tạm dịch:

- concentration gradient: sự dịch chuyển phân tử (vật chất?) từ nơi có nồng độ cao đến nơi có nồng độ thấp (1 số trang nói là thang đo nồng độ)

- thông lượng (flux): lượng vật chất chạy xuyên quan 1 khu vực cụ thể trong 1 đơn vị thgian

- sink: vùng mà nồng độ = 0

Các yêu cầu trong quá trình xây dựng và báo cáo BTL LTSS:

* Code
* Simulate(Matlab , Octave , Scilab , Python)
* SLide: 1.Bài toán ; 2.Thuật toán ; 3.Cài đặt ; 4. Demo ; 5. Kết quả
* Mỗi nhóm báo cáo trong vòng 10 đến 15

5 phút , tầm 10 đến 15 trang

GIẢI SỐ

1. Giới thiệu

Mô hình thời gian liên tục và rời rạc

Làm sao để đạt được mô phỏng (song song) cho các mô hình đó.

Hình 1: Lý thuyết (Trái) / map mô hình với máy ảo (Phải)

Mô hình toán học: Phương trình khuếch tán

Bao nhiêu cách mà lời giải số có thể đạt đc phương trình vi phân từng phần cụ thể

Sao để map sang mô hình máy ảo song song.

Phương trình khuếch tán

quy trình khuếch tán như: Nhiệt học chất rắn hay hòa tan các chất trong dung môi

Hợp chất hóa học:

- nồng độ c (đ/v: số lượng phân tử / m3)

- chỉ bị hòa tan trong dung môi (Không bay hơi)

Bảo toàn khối lượng + Định luật Fick: thông lượng khuếch tán quan hệ tuyến tình với concentration gradient => Xây dựng phương trình

concentration gradient ∇c tồn tại do thông lượng dòng (net flux) của chất tan tồn tại.

thông lượng J do đó đo bằng số lượng phân tử / m^2s

Định luật Flux: J = -D∇ c

D: hệ số khuếch tán (m2/s)

Một lượng vô cùng nhỏ dV = dxdydz

Lượng vật chất chạy khối theo hướng x: (J x ( x ) − J x ( x + d x )) d y d z tương tự 2 hướng kia

Tổng lượng vật chất tăng lên mỗi đơn vị thời gian bằng tổng lượng vật chất khuếch tán vào khối

∂ c

∂ t d V = (J x ( x ) − J x ( x + d x )) d y d z + (J y ( y ) − J y ( y + d y )) d x d z + (J z ( z ) − J z ( z + d z )) d x d y

Chia cả 2 vế cho dV, limit dx, dy, dz → 0

Kết hợp định luật Fick

∂ c

= D ∇ 2 c .

∂ t

Không thú vị: c = hằng số

Thú vị: Một số cách giải chính xác

Cần gọi biến đổi Laplace

miền ko giới hạn 1 chiều-∞ < x < ∞ . thì phương trình khuếch tán thành: ∂c/∂t = D ∂ 2 c/∂x 2

x = 0, t = 0 ví dụ khi vật chất vừa đi vào hệ thống và đc khuếch tán

Điều kiện khởi tạo này có thể đc biểu diễn bằng hàm Dirac δ(x)

c(x,t) = 0 for t < 0 and c(x, t=0) = δ(x).

Lời giải bài toán: [5]

t >0 đường cong rộng hơn biên độ nhỏ hơn

phần bên dưới đường cong vẫn là hằng số thể hiện tổng khổis lượng đc bảo toàn

có thể set c = const và t bắt đầu = 0

điều kiện biên này mô phỏng 1 lượng vô hạn chất hòa tan ??? tại 1 nồng độ = const nhất định.

môi trường ? nửa vô hạn 0 <= x < oo

Điều kiện biên được biểu diễn theo hàm Heavyside step H(t):

Với H(t) ta có thể biểu diễn điều kiện khởi tạo và điều kiện biên : c(x,t) = 0 for t < 0 and

c(x=0, t) = H(t).

Lời giải cho pt khuếch tán giờ trở thanfh: [6]

1 biến thể: bounded domain 0 ≤ x ≤ 1, nguồn là vật chất, bên kia là sink

sink có thể là mặt hấp tụ thành phần hóa học

Đk biên trở thành: c(x,t) = 0 for t < 0, c(x=0, t) = 0, and c(x=1, t) = H(t).

Lời giải trở thành [7]

Hình 6: thgian tăng, dung dịch (chất tan?) nhanh chóng khuếch tán thành trjang thái ổn định, tuyến tính với nồng độ. Khi ở trạng thái này, ko còn sự phụ thuộc vào thgian nữa, mọi yếu tố (dẫn xuất?) liên quan đến thgian có thể bị loại bỏ khỏi phương trình => chỉ cần nhìn

time-independent diffusion equation, ∂ 2 c/∂x 2 = 0.

boundary conditions c(x=0) = 0 and c(x=1) = 1, dễ chứng minh c(x) = x

vd này khác vd trc. trong vd trc nồng độ là hằng số trong 1 thgian dài, ko có sự tồn tại của concentration gradients vì thế ko tồn tại net mass flux trong hệ thống

Trong th này, trạng thái ổn định là linear profile a constant flux J = -D ∇c = -D exists in the system Do dòng vật chất hằng sô tại x = 1 and bị bỏ tại x = 0

trạn thái ổn định như thế thg đc gọi là Non-Equilibrium Steady State (NESS), nghĩa là hệ thống ở trạng thái ổn định nhưng ko trong trạng thái cân bằng nhiệt động

khuếch tán trong 1 chiều khó đặt đc

xét 2 chiều

0 ≤ x ≤ 1 and 0 ≤ y ≤ 1

boundary conditions: c ( x , y = 1 ; t ) = H ( t ) and c ( x , y = 0 ; t ) = 0 .

in the x-direction we will always assume period boundary conditions (ĐK biên thgian ???): c ( x = 0 , y ; t ) = c ( x = 1 , y ; t ) ,

hoặc tổng quát: c ( x = ξ , y ; t ) = c ( x = ξ + 1 , y ; t ) .

Do các đk biên định kì?, sự khếch tán bản chất trở thành 1 chiều và lời giải chính xác chính là [7] với x thay bởi y

[7] có ích để cho chúng ta ý nghĩa khi học độ chính xác của numerical solution of the two dimensional diffusion equation

why we need numerical solutions to the diffusion equation????

square object is situated in the domain

vật thể giả sự (đang) bị vắng m

điều kiện biên là trạng thái c on-object = 0. ĐK này ko cho lời giải (có thể?) phân tích, phương trình kheueehcs tans đc giải theo đại số sử dụng kĩ thuật sau

sự hiện diện của vật thể thay đổi quyết liệt nồng độ

dù sao lời giải trạng thái ổn định vẫn giữ nghuyên 1 hàm trơn tru , đc kì vọng từ khueehcs tán

D.3. Phương pháp (hàm?? - method) lặp đi lặp lại

Ý tưởng chung:

Tìm kiếm 1 lời giải xấp xỉ thông qua 1 vài thủ tục lặp đi lặp lại

Ý tưởng laf xem xét sự lặp lại của dạng x(n+1) = Φ (x (n) ), n = 0, 1, 2, … (BẢn sao?) n là 1 số lặp đi lặp lại

Cho trước lời giải giả định x(0) và 1 hàm lặp lại Φ cho đến khi nào tiêu chuảna hội tụ hỏa mãn, thường là ||x (n+1) - x (n) || < ε với ε là số nhỏ

Với hệ thống tuyến tính: Ax = b, một thủ tục lặp lại có thể được xây dựng:

Ax = b → Bx + (A - B)x = b

Bx (n+1) + (A - B)x (n) = b

…

nhiều khả năng để chọn ma trận B dẫn đến nhiều phương pháp lặp đi lặp lại

The Gauss-Seidel Iterative Method

trong pp Jacobi: dùng kq từ lần lặp trc để cập nhật 1 điểm dù ta đã có kết quả mới khả dụng

Ý tưởng: áp dụng kết quả mới sớm nhất có thể

để viết ct cho GS, ra phải xác định thứ tự để cập nhật điểm grid

giả sử 1 hàng khôn ngoan cập nhật thủ tục (vd ta tăng l khi giữ m fixed), ta có ct GS lăp lại:

[18]

ưu ddiemr thấy ngay của lặp GS là bộ nhớ. Jacobi cần 2 mảng, 1 lưu kq cũ, 1 lưu kq mới. GS lưu ngay kq mới sớm nhất có thể => chỉ cần 1 mảng, đặc biệt với grids lớn, việc này tiết kiệm bộ nhơ ghê gớm => ta nói có thể tính toán lặp GS tại chỗ (in place)

GS nhanh hơn Jacobi

theo lý thuyết GS cần 1 nhân tố trong 2 (vòng/lần??) lặp ít hơn Jacobi

N = 40, số vòng lặp cần cho GS và Jac trong hình 13. Lượng số vòng lặp đc giảm là rõ, gần với nhân tố của 2 như đã dự đoán trong lý thuyết, nghĩa là lặp GS vẫn chưa hiệu quả lắm so vs các phương pháp trực tiếp. Dù sao thì GS cũng chỉ cách 1 bước so vs SOR method (phương pháp rất bá)

lặp GS đặt ra thách thức mới cho tính toán song song. Đầu tiên ta cần kết luận cơ chế song song trong lặp Jac giờ phế hoàn toàn trong lặp GS. lặp GS có vẻ vốn có tuần tự như cái cách mà nó đc giới thiueje :V, với hàng thông minh (lưu trữ? sắp xếp ? trật tự) các tính toán. Tuy nhiên cái hàng thông minh trật tự này chỉ là 1 lựa chọn tiện lợi. Hóa ra là nếu ta lấy 1 thứ tự khác của tính toán, ta có thể khôi phục cơ chế song song trong GS lặp. case này hay khi mà sắp lại thứ tự tính toán lại cung cấp cơ chế song song :V. Hãy nhớ, vì nó có thể giúp bạn trong tương lai khi tìm cơ chế song song trong các thuật toán :V

GS chỉ cần lặp 4.10^3 Jac cần lặp 7.10^3 khi N = 40

Ý tưởng của việc thứ tự lại các tính toán như sau: Đầu tiên, xây dựng lưới tính toán như 1 bàn cờ checker :v, với các điểm lưới đỏ và đen. Tiếp, cho khuôn tô trong các thủ tục update chỉ mở rộng đến các hàng xóm gần nhất, nghĩa là mọi điểm đỏ độc lập với nhau (chúng chỉ phụ thuộc vào điểm đen) và ngược lại. Vì thế, thay vì hàng khôn ngoan thứ tự, ta có thể thứ tự đỏ-đen, ta update lần đầu mọi điểm đỏ, sau đó điểm đen. Chúng ta cũng gọi GS lặp này, vì mặc dù thứ tự trong các điểm lưới đc update giờ khác nhau, chúng ta còn tính toán tại chỗ và sd kết quả sớm nhất khi chúng khả dụng [Figure 14]

Thứ tưj đỏ đen này phục hồi cơ chế song song. Chúng ta h có thể cập nhật mọi điểm đỏ 1 cách song song, theo sau là cập nhật song song mọi điểm đen. Mã giả cho GS song song với thứ tự đỏ đen trong Thuật toán 4. Việc tính toán h chia thanfhc 2 ohaanf, và trc mỗi phần 1 giao tiếp với vi xử lý hàng xóm là cần thiết. Từ cái nhìn đầu tiên, có vẻ GS song song cần gấp đôi thgian giao tiếp trong phần trao đổi. But nó ếu đúng :V, vì nó ko cần thiết trao đổi toàn bộ các điểm biên, chỉ nửa thôi. Điều này bởi vì để update điểm đỏ ta chỉ cần điểm đen, cho nên chỉ có điểm biên đen mới cần trao đổi. Có nghĩa là, so sánh vs Jac song song, ta chỉ x2 thgian cài đặt cần thiết cho hoạt động trao đổi, nhưng giữ thgian gửi là hằng số. Cuối cùng điều kiện dùng song song tương tự như trc khi áp dụng

/\* only the inner loop of the parallel Gauss-Seidel method with \*/

/\* Red Black ordering \*/

do {

exchange boundary strips with neighboring processors;

for all red grid points in this processor {

update according to Gauss-Seidel iteration;

}

exchange boundary strips with neighboring processors;

for all black grid points in this processor {

update according to Gauss-Seidel iteration;

}

obtain the global maximum δ of all local δ l values

}

while ( δ > tolerance)

Algorithm 4: The pseudo code for parallel Gauss-Seidel iteration with red-black ordering.

Một nhận xét cuồi cùng khác nữa là thay vì áp dụng thứ tự đen-đỏ cho mỗi điểm lưới trong miền, ta có thể tạo ra 1 hạt thô?? thứ tự đỏ đen. Ở đây thứ tự đỏ đen đã xong ở mức lưới phân hủy??? Thủ tục h có thể đc update miền đỏ trc sau đó là update miền đen. với mỗi miền update có thể đc hoàn tất với kế hoạch của thứ tự hàng khôn ngoan ??? Vì sao???

Parallel Simulation of the Time Independent Diffusion Equation

Trạng thái cân bằng là trạng thái mà thông lượng là hằng số (chứ ko phải nồng độ là hằng số tại mọi điểm)