dx, dy, dz → 0 chúng ta thu được:



Cuối cùng kết hợp với kết quả định luật Fick ta được:



Trong báo cáo này, để đơn giản, chúng ta xét một trường hợp cụ thể của khuyếch tán. Đó là khuếch tán theo một phương.

Xét một miền hình vuông, và không mất tính tổng quát, chúng ta giả sử 0 ≤ x ≤ 1 và 0 ≤ y ≤ 1. Thêm vào đó, chúng ta luôn giả sử điều kiện biên sau: c(x, = 1;t) = H(t) và c(x, y = 0; t) = 0. Vì vậy trên cùng của miền khảo sát, nồng độ luôn bằng 1 và ở dưới cùng của miền khảo sát, nồng độ chất tan luôn bằng 0. Thêm vào đó, theo phương x, chúng ta cũng sẽ luôn giả sử thoả mãn điều kiện biên tuần hoàn: c(x = 0, y; t) = c(x = 1, y; t) hoặc tổng quát hơn, c( x = ξ , y; t ) = c( x = ξ + 1, y; t ). Điều kiện biên này được thể hiện trong hình 7. Chú ý rằng bởi vì điều kiện biên tuần hoàn, quá trình khuếch tán trở thành khếch tán theo phương và nghiệm của phương trình khuếch tán là công thức 7. Lời giải này rất hữu ích bởi nó cho chúng ta phương pháp đánh độ chính xác của phương pháp số của phương trình khuếch tán 2 chiều. Cuối cùng, để đưa ra một lý do tại sao chúng ta cần phương pháp số trong giải quyết bài toán khuếch tán, chúng ta xem qua hình 8, trong đó một đối tượng hình vuông được đặt trong vùng. Đối tượng giả sử là một vật hấp phụ, ví dụ điều kiện trạng thái đối tượng đơn giản là con-object = 0. Trong tình huống này, chúng ta không thể đưa ra được lời giải lý thuyết, tuy nhiên phương trình khuyếch tán có thể được giải bằng phương pháp số mà chúng ta sẽ trình bày trong mục tiếp theo.

Trong nhiều trường hợp ta chỉ quan tâm tới trạng thái có nồng độ ổn định cuối cùng, mà không quan tâm tới quá trình để đạt được trạng thái đó. Điều này là có thể bởi vì khuếch tán là một quá trình rất nhanh khi so sánh với các quá trình khác trong hệ thống và có thể bỏ qua sự chuyển biến. Trong báo cáo này, chúng em chỉ xem xét dạng độc lập thời gian.

**II. Phương pháp số học giải quyết bài toán khuếch tán độc lập thời gian**

***1. Phương trình khuếch tán độc lập thời gian***

Đặt tất cả thời gian dẫn xuất trong phương trình khuếch tán về 0, ta thu được phương trình khuếch tán độc lập thời gian sau:

∇2c = 0

Đây là phương trình Laplace nổi tiếng mà chúng ta bắt gặp rất nhiều trong các lĩnh vực khoa học, kỹ thuật. Vì vậy rất hữu ích để tìm hiểu chi tiết phương trình này.

Chúng ta hãy xem xét lại trong một miền 2 chiều, nồng độ C không còn phụ thuộc vào biến thời gian, vì vậy chúng ta biểu diễn nồng độ C trên một điểm lưới Cl,m. Thay thế các công thức hữu hạn trong không gian dẫn xuất vào phương trình trên ta thu được:

Cl,m = [Cl+1,m + Cl-1,m + Cl,m+1 + Cl,m-1], với mọi (l,m) [2]

Phương trình [2] bao gồm (L+1) x (L+1) phương trình tuyến tính và (L+1) x (L+1) ẩn số. Các ẩn số là giá trị của nồng độ tại tất cả các điểm lưới. Công việc của chúng ta là giải hệ các phương trình này. Phương trình có thể được viết lại theo công thức sau:

Ax = b, [3]

trong đó, A là ma trận vuông NxN, X và b là Nx1 vector. Ví dụ, chúng ta sẽ chỉ ra công thức tường minh của A trong miền một chiều. Xét miền một chiều với đoạn rời rạc . Ta có

cl = c(l.) với và giả sử rằng nồng độ tại biên là cố định, tức là c0 = C0, cL = CL.

Theo phương trình Laplace trong miền 1 chiều vì vậy các phương trình hữu hạn trở thành:

Trong ví dụ này, ta thấy A là ma trận 3-đường chéo với N = L -1. Khi xét nhiều chiều hơn, A vẫn là dải nhưng có cấu trúc phức tạp hơn. Bạn có thể tự mình tìm ma trận A trong một miền 2 chiều với các điều kiện biên như ở các phần trước.

Lời giải cho hệ phương trình tuyến tính là một công việc khổng lồ, chúng ta chỉ đưa ra một số kết quả quan trọng và thảo luận tác động liên quan tới sự song song hóa. Một cuộc thảo luận sâu hơn có thể tìm đọc ở phục lục, phần E và các tài liệu tham khảo trong đó.

Có thể giải hệ phương trình tuyến tính Ax = b theo phương pháp trực tiếp (direct) hoặc lặp (iterative). Phương pháp trực tiếp giải một phương trình trực tiếp trong một số số lỗi. Trong khi, Phương pháp lặp tìm kiếm một giải pháp gần đúng thông qua một số thủ tục lặp đi lặp lại. Bảng 1 chỉ ra độ phức tạp tính toán cho cả 2 phương pháp, với ma trận dải(thưa) và dày. Với phương pháp lặp thì số bước lặp cần thiết để tìm ra lời giải với 1 độ chính xác nhất định thì nhỏ hơn N. Trong khi phương pháp trực tiếp cho phép tìm ra lời giải với độ chính xác là số nhưng với chi phí tính toán cao hơn. Đặc biệt là khi các ma trận có kích thước rất lớn, thường bắt gặp trong các bài toán mô phỏng ngày hôm nay, thì độ phức tạp tính toán trở thành 1 vấn đề cực kỳ lớn, ngay cả khi thực hiện trên các siêu máy tính có sự song song hóa cao. Đây là lý do quan trọng để chúng ta xem xét phương pháp lặp. Chúng có chi phí thấp hơn, dễ song song hóa nhưng chỉ tìm được lời giải gần đúng. Trong nhiều trường hợp, sự hội tụ rất chậm, vì vậy phương pháp lặp cần sử dụng những kỹ thuật chuyên biệt hóa (với các điều kiện ràng buộc trước) để phục hồi sự hội tụ. Vấn đề này sẽ không được thảo luận ở đây.

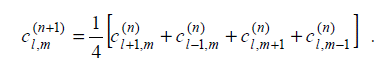
|  |  |  |
| --- | --- | --- |
|  | Ma trận dày | Ma trận dải (thưa) |
| Phương pháp trực tiếp |  |  |
| Phương pháp lặp |  |  |

Bảng 1: Độ phức tạp tính toán của phương pháp trực tiếp và lặp.

***2. Phương pháp giải***

***Phương pháp đệ quy Jacobi***

Bây giờ chúng ta quay trở lại sự chú ý của chúng tôi với các trường hợp cụ thể của chương trình khác biệt hữu hạn cho phương trình Laplace hai chiều (Eq. [14]). Phương trình này ngay lập tức cho thấy một chương trình đệ quy:

[4]

Như trước đây, n là chỉ số lần lặp. Chương trình đệ quy này được gọi là đệ quy Jacobi. Lưu ý các mối quan hệ của phương trình [4] với các chương trình khác biệt hữu hạn cho thời gian phụ thuộc vào phương trình khuếch tán, phương trình. . Nếu chúng ta lấy bởi các điều kiện cho phép CFL bước thời gian tối đa, tức là *D**x*2/*t* = ¼ và sử dụng trong phương trình. chúng ta tái tạo lặp Jacobi của phương trình. Nói cách khác, chúng ta kỳ vọng lặp Jacobi cũng sẽ bị ảnh hưởng từ một số lượng lớn tương đối của lần lặp lại cần thiết trước khi hội tụ.

Thuật toán 2 cung cấp code cho các vòng lặp bên trong của một đệ quy Jacobi. Chúng tôi một lần nữa đảm nhận miền vuông với ranh giới định kỳ trong x-hướng và ranh giới cố định trong y hướng. Hơn nữa, một tiêu chí cụ thể dừng được thực hiện. Chúng tôi yêu cầu



Sự khác biệt về nồng độ giữa hai lần lặp lại trên tất cả các điểm lưới nên được nhỏ hơn một số lượng nhỏ e. Đây là một điều kiện dừng khá là nghiêm trọng. Những người khác cũng có thể được sử dụng, ví dụ tính toán sự khác biệt có ý nghĩa và đòi hỏi điều này cần được nhỏ hơn một số lượng nhỏ. Ở đây chúng ta sẽ không phải trả bất kỳ sự chú ý hơn nữa các tiêu chí dừng lại, nhưng bạn nên nhận ra đây là một vấn đề quan trọng cần được xem xét trong bất kỳ ứng dụng mới của một phương pháp đệ quy.

/\* Jacobi update, square domain, periodic in x, fixed \*/

/\* upper and lower boundaries \*/

**do** {

= 0

**for** i=0 to max {

**for** j=0 to max {

**if**(*cij* is a source) *cij*(*n*+1) = 1.0

**else if**(*cij* is a sink) *cij*(*n*+1) = 0.0

**else** {

/\* periodic boundaries \*/

west = (i==0) ? *cmax*-1,*j*(*n*) : *ci*-1,*j*(*n*)

east = (i==max) ? *c*1,*j*(*n*) : *ci*+1,*j*(*n*)

/\* fixed boundaries \*/

south = (j==0) ? *c*0 : c*i*,*j*-1(*n*)

north = (j==max) ? *c*L : *ci*,*j*+1(*n*)

*cij*(*n*+1) = 0.25 \* (west + east + south + north)

}

/\* stopping criterion \*/

**if**(|*cij*(*n*+1) - *cij*(*n*)| > tolerance) = |*cij*(*n*+1) - *cij*(*n*)|

}

}

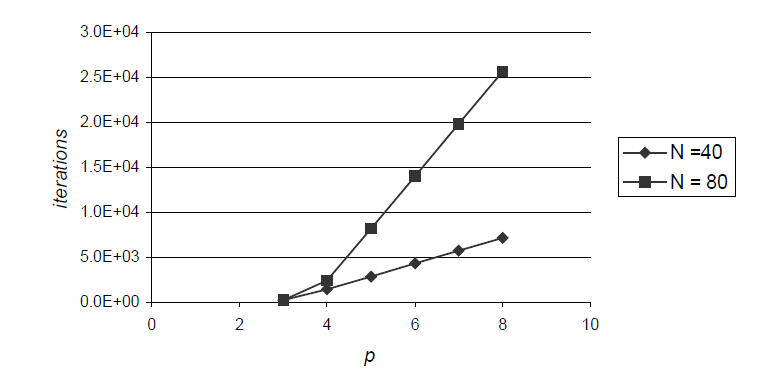
while (> tolerance)

Là một ví dụ một lần nữa chúng ta lấy miền vuông hai chiều, 0 £ x, y £ 1 với

điều kiện biên định kỳ trong x-hướng, c (x, y) = c (x +1, y), và các giá trị cố định cho

ranh giới trên và dưới, c (x, y = 0) = 0 và c (x, y = 1) = 1. Vì giải pháp đối xứng sẽ không phụ thuộc vào x, và rằng các giải pháp chính xác là một hồ sơ tập trung tuyến tính đơn giản: c (x, y) = y (lấy được kết quả này chính mình!). Sự sẵn có của các giải pháp chính xác cho phép chúng ta đo các lỗi trong một phương pháp lặp đi lặp lại như một chức năng của tiêu chí dừng áp dụng. Chúng tôi sẽ cho một số kết quả sau khi đã giới thiệu các phương pháp lặp đi lặp lại khác, cho phép so sánh. Ở đây chúng tôi đầu tiên tập trung vào số lần lặp lại cần thiết cho hội tụ.

Thiết lập số v cng b điều kiện dừng  = 10-p , với p là một số nguyên dương . Tiếp theo chúng ta đo số lần lặp lại đó là cần thiết cho sự hội tụ , và chúng tôi làm điều này cho hai kích cỡ, N = 40 và N = 80 ( N = L + 1) . Như đoán đầu tiên cho các giải pháp chúng ta chỉ mất tất cả sự tập trung dần đến số 0. Kết quả, thể hiện trong hình 2 , dường như cho thấy sự phụ thuộc tuyến tính giữa số lượng và lặp lại p , tức là phụ thuộc tuyến tính trên log ( e) . Hơn nữa, kết quả cho thấy một sự phụ thuộc N2 . Những kết quả này có thể thu được từ phân tích toán học của thuật toán , xem ví dụ [ 2, 3 ] . Phụ thuộc N2 này cũng đã được tìm thấy đối với trường hợp phụ thuộc thời gian (xem thảo luận trong phần C.1) . Cuối cùng lưu ý số lần lặp lại có thể dễ dàng trở nên lớn hơn nhiều so với số lượng các điểm (N2) và trong trường hợp đó phương pháp trực tiếp để giải quyết các phương trình được ưa thích. Tóm lại, đệ quy Jacobi hoạt động, nhưng chỉ là một ví dụ và chắc chắn không được sử dụng trong các ứng dụng thực tế. Tuy nhiên , phiên Jacobi là một bước đi hướng tới một thủ tục đệ quy rất hiệu quả .



Hình 2: số lần lặp lại cho đệ quy Jacobi như một chức năng của tình trạng dừng và như là một chức năng của số lượng các điểm lưới dọc theo mỗi chiều.

**Phương pháp *Gausse-Seidel***

Phương pháp *Jacobi* là chưa hiệu quả, chúng ta sẽ từng bước để cải tiến nó. Phương pháp *Gausse-Seidel* cũng là một ý tưởng để thực hiện điều đó. Trong phương pháp *Jacobi* chúng ta luôn sử dụng kết quả từ những vòng lặp ngay trước để cập nhật điểm, thậm chí có kết quả mới vừa được chấp nhận. Chúng ta có công thức của vòng lặp *Gausse-Seidel* :

Một ưu điểm của phương pháp này đó chính là bộ nhớ. Trong phương pháp *Jacobi* chúng ta phải cần tới 2 mảng, một để lưu kết quả cũ, một cho kết quả mới. Với *Gausse-Seidel* chúng ta sử dụng kết quả mới ngay khi có thể. Vậy chỉ cần tới một mảng để lưu chúng.

Tuy nhiên sự giảm tuyến tính với một hằng số của số vòng lặp là chưa thể thấy được, hằng số này rất gần với hệ số được dự đoán bởi thuật toán. Điều này cho thấy phương pháp này vẫn chưa có hiệu quả rõ rệt (bàng cách só sánh với thuật toán trực tiếp).

**III. Yêu cầu**

* **Cài đặt chương trình Matlab hoặc C để cài đặt phương pháp *Gausse-Seidel* giải bài toán Heat Equation.**
* **Dùng matlab để vẽ hình, tạo clip mô phỏng kết quả đạt được**

**Các tham số đầu vào sinh viên tự chọn một cách hợp lý.**

VD các tham số đầu vào :

#define l 20

#define m 20

#define epsilon 0.001

#define tolerance 0.001

**Giá trị ban đầu sinh viên tự chọn.**

Ví dụ hàm khởi tạo giá trị ban đầu:

void KhoiTao(float \*C)

{

int i,j;

for ( i = 0 ; i < l ; i++ )

for ( j = 0 ; j < m ; j++ ){

if (i==0)

\*(C+i\*m+j) = 1.0;

else

\*(C+i\*m+j) = 0.0;

}

}

**Điều kiện biên :** Khi tính tại các điểm lưới i=0, i=l-1, j=0, j=m-1 thì cần giá trị của C tại các điểm nằm ngoài miền dữ liệu sẵn có. Để có đủ thông tin, chúng ta sử dụng điều kiện biên. Giá trị của điều kiện biên phụ thuộc vào từng bài toán. Ví dụ bạn có thể chọn điều kiện biên như sau :

- Cố định giá trị: C(0,j) = 1, C(l-1,j) = 0 với mọi j

- Tuần hoàn : C(i,-1) = C(i,m-1) ; C(i,m) = C(i,0) với mọi i