Tạp chí Tin học và Điều khiển học, T.22, S.4(2006), 358–376

THUẬT TOÁN SONG SONG TÍNH DÒNG CHẢY HAI CHIỀU SỬ DỤNG LƯỚI TAM GIÁC

NGUYỄN ĐỨC LẠNG[1], TRẦN GIA LỊCH[2], LÊ ĐỨC[3]

[1] *Khoa Khoa học Tự nhiên, Đại học Thái Nguyên*
[2] *Viện Toán học, Viện Khoa học và Công nghệ Việt Nam*
[3] *Trung tâm dự báo Khí tượng Thuỷ văn Trung ương*

Abstract. A model computing two-dimensional flows with arbitrary geometry is designed. Two first-order approximation techniques of spatial derivatives (the Green's theorem technique and the directional derivative technique) have been proposed for finite differences on unstructured triangular meshes. Three semi-implicit time matching methods beside the third order Adams-Bashforth scheme are used in integrating the water shallow equations written in both non-conservative and conservative forms. To remove spurious waves, a filter has been design. The model code is paralleled for both of shared memory and distributed memory computers with CSP language. The model is tested on rectangular grids triangularized after the 8-neighbours strategy. An analytical test and a real application in simulating tidal waves show reasonable results.

Tóm tắt. Một mô hình tính toán dòng chảy hai chiều được thiết kế cho miền tính có hình dạng bất kỳ. Hai kỹ thuật xấp xỉ đạo hàm không gian bậc nhất (sử dụng định lý Green và sử dụng đạo hàm hướng) được áp dụng trong sai phân hữu hạn trên lưới tam giác phi cấu trúc. Cùng với sơ đồ tích phân thời gian bậc ba *Adams-Bashforth*, ba sơ đồ tích phân bán ẩn được sử dụng tích phân hệ phương trình nước nông dưới dạng không bảo toàn hoặc bảo toàn. Để loại bỏ các sóng nhiễu xuất hiện khi tích phân, mô hình cần sử dụng thêm một bộ lọc. Mã nguồn chương trình được song song hóa cho hệ thống máy tính song song chia sẻ nhớ và máy tính song song phân tán nhớ sử dụng ngôn ngữ CSP. Kiểm thử mô hình được thực hiện trên lưới chữ nhật đã được tam giác hóa thông qua chiến thuật sử dụng tám điểm kề. Kiểm thử với một bài toán có nghiệm giải tích cũng như ứng dụng thực tế trong tính toán thủy triều đều cho các kết quả hợp lý.

1. GIỚI THIỆU

Mô hình tính dòng chảy trong sông, hồ hay tại các vùng ven biển có khả năng mô phỏng được nhiều hiện tượng tự nhiên, các hiện tượng được phân bố trên một dải rộng từ nhỏ đến lớn, đòi hỏi một lưới với độ phân giải biến đổi phụ thuộc vào từng bài toán nhất định. Do đó trong mô phỏng lưới phi cấu trúc sẽ thích hợp hơn so với lưới phân bố đều có cấu trúc. Có hai phương pháp mô hình hóa thường dùng sử dụng lưới phi cấu trúc là thể tích hữu hạn và phần tử hữu hạn. Một ô lưới dạng tam giác, tứ giác hay đa giác đóng vai trò một phần tử cơ sở, trên đó ta giải hệ phương trình đại số. Người ta thường sử dụng phương pháp thể tích hữu hạn hơn là phần tử hữu hạn do hệ phương trình dưới dạng thông lượng của nó đảm bảo tính bảo toàn của động lượng, khối lượng và năng lượng trong kết quả tính toán (một

yêu cầu quan trọng với các bài toán trong cơ học chất lỏng). Điểm mấu chốt trong phương pháp là phải đảm bảo tính duy nhất trong tính toán thông lượng trao đổi giữa hai ô lưới bất kỳ.

Ngoài hai phương pháp thông dụng cơ bản đã nêu trên người ta còn sử dụng phương pháp sai phân hữu hạn. Phương pháp sai phân hữu hạn gắn liền với việc sử dụng lưới đồng nhất (thông thường là lưới chữ nhật độ phân giải cố định). Điều này bắt nguồn từ kỹ thuật xấp xỉ đạo hàm dựa trên khai triển Taylor mà phương pháp sử dụng. Trong bài báo này một số kỹ thuật xấp xỉ đạo hàm khác sẽ được tiếp cận với mục đích mở rộng phương pháp sai phân hữu hạn trên các lưới phi cấu trúc.

2. THIẾT LẬP MÔ HÌNH

2.1. Hệ phương trình cơ sở

Mô hình được thiết lập dựa trên hệ phương trình Saint Venant hai chiều mô tả chuyển động của chất lỏng dưới dạng không bảo toàn

$$\begin{cases} \dfrac{\partial u}{\partial t} + u\dfrac{\partial u}{\partial x} + v\dfrac{\partial u}{\partial y} + g\dfrac{\partial (h+z)}{\partial x} - diff(u) - fv + \dfrac{\tau_{bx}}{\rho h} - \dfrac{\tau_{wx}}{\rho h} = 0 \\ \dfrac{\partial v}{\partial t} + u\dfrac{\partial v}{\partial x} + v\dfrac{\partial v}{\partial y} + g\dfrac{\partial (h+z)}{\partial y} - diff(v) + fu + \dfrac{\tau_{by}}{\rho h} - \dfrac{\tau_{wy}}{\rho h} = 0 \\ \dfrac{\partial h}{\partial t} + \dfrac{\partial hu}{\partial x} + \dfrac{\partial hv}{\partial y} = 0 \end{cases} \tag{1}$$

hoặc dưới dạng bảo toàn

$$\begin{cases} \dfrac{\partial p}{\partial t} + \dfrac{\partial}{\partial x}\Big(\dfrac{p^2}{h}\Big) + \dfrac{\partial}{\partial y}\Big(\dfrac{pq}{h}\Big) + gh\dfrac{\partial (h+z)}{\partial x} - diff(p) - fq + \dfrac{\tau_{bx}}{\rho} - \dfrac{\tau_{wx}}{\rho} = 0 \\ \dfrac{\partial q}{\partial t} + \dfrac{\partial}{\partial x}\Big(\dfrac{pq}{h}\Big) + \dfrac{\partial}{\partial y}\Big(\dfrac{q^2}{h}\Big) + gh\dfrac{\partial (h+z)}{\partial y} - diff(q) + fp + \dfrac{\tau_{by}}{\rho} - \dfrac{\tau_{wy}}{\rho} = 0 \\ \dfrac{\partial h}{\partial t} + \dfrac{\partial p}{\partial x} + \dfrac{\partial q}{\partial y} = 0 \end{cases} \tag{2}$$

trong đó, u, v : vận tốc dòng chảy tính trung bình theo độ sâu; p, q : lưu lượng nước; h : độ sâu; z : độ cao địa hình đáy; g : gia tốc trọng trường; $diff$: số hạng khuếch tán rối; f : tham số Coriolis; ρ : khối lượng riêng nước; $\tau_{bx},\ \tau_{by}$: ứng suất ma sát tại đáy; $\tau_{wx},\ \tau_{wy}$: ứng suất bề mặt do gió.

Ứng suất đáy có thể xác định từ công thức *Chezy* như sau

$$\frac{\tau_b}{\rho} = \frac{g|\vec{V}|\vec{V}}{(C_h)^2}$$

với C_h là hệ số *Chezy* được cho bởi

$$C_h = \frac{R^{1/6}}{n}$$

trong đó, R là bán kính thủy lực và n là hệ số nhám *Manning*. Một công thức kinh nghiệm khác cho ứng suất đáy

$$\frac{\vec{\tau}_b}{\rho h} = \frac{gn^2|\vec{V}|\vec{V}}{h^{4/3}}$$

ứng suất bề mặt do gió được tính theo công thức sau

$$\vec{\tau}_w = C_d \frac{\rho_a}{\rho} |\vec{W}| \vec{W}$$

với C_d : hệ số kéo theo; ρ_a : khối lượng riêng không khí; $\vec{W}$: gió bề mặt.

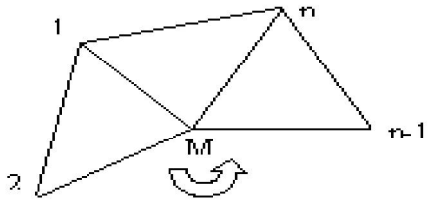
Các số hạng rối được thiết lập theo các sơ đồ vận chuyển động lượng rối dưới dạng bậc hai, bậc bốn hay bậc sáu. Một sơ đồ nếu được lựa chọn thích hợp sẽ lọc bỏ các sóng nhiễu xuất hiện trong nghiệm tính toán dưới dạng các sóng ngắn cao tần trong quá trình tích phân, ngoài ra còn đảm bảo độ ổn định của sơ đồ số.

2.2. Xấp xỉ đạo hàm không gian

Giả định f là hàm số mà ta muốn xác định đạo hàm không gian của nó một cách gần đúng. Do mô hình được thiết kế trên lưới tam giác nên các kỹ thuật mới xấp xỉ đạo hàm không gian cần được xây dựng một cách thích hợp khác với kỹ thuật khai triển *Taylor* trên lưới đều chữ nhật thông thường. Có hai kỹ thuật cho phép xấp xỉ các đạo hàm không gian: sử dụng định lý *Green* và xác định đạo hàm theo hướng.

2.2.1. Sử dụng định lý Green

Giả sử M là điểm tại đó ta muốn xác định các đạo hàm không gian. Đánh số các điểm xung quanh M trên lưới tam giác theo chiều ngược chiều kim đồng hồ lần lượt là $1, 2, \cdots, n$ (Hình 1)



Hình 1. Sơ đồ các điểm lưới minh họa cách thiết lập công thức xấp xỉ các đạo hàm không gian theo định lý Green

Diện tích đa giác hình thành bởi các cạnh $12, 23, \cdots, n1$ được ký hiệu là S, chu vi được ký hiệu là C. Áp dụng định lý *Green* với tích phân thực hiện trên miền S theo các đạo hàm không gian của f ta được

$$\begin{cases} \iint\limits_S \dfrac{\partial f}{\partial x} dS = \oint\limits_C f \cos(\vec{n}, x) dC = \oint\limits_C f dy \\ \iint\limits_S \dfrac{\partial f}{\partial y} dS = -\oint\limits_C f \cos(\vec{n}, y) dC = -\oint\limits_C f dx \end{cases}$$

với $\vec{n}$ là vector pháp tuyến ngoài của C. Từ đây ta được

$$\begin{cases} \overline{\dfrac{\partial f}{\partial x}} = \dfrac{1}{S} \oint\limits_C f dy \\ \overline{\dfrac{\partial f}{\partial y}} = -\dfrac{1}{S} \oint\limits_C f dx \end{cases}$$

Bây giờ ta giả định các đạo hàm không gian không biến đổi trong S

$$\begin{cases} \overline{\dfrac{\partial f(x,y)}{\partial x}} = \dfrac{\partial f(x,y)}{\partial x}; & (x,y) \in S \\ \dfrac{\partial f(x,y)}{\partial y} = \dfrac{\partial f(x,y)}{\partial y}; & (x,y) \in S \end{cases}$$

và đạo hàm không gian tại M có thể tính được theo công thức xấp xỉ sau

$$\begin{cases} \left.\dfrac{\partial f}{\partial x}\right|_M = \dfrac{1}{S}\oint\limits_C f dy + O(h_x) \\ \left.\dfrac{\partial f}{\partial y}\right|_M = -\dfrac{1}{S}\oint\limits_C f dx + O(h_y) \end{cases} \tag{3}$$

trong đó, h_x, h_y là khoảng cách cực đại từ M tới các đỉnh của đa giác S tương ứng theo trục x và y.

Tích phân đường theo đường cong kín C trong (3) là tổng tích phân theo các cạnh của đa giác S. Thực hiện xấp xỉ tuyến tính đơn giản cho mỗi tích phân này ta được

$$\begin{aligned} \oint\limits_C f dy &= \int\limits_{12} f dy + \int\limits_{23} f dy + \cdots + \int\limits_{n1} f dy \\ &= \frac{f_1 + f_2}{2}(y_2 - y_1) + \frac{f_2 + f_3}{2}(y_3 - y_2) + \cdots + \frac{f_n + f_1}{2}(y_1 - y_n) \\ \oint\limits_C f dx &= \int\limits_{12} f dx + \int\limits_{23} f dx + \cdots + \int\limits_{n1} f dx \\ &= \frac{f_1 + f_2}{2}(x_2 - x_1) + \frac{f_2 + f_3}{2}(x_3 - x_2) + \cdots + \frac{f_n + f_1}{2}(x_1 - x_n) \end{aligned}$$

Cũng áp dụng định lý *Green* khi tính diện tích S ta có

$$S = \iint\limits_S dS = \oint\limits_C x dy = -\oint\limits_C y dx$$

với f có thể là x hoặc y. Từ đây ta có hai công thức xấp xỉ đạo hàm không gian tại M như sau

$$\begin{cases} \left.\dfrac{\partial f}{\partial x}\right|_M = \dfrac{f_1 y_2 - f_2 y_1 + \cdots + f_n y_1 - f_1 y_n}{x_1 y_2 - x_2 y_1 + \cdots + x_n y_1 - x_1 y_n} \\ \left.\dfrac{\partial f}{\partial y}\right|_M = -\dfrac{f_1 x_2 - f_2 x_1 + \cdots + f_n x_1 - f_1 x_n}{x_1 y_2 - x_2 y_1 + \cdots + x_n y_1 - x_1 y_n} \end{cases} \tag{4}$$

Các đạo hàm không gian cấp cao hơn cũng được xấp xỉ tương tự. Để xác định đạo hàm cấp n trước hết ta xác định đạo hàm cấp $(n-1)$ tại tất cả các điểm trong miền tính, sau đó áp dụng công thức (4) với f là các đạo hàm cấp $(n-1)$. Dưới đây là công thức xác định các đạo hàm cấp hai từ các đạo hàm bậc nhất

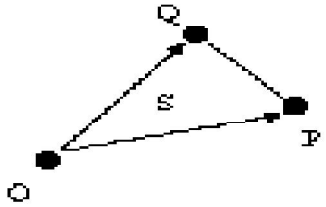
$$\begin{cases} \left.\dfrac{\partial^2 f}{\partial x^2}\right|_M = \dfrac{f'_1 y_2 - f'_2 y_1 + \cdots + f'_n y_1 - f'_1 y_n}{x_1 y_2 - x_2 y_1 + \cdots + x_n y_1 - x_1 y_n} \\ \left.\dfrac{\partial^2 f}{\partial y^2}\right|_M = -\dfrac{f'_1 x_2 - f'_2 x_1 + \cdots + f'_n x_1 - f'_1 x_n}{x_1 y_2 - x_2 y_1 + \cdots + x_n y_1 - x_1 y_n} \end{cases}$$

2.2.2. Sử dụng đạo hàm theo hướng

Nếu $\vec{n}$ là một vector làm với vector đơn vị trên trục Ox một góc α thì đạo hàm không gian của f theo hướng của vector $\vec{n}$ sẽ được cho bởi

$$f'_{\vec{n}} = \frac{\partial f}{\partial x}\cos\alpha + \frac{\partial f}{\partial y}\sin\alpha. \tag{5}$$

Bây giờ giả sử O là điểm tại đó ta muốn xấp xỉ các đạo hàm không gian. Để thực hiện điều này ta sử dụng đạo hàm theo hướng tại O tới hai điểm lân cận là P và Q (Hình 2).



Hình 2. Sơ đồ các điểm lưới minh họa cách thiết lập công thức xấp xỉ các đạo hàm không gian từ đánh giá đạo hàm theo hướng

Trước hết ta định nghĩa góc giữa các vector $\vec{i}, \overrightarrow{OP}, \overrightarrow{OQ}$ như sau

$$\alpha_P = \widehat{\overrightarrow{OP}, \vec{i}}; \quad \alpha_Q = \widehat{\overrightarrow{OQ}, \vec{i}}$$

Sử dụng (5) với vector $\vec{n}$ lần lượt là $\overrightarrow{OP}$ và $\overrightarrow{OQ}$ ta có

$$\begin{cases} f'_{\overrightarrow{OP}} = \dfrac{\partial f}{\partial x}\Big|_O \cos\alpha_P + \dfrac{\partial f}{\partial y}\Big|_O \sin\alpha_P = \dfrac{f_P - f_O}{OP} + O(OP) \\ f'_{\overrightarrow{OQ}} = \dfrac{\partial f}{\partial x}\Big|_O \cos\alpha_Q + \dfrac{\partial f}{\partial y}\Big|_O \sin\alpha_Q = \dfrac{f_Q - f_O}{OQ} + O(OQ) \end{cases}$$

Bỏ qua các số hạng bậc cao, ta có

$$\begin{cases} \dfrac{\partial f}{\partial x}\Big|_O \cos\alpha_P + \dfrac{\partial f}{\partial y}\Big|_O \sin\alpha_P = \dfrac{f_P - f_O}{OP} \\ \dfrac{\partial f}{\partial x}\Big|_O \cos\alpha_Q + \dfrac{\partial f}{\partial y}\Big|_O \sin\alpha_Q = \dfrac{f_Q - f_O}{OQ} \end{cases}$$

Giải hệ này với một số biến đổi ta thu được nghiệm sau

$$\begin{cases} \dfrac{\partial f}{\partial x}\Big|_O = \dfrac{1}{2S}[f_O(y_P - y_Q) + f_P(y_Q - y_O) + f_Q(y_O - y_P)] \\ \dfrac{\partial f}{\partial y}\Big|_O = -\dfrac{1}{2S}[f_O(x_P - x_Q) + f_P(x_Q - x_O) + f_Q(x_O - x_P)] \end{cases} \tag{6}$$

Quay trở lại với Hình 1, khi ta muốn xấp xỉ đạo hàm không gian tại M mà lân cận M là một tập hợp n điểm. Như vậy áp dụng (6) cho mỗi tam giác $M12, M23, \cdots, Mn1$ ta có được n đánh giá khác nhau cho đạo hàm không gian tại M. Cách đơn giản nhất để thu được một công thức xác định trong trường hợp này là lấy trung bình của tất cả các đánh giá này

$$\begin{cases} \dfrac{\partial f}{\partial x}\Big|_O = \dfrac{1}{n}\Big(\dfrac{\partial f}{\partial x}\Big|_{M12} + \dfrac{\partial f}{\partial x}\Big|_{M23} + \cdots + \dfrac{\partial f}{\partial x}\Big|_{Mn1}\Big) \\ \dfrac{\partial f}{\partial y}\Big|_O = \dfrac{1}{n}\Big(\dfrac{\partial f}{\partial y}\Big|_{M12} + \dfrac{\partial f}{\partial y}\Big|_{M23} + \cdots + \dfrac{\partial f}{\partial y}\Big|_{Mn1}\Big) \end{cases}$$

Trung bình hóa có thể thực hiện phức tạp hơn nếu ta tính đến trọng số cho mỗi tam giác phụ thuộc vào diện tích cũng như khoảng cách của nó đến M. Tuy nhiên trong các phần sau ta chỉ thực hiện đơn giản là trung bình số học.

Tính toán với các đạo hàm bậc cao cũng được thực hiện tương tự như đã nói trong phần sử dụng định lý *Green.*

2.3. Phương pháp tích phân

Có nhiều phương pháp tích phân thời gian khác nhau có thể tìm được trong các tài liệu động lực học chất lỏng tính toán. Ta có thể lựa chọn một phương pháp thích hợp kết hợp với hai kỹ thuật xấp xỉ đạo hàm không gian đã nói trên. Sơ đồ *Adams-Bashforth* bậc ba sẽ là một lựa chọn tốt nhờ độ chính xác cao (bậc ba về thời gian) và hiệu quả kinh tế trong tính toán (đây là phương pháp hiện). Chúng tôi đã lựa chọn sử dụng sơ đồ này cho hệ phương trình cơ bản dưới dạng bảo toàn (2). Ký hiệu f là đại lượng mà ta muốn tích phân theo thời gian, sơ đồ *Adams-Bashforth* bậc ba có dạng sau

$$f^{n+1} = f^n + \frac{1}{12}\Big(23\frac{\partial f^n}{\partial t} - 16\frac{\partial f^{n-1}}{\partial t} + 5\frac{\partial f^{n-2}}{\partial t}\Big)\Delta t$$

với chỉ số trên là chỉ số thời gian, Δt là bước thời gian.

Với hệ phương trình dạng không bảo toàn (1) chúng tôi sử dụng phương pháp bán ẩn thông qua xử lý ẩn các số hạng bình lưu và xây dựng ba phương pháp bán ẩn khác nhau, chúng được lần lượt trình bày dưới đây.

Phương pháp 1: Viết lại hệ phương trình (1) dưới dạng sau với các số hạng ở vế phải được biểu diễn chung bằng đại lượng F

$$\begin{cases} \dfrac{\partial u}{\partial t} + u\dfrac{\partial u}{\partial x} = F_u \\ \dfrac{\partial v}{\partial t} + v\dfrac{\partial v}{\partial y} = F_v \\ \dfrac{\partial h}{\partial t} + h\Big(\dfrac{\partial u}{\partial x} + \dfrac{\partial u}{\partial y}\Big) = F_h \end{cases}$$

và sai phân hóa ta được

$$\begin{cases} \dfrac{u_i^n - u_i^{n-1}}{\Delta t} + u_i^n \dfrac{\partial u}{\partial x}\Big|_i^{n-1} = F_{ui}^{n-1} \\ \dfrac{v_i^n - v_i^{n-1}}{\Delta t} + v_i^n \dfrac{\partial v}{\partial y}\Big|_i^{n-1} = F_{vi}^{n-1} \\ \dfrac{h_i^n - h_i^{n-1}}{\Delta t} + h_i^n \Big(\dfrac{\partial u}{\partial x} + \dfrac{\partial v}{\partial y}\Big)\Big|_i^{n-1} = F_{hi}^{n-1} \end{cases} \tag{7}$$

Các đạo hàm không gian trong (7) được xấp xỉ theo hai kỹ thuật chính đã nhắc tới trong

phần trên. Giải hệ phương trình này ta thu được lời giải hiện dưới dạng sau

$$\begin{cases} u_i^n = \dfrac{u_i^{n-1} + \Delta t F_{ui}^{n-1}}{1 + \Delta t\Big(\dfrac{\partial u}{\partial x}\Big)\Big|_i^{n-1}} \\ v_i^n = \dfrac{v_i^{n-1} + \Delta t F_{vi}^{n-1}}{1 + \Delta t\Big(\dfrac{\partial v}{\partial y}\Big)\Big|_i^{n-1}} \\ h_i^n = \dfrac{h_i^{n-1} + \Delta t F_{hi}^{n-1}}{1 + \Delta t\Big(\dfrac{\partial u}{\partial x} + \dfrac{\partial v}{\partial y}\Big)\Big|_i^{n-1}} \end{cases} \tag{8}$$

Đây là phương pháp tích phân bán ẩn thứ nhất.

Phương pháp 2: hệ phương trình (1) được viết lại

$$\begin{cases} \dfrac{\partial u}{\partial t} + u\dfrac{\partial u}{\partial x} + v\dfrac{\partial u}{\partial y} = F_u \\ \dfrac{\partial v}{\partial t} + u\dfrac{\partial v}{\partial x} + v\dfrac{\partial v}{\partial y} = F_v \\ \dfrac{\partial h}{\partial t} + h\Big(\dfrac{\partial u}{\partial x} + \dfrac{\partial v}{\partial y}\Big) + u\dfrac{\partial h}{\partial x} + v\dfrac{\partial h}{\partial y} = 0. \end{cases}$$

Sai phân hóa hệ này ta được

$$\begin{cases} \dfrac{u_i^n - u_i^{n-1}}{\Delta t} + u_i^n \dfrac{\partial u}{\partial x}\Big|_i^{n-1} + v_i^n \dfrac{\partial u}{\partial y}\Big|_i^{n-1} = F_{ui}^{n-1} \\ \dfrac{v_i^n - v_i^{n-1}}{\Delta t} + u_i^n \dfrac{\partial v}{\partial x}\Big|_i^{n-1} + v_i^n \dfrac{\partial v}{\partial y}\Big|_i^{n-1} = F_{vi}^{n-1} \\ \dfrac{h_i^n - h_i^{n-1}}{\Delta t} + h_i^n \Big(\dfrac{\partial u}{\partial x} + \dfrac{\partial v}{\partial y}\Big)\Big|_i^{n-1} + u_i^n \dfrac{\partial h}{\partial x}\Big|_i^{n-1} + v_i^n \dfrac{\partial h}{\partial y}\Big|_i^{n-1} = 0. \end{cases}$$

Hệ này cho ta một hệ phương trình tuyến tính ba ẩn

$$\begin{cases} a_1 u_i^n + b_1 v_i^n = d_1 \\ a_2 u_i^n + b_2 v_i^n = d_2 \\ a_3 u_i^n + b_3 v_i^n + c_3 h_i^n = 0 \end{cases}$$

trong đó,

$$\begin{cases} a_1 = 1 + \Delta t \dfrac{\partial u}{\partial x}\Big|_i^{n-1}; \quad b_1 = \Delta t \dfrac{\partial u}{\partial y}\Big|_i^{n-1}; \quad d_1 = u_i^{n-1} + \Delta t F_{ui}^{n-1}; \\ a_2 = \dfrac{\partial v}{\partial x}\Big|_i^{n-1}; \quad b_2 = 1 + \Delta t \dfrac{\partial v}{\partial y}\Big|_i^{n-1}; \quad d_2 = v_i^{n-1} + \Delta t F_{vi}^{n-1}; \\ a_3 = \Delta t \dfrac{\partial h}{\partial x}\Big|_i^{n-1}; \quad b_3 = \Delta t \dfrac{\partial h}{\partial y}\Big|_i^{n-1}; \quad c_3 = 1 + \Delta t\Big(\dfrac{\partial u}{\partial x} + \dfrac{\partial v}{\partial y}\Big)\Big|_i^{n-1} \end{cases}$$

Giải hệ ta được

$$\begin{cases} u_i^n = \dfrac{d_1 b_2 - d_2 b_1}{a_1 b_2 - a_2 b_1} \\ v_i^n = \dfrac{a_1 d_2 - a_2 d_1}{a_1 b_2 - a_2 b_1} \\ h_i^n = -\dfrac{a_3 u_i^n + b_3 v_i^n}{c_3} \end{cases}$$

Phương pháp 3: viết lại hệ phương trình (1) dưới dạng

$$\begin{cases} \dfrac{\partial u}{\partial t} + u\dfrac{\partial u}{\partial x} + v\dfrac{\partial u}{\partial y} + g\dfrac{\partial h}{\partial x} = F_u \\ \dfrac{\partial v}{\partial t} + u\dfrac{\partial v}{\partial x} + v\dfrac{\partial v}{\partial y} + g\dfrac{\partial h}{\partial y} = F_v \\ \dfrac{\partial h}{\partial t} + h\Big(\dfrac{\partial u}{\partial x} + \dfrac{\partial v}{\partial y}\Big) + u\dfrac{\partial h}{\partial x} + v\dfrac{\partial h}{\partial y} = 0 \end{cases}$$

sai phân hoá được

$$\begin{cases} \dfrac{u_i^n - u_i^{n-1}}{\Delta t} + u_i^{n-1}\dfrac{\partial u}{\partial x}\Big|_i^n + v_i^{n-1}\dfrac{\partial u}{\partial y}\Big|_i^n + g\dfrac{\partial h}{\partial x}\Big|_i^n = F_{ui}^{n-1} \\ \dfrac{v_i^n - v_i^{n-1}}{\Delta t} + u_i^{n-1}\dfrac{\partial v}{\partial x}\Big|_i^n + v_i^{n-1}\dfrac{\partial v}{\partial y}\Big|_i^n + g\dfrac{\partial h}{\partial y}\Big|_i^n = F_{vi}^{n-1} \\ \dfrac{h_i^n - h_i^{n-1}}{\Delta t} + h_i^{n-1}\Big(\dfrac{\partial u}{\partial x} + \dfrac{\partial v}{\partial y}\Big)\Big|_i^n + u_i^{n-1}\dfrac{\partial h}{\partial x}\Big|_i^n + v_i^{n-1}\dfrac{\partial h}{\partial y}\Big|_i^n = 0 \end{cases}$$

Với N điểm trong miền tính cho ta một $3N$ phương trình với $3N$ ẩn. Đây là một hệ phương trình đại số tuyến tính với ma trận hệ số là ma trận thưa. Do đó ta có thể giải hệ này bằng nhiều phương pháp lặp khác nhau.

Khác biệt cơ bản của phương pháp này với hai phương pháp trên nằm trong cách xử lý ẩn số hạng bình lưu. Nếu hai phương pháp trước xử lý ẩn tốc độ dòng chảy thì phương pháp này xử lý ẩn các số hạng đạo hàm không gian.

Trên thực tế ba phương pháp bán ẩn nêu trên được thực hiện phức tạp hơn. Tại mỗi bước tích phân thời gian một chương trình lặp giải được thực hiện nhằm nâng cao độ chính xác của kết quả. Điều này có nghĩa rằng ta không xác định ngay giá trị biến tại thời điểm kế tiếp mà đưa vào một quá trình lặp giải cho tới khi đạt tới một độ chính xác cần thiết. Quá trình này có thể mô tả ngắn gọn như sau: biểu diễn các biến mô hình bởi vector $\vec{q}$, giá trị các biến mô hình tại thời điểm $(n+1)$ có thể được biểu diễn qua một công thức tổng quát từ giá trị các biến tại thời điểm n:

$$\vec{q}^{n+1} = F(\vec{q}^{n})$$

Quá trình lặp giải sẽ được biểu diễn dưới dạng giải mã:
$q = q^n$
do
$q_0 = q$
$q = F(q)$
$error = max|q_i - q_{0i}|$
while (error > epsilonq_{0i})*
$q^{n+1} = q$
trong đó, *error* là sai số cực đại giữa hai lần lặp liên tiếp, *epsilon* là sai số tương đối cho phép nhằm xác định sự hội tụ (trong phần này chúng tôi lấy bằng $1^{0/00}$), q_0 là biến trung gian lưu trữ giá trị các biến mô hình cho lần lặp hiện tại giúp xác định giá trị sai số, chỉ số dưới i xác định điểm tại đó sai số cực đại. Với sai số lựa chọn là $1^{0/00}$, thông thường số lần lặp nằm trong khoảng từ 2 đến 5.

Ví dụ khi lặp giải với công thức (8), tính giá trị các biến tại thời điểm n từ giá trị các biến tại thời điểm $(n-1)$, tại lần lặp thứ j ta có

$$\begin{cases} u_i^{(j)} = \dfrac{u_i^{(j-1)} + \Delta t F_{ui}^{(j-1)}}{1 + \Delta t \Big(\dfrac{\partial u}{\partial x}\Big)\Big|_i^{(j-1)}} \\ v_i^{(j)} = \dfrac{v_i^{(j-1)} + \Delta t F_{vi}^{(j-1)}}{1 + \Delta t \Big(\dfrac{\partial v}{\partial y}\Big)\Big|_i^{(j-1)}} \\ h_i^{(j)} = \dfrac{h_i^{(j-1)} + \Delta t F_{hi}^{(j-1)}}{1 + \Delta t \Big(\dfrac{\partial u}{\partial x} + \dfrac{\partial v}{\partial y}\Big)\Big|_i^{(j-1)}} \end{cases}$$

Riêng tại lần lặp đầu tiên $j = 0$ ta lấy giá trị các biến ban đầu bằng giá trị của chúng tại thời điểm hiện tại

$$u_i^{(0)} = u_i^{n-1}; \quad v_i^{(0)} = v_i^{n-1}; \quad h_i^{(0)} = h_i^{n-1}$$

Phép lặp sẽ kết thúc khi cực đại sai số tương đối, nhỏ hơn một giá trị ε cho trước nào đó

$$\max\Big(\frac{h_i^{(j)} - h_i^{(j-1)}}{h_i^{(j-1)}}\Big) < \varepsilon$$

Cuối cùng ta có được giá trị các biến cho bước thời gian kế tiếp n : u_i^n, v_i^n và h_i^n.

2.4. Điều kiện biên

Nếu chuyển các số hạng rối sang phải và xem như một hàm đã biết, hệ phương trình (1) có thể biến đổi thành dạng đối xứng có tính chất hyperbolic tựa đối xứng. Các giá trị riêng của ma trận *Jacobian* của hệ này cho ta tốc độ pha của các sóng đi vào và đi ra miền tính. Các tốc độ này phụ thuộc vào thành phần vận tốc pháp tuyến u_n và vận tốc sóng trọng trường c. Do đó số điều kiện biên cần cho trước bằng số sóng đi vào miền tính. Bởi vậy số điều kiện biên phụ thuộc vào từng bài toán cụ thể, đòi hỏi ta phải thiết kế một bộ khung tổng thể trong xử lý điều kiện biên. Dưới đây là các dạng điều kiện biên được mô hình xử lý

Điều kiện biên cho trước có thể là vận tốc dòng chảy u, v; hoặc lưu lượng p, q; hoặc độ sâu mực nước h.

Điều kiện biên tại biên cứng: thành phần vector pháp tuyến u_n bằng không.

Điều kiện biên tại biên lỏng: là một hàm cho trước hoặc điều kiện phát xạ.

Phụ thuộc vào số điều kiện biên cho trước mà ta cần bổ xung một số các phương trình trên biên. Sử dụng phương pháp đặc trưng (xem [10]) đã thiết lập dạng của các phương trình bổ xung này trong trường hợp đường biên song song với các trục tọa độ. Tuy nhiên do mô hình được xây dựng dựa trên lưới tam giác nên những phương trình như vậy không thể áp dụng trực tiếp, do đó chúng tôi hướng đến một cách tiếp cận đơn giản hơn. Nếu giá trị một biến không được cho trước, giá trị của nó sẽ được xác định từ các phương trình sai phân trên biên.

2.5. Lọc nhiễu

Bộ lọc sẽ được thực hiện tại tất cả các điểm sau một số bước thời gian nào đó với một số lần cho trước. Dưới đây là ví dụ lọc loại bỏ sóng cao tần với độ sâu h. Giá trị của h tại M (Hình 2.1) được xác định như sau

$$\hat{h}_M = (1-w)h_M + w\overline{h_M} \tag{9}$$

với w là trọng số lọc (được lấy bằng 0.02 trong mô hình này) và trung bình của h được tính từ tất cả các điểm xung quanh

$$\overline{h_M} = \frac{1}{n}(h_1 + h_2 + \cdots + h_n)$$

Sử dụng bộ lọc (9) tính bảo toàn khối lượng có thể bị vi phạm, tuy nhiên khi thử nghiệm với các bài toán khác nhau những vi phạm này là không đáng kể như trình bày trong phần dưới đây.

2.6. Song song hóa

Nhằm tăng cường hiệu năng tính toán (chủ yếu là về mặt thời gian) của mô hình chúng tôi đã thực hiện song song hóa chương trình. Có hai hướng song song hóa chính liên quan đến việc hệ máy tính hỗ trợ bộ nhớ chia sẻ hay bộ nhớ phân tán. Chúng tôi đã thực hiện song song hóa theo cả hai hướng này sử dụng ngôn ngữ mô tả song song CSP (Communicating Sequential Processes) đã được thực hiện trên Java dưới dạng thư viện JCSP (Java Communicating Sequential Processes).

Tác giả của CSP là Hoare [5] khi ông đa ra ý tưởng tương tác của các tiến trình song song qua kênh tin. Mô hình CSP có một nền tảng toán học vững chắc đã được kiểm chứng và tiếp tục phát triển trong khoảng 20 năm qua. Nó được thực hiện trong các ngôn ngữ như Occam và Lotus, được tích hợp vào trong các thư viện của C, C++ và Java. CSP có ứng dụng hiển nhiên nhất trong đặc tả, thiết kế, thực hiện hệ thống tính toán. Ý tưởng cơ bản là phân giải hệ thống thành các tiến trình song song tương tác lẫn nhau. Kết cấu song song giữa các tiến trình cũng đơn giản như kết cấu tuần tự của các câu lệnh trong ngôn ngữ lập trình truyền thống.

2.6.1. Song song hóa trên hệ chia sẻ nhớ

Trong phần này và phần sau chúng tôi thực hiện song song hóa cho mô hình tính toán dòng chảy hai chiều sử dụng sơ đồ tích phân *Adam-Bashforth* bậc ba.

Nhìn chung song song hóa trên hệ chia sẻ nhớ luôn đơn giản hơn so với song song hóa trên hệ phân tán nhớ. Khi không gian dữ liệu là chung cho mọi bộ xử lý trên hệ chia sẻ nhớ nhiệm vụ cơ bản sẽ nằm ở song song hóa tác vụ. Với mô hình của ta điều này thực hiện dễ dàng do tính toán tại mỗi điểm chỉ liên quan đến các điểm kề với nó. Mỗi bộ xử lý sẽ chỉ cần tập trung xử lý trên tập hợp các điểm trong miền tính được xác định từ đầu cho nó. Tuy nhiên tại mỗi bước thời gian các bộ xử lý phải được đồng bộ, đảm bảo mức thời gian tích phân, không có điểm nào thực hiện nhanh hơn điểm nào.

Dưới đây là đoạn mã nguồn chương trình chính tích phân thời gian

```
for(int it=0;it<nt;it++)
{   if (update) setBoundary((it+1)*dt);
    for(int k=0;k<n;k++) point[k].computeFlux();
    for(int k=0;k<n;k++) point[k].derivative();
    for(int k=0;k<n;k++) point[k].integrate(it);
    if ((filter) && (it%filterStep == 0))
```

```
{ for(int i=0;i<npass;i++)
    { for(int k=0;k<n;k++) point[k].filter(weight);
      for(int k=0;k<n;k++) point[k].update();}}}
```

Ở đây *point* là mảng một chiều quản lý các điểm lưới trong miền tính, *nt* là số bước thời gian, *dt* là bước thời gian, *it* là chỉ số chạy của thời gian, *n* là số điểm lưới, *k* là chỉ số chạy của các điểm, *npass* là số lần lặp, *i* là chỉ số lặp, *weight* là trọng số lặp, *filterStep* là chu kỳ lặp tính theo số bước thời gian, *update* cho biết ta có cập nhật biên hay không, *filter* cho biết ta có thực hiện lọc hay không. Cách thực hiện của chương trình khá đơn giản: trước hết ta thực hiện cập nhật giá trị trên biên, tính các thông lượng từ đó xác định được đạo hàm không gian. Sau khi tích phân ta sẽ thực hiện lọc kết quả nếu được yêu cầu.

Từ đoạn mã trên ta thấy trong mỗi bước thời gian ta cần hoàn thành năm công đoạn tính toán bao gồm tính toán thông lượng, xác định đạo hàm, tính tích phân, lọc và cập nhật dữ liệu trong đó hai công đoạn cuối cùng là tùy chọn. Phân chia tính toán thành năm công đoạn là bắt buộc do mỗi công đoạn không thể bắt đầu nếu công đoạn khác chưa kết thúc. Bởi vậy việc song song hóa đoạn mã này cũng đòi hỏi ta phải thực hiện song song năm công đoạn tương ứng. Dưới đây là đoạn mã sau khi được song song hóa

```
for(int it=0;it<nt;it++)
{   { if (update) setBoundary((it+1)*dt); computingFluxParallel.run();
    derivativeParallel.run();
    for(int k=0;k<nThread;k++) integrateProcess[k].setIT(it);
    integrateParallel.run();
    if ((filter) && (it%filterStep == 0))
    { for(int i=0;i<npass;i++)
        { filterParallel.run();
          updateParallel.run();}}}
```

Như thế ngữ nghĩa của đoạn mã song song hoàn toàn giống như ngữ nghĩa của đoạn mã tuần tự đã nêu trên. Dễ thấy mỗi công đoạn đều thực hiện tính toán tuần tự trên tập hợp các điểm trong miền tính, thể hiện qua vòng lặp *for* trong chương trình đã được chuyển thành một tập các tiến trình tính toán song song. Như vậy khi song song hóa ta sẽ phân phối các điểm này một cách cân bằng cho các bộ xử lý. Ví dụ với một tiến trình song song tính toán đạo hàm

```
class DerivativeProcess implements CSProcess
{   private int kstart, kend;
    private Barrier barrier;
    public DerivativeProcess(int kstart, int kend, Barrier barrier)
    { this.kstart = kstart;
    this.kend = kend;
    this.barrier = barrier;}
    public void run()
    { for(int k=kstart;k<kend+1;k++) point[k].derivative();
      barrier.sync();}}
```

Tiến trình tính đạo hàm được thể hiện dưới dạng một lớp nội *DerivativeProcess* bao gồm ba thông số cơ bản là chỉ số bắt đầu *start*, kết thúc *end* của miền tính toán riêng cho tiến

trình này và tiến trình đồng bộ *barrier* để tiến trình có thể thực hiện đồng bộ với các tiến trình khác.

Cơ chế đồng bộ *Barrier* được thực hiện thông qua phương thức *sync()*. Với tham số *nThread* là số tiến trình, *Barrier* sẽ chỉ được mở ra khi đã có đầy đủ *nThread* tiến trình có lời gọi đồng bộ *sync()*. Nếu chưa đủ, tiến trình thực hiện lời gọi *sync()* sẽ bị treo tại điểm thực hiện lời gọi cho tới khi *Barrier* được mở ra cho phép tiến trình tiếp tục thực hiện.

Cách phân chia miền tính toán cho mỗi tiến trình được thực hiện đơn giản bằng cách phân chia đều n điểm của miền tính cho số tiến trình *nThread*. Chỉ số đầu và cuối cho mỗi tiến trình được tính như sau

```
for(int k=0;k<nThread;k++)
{   start[k] = k*(n/nThread);
    end[k] = (k+1)*(n/nThread)-1;}
end[nThread-1] = n-1;
```

Cách thực hiện tính toán song song trên hệ máy tính chia sẻ nhớ với *Java* có điểm mạnh là số tiến trình không nhất thiết phải bằng số bộ vi xử lý của hệ thống. Người sử dụng hoàn toàn được độc lập trong lựa chọn số tiến trình song song. Trên cơ sở này *Java* sẽ phân bố các tiến trình vào số các bộ vi xử lý.

2.6.2. Song song hóa trên hệ phân tán nhớ

Song song hóa trên hệ phân tán nhớ thực hiện khó hơn so với song song hóa trên hệ chia sẻ nhớ. Bởi thực chất bài toán của ta là bài toán song song về dữ liệu nên thực hiện trên hệ chia sẻ nhớ khá đơn giản với một không gian địa chỉ chung. Khi chuyển sang hệ phân tán, trao đổi dữ liệu giữa các tiến trình trở nên quan trọng khi các tiến trình phải trao đổi kết quả tính toán trên biên cho nhau. Ngoài ra bài toán đồng bộ giữa các tiến trình cũng có một tầm quan trọng nhất định.

Để thực hiện phần này chúng tôi thực hiện tính toán song song trong một mạng máy tính, tận dụng các máy tính khác nhau cho các tiến trình khác nhau. Tính toán được thực hiện trên các tiến trình *client*. Tất cả các tiến trình tính toán này được phối hợp hoạt động nhờ một tiến trình *server*. Có những khó khăn nhất định trong phân chia miền tính cho các tiến trình khác nhau. Nếu như song song hóa trên hệ chia sẻ nhớ, hình dạng miền tính cho mỗi tiến trình không đóng vai trò gì thì khi chuyển sang song song hóa trên hệ phân tán nhớ hình dạng miền tính cần được làm càng đơn giản càng tốt bởi các tiến trình cần trao đổi dữ liệu trên các vùng biên xung quanh miền tính. Cho nên chúng tôi thực hiện phương án song song hóa theo hàng của miền tính tức là chia miền tính thành các dải song song. Như vậy phần biên trao đổi giữa các miền tính đơn giản chỉ là một hàng của miền tính. Tất nhiên cách thực hiện này chỉ có được khi ta sử dụng lưới tam giác trên các nút lưới phân bố đều trên lưới chữ nhật:

```
for(int k=0;k<nThread;k++)
{   start[k] = k*(ny/nThread)*nx;
    end[k] = (k+1)*(ny/nThread)*nx-1;}
end[nThread-1] = n-1;
```

Trước hết tiến trình *server* chỉ đảm nhận vai trò nhập dữ liệu ban đầu cũng như dữ liệu biên và phân phối dữ liệu này tới các tiến trình *client*. Kết quả tính toán cuối cùng cũng được tập trung về tiến trình *client* với mục đích kết xuất hay hiển thị. Trong quá trình tính

toán tiến trình *server* sẽ đồng bộ hoạt động giữa các tiến trình *client*. Quá trình đồng bộ này bao gồm quá trình đồng bộ tính toán các thông lượng cho xác định đạo hàm và quá trình trao đổi dữ liệu cho quá trình lọc. Dưới đây là hoạt động của tiến trình *server*:

```
for(int it=0;it<nt;it++)
  { if (update)
  { setBoundary((it+1)*dt)
  int m = 0;
  for(int k=0;k<n;k++)
  { if (point[k] instanceof BoundaryPoint)
  { bc.pValue[m] = ((BoundaryPoint)point[k]).pValue;
  bc.qValue[m] = ((BoundaryPoint)point[k]).qValue;
  bc.hValue[m] = ((BoundaryPoint)point[k]).hValue;
  m++;}}
  for(int k=0;k<nThread;k++) toClient[k].write(bc);}
  int nProcess = 0;
  while(nProcess != nThread)
  { fromClient.read();
  nProcess++;}
  for(int k=0;k<nThread;k++) toClient[k].write("DERIVATIVE");
  if ((filter) && (it%filterStep == 0))
  { for(int i=0;i<npass;i++)
  { nProcess = 0;
  while(nProcess != nThread)
  { fromClient.read();
  nProcess++;}
  for(int k=0;k<nThread;k++) toClient[k].write("FILTER");}}}
```

Giống như quá trình thực hiện tính toán trên một máy, điều kiện biên được cập nhật trước tiên. Sau khi được cập nhật các giá trị trên biên được chuyển vào biến *bc* và gửi qua các kênh tin *toClient* tới mọi tiến trình *client*. Tiếp đó *server* thực hiện vai trò điều độ giữa các tiến trình trước khi thực hiện tính đạo hàm. Chỉ khi tất cả các tiến trình đã tính xong các thông lượng cần cho tính đạo hàm, đã trao đổi kết quả tính với nhau, *server* mới cho phép các tiến trình *client* tiếp tục tính toán thông qua lệnh *DERIVATIVE* tới các tiến trình *client*. Quá trình đồng bộ với chương trình lọc nhiễu các sóng ngắn phát sinh cũng được thực hiện tương tự. Riêng phần phân phối điều kiện ban đầu của *server* tới các *client* cũng như quá trình thu thập các kết quả tính toán phân tán từ *client* tới *server* không được nhắc đến ở đây.

Về cơ bản tính toán trên một tiến trình *client* cũng không có gì khác so với chương trình tính toán thực hiện trên một máy mà ta đã trình bày trong phần trên. Những khác biệt cơ bản trong mã nguồn chương trình chủ yếu nằm trong những đoạn mã cần thiết để trao đổi dữ liệu cũng như đồng bộ giữa các tiến trình

```
for(int it=0;it<ic.nt;it++)
{ if (ic.update)
{ bc = (BoundaryCondition)fromServer.read();
```

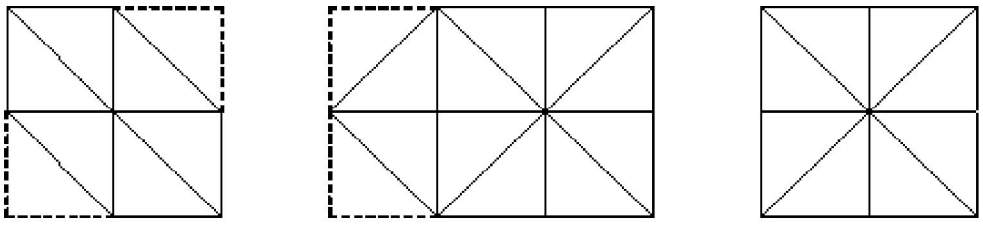
```
setBoundary();}
for(int k=ic.start;k<=ic.end;k++) point[k].computeFlux();
exchangeFlux();
toServer.write("READY");
fromServer.read();
for(int k=ic.start;k<=ic.end;k++) point[k].derivative();
for(int k=ic.start;k<=ic.end;k++) point[k].integrate(it);
if ((ic.filter) && (it%ic.filterStep == 0))
{ for(int i=0;i<ic.npass;i++)
{ exchangeFilter();
toServer.write("READY");
fromServer.read();
for(int k=ic.start;k<=ic.end;k++) point[k].filter(0.02);
for(int k=ic.start;k<=ic.end;k++) point[k].update();}}}
```

Ở đây *ic* là điều kiện ban đầu mà tiến trình *client* nhận được từ tiến trình *server* với các chỉ số *start* và *end* cho ta chỉ số đầu cuối miền tính của tiến trình này. Điều kiện biên nhận được từ *server* qua kênh tin *fromServer* được gán vào các biến nằm trên biên miền tính. Kết quả tính toán thông lượng của tiến trình *client* sau đó được trao đổi với các tiến trình *client* khác. Kết thúc quá trình tính toán này, mọi tiến trình *client* sẽ gửi thông điệp *READY* tới tiến trình *server* và đợi tín hiệu đồng bộ để tiếp tục tính đạo hàm và tích phân. Nếu thực hiện lọc ta cần thêm một quá trình trao đổi dữ liệu lọc và đồng bộ tương tự như trên.

3. KIỂM THỬ MÔ HÌNH

Chúng tôi chọn tất cả các điểm trong miền tính từ đỉnh của một lưới hình chữ nhật. Tuy vậy mọi xử lý tính toán đều được thực hiện trong khuôn khổ của một lưới tam giác phi cấu trúc. Hình 3 thể hiện một số chiến thuật sinh lưới tam giác từ lưới chữ nhật.

Phần kiểm thử sẽ sử dụng chiến thuật tám điểm kề và trong các hình vẽ, nghiệm giải tích nếu có được thể hiện bằng đường gạch đứt, nghiệm số bằng đường liền. Đường gạch đứt cũng được sử dụng cho độ cao địa hình trong một số trường hợp. Khi điều này xảy ra, chúng tôi có dẫn giải rõ ràng.



Hình 3. Ba chiến thuật sinh lưới tam giác từ lưới chữ nhật
6 điểm kề (trái), 4 và 8 điểm kề (giữa) và 8 điểm kề (phải)

3.1. Bài toán dòng chảy dừng dưới tới hạn qua địa hình không đều

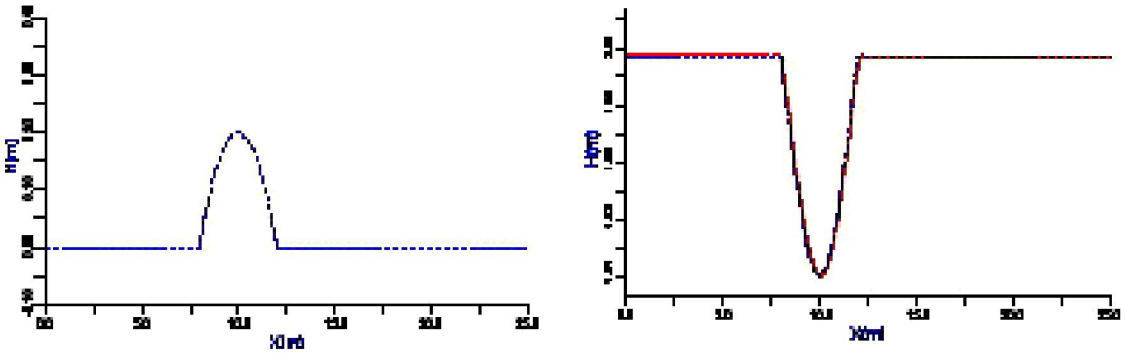
Trong phần này chúng tôi khảo sát dòng chảy một chiều qua địa hình biến đổi trên một kênh thẳng có độ rộng không đổi. Ma sát đáy được bỏ qua. Với bài toán dừng, lưu lượng không

biến đổi dọc theo kênh như có thể thấy từ phương trình liên tục. Phương trình bảo toàn động lượng cho ta định lý *Bernoulli*

$$\frac{1}{2}u^2 + g(h+z) = \frac{q^2}{2h^2} + g(h+z) = \text{const}$$

Như thế nếu cho trước lưu lượng tại biên thượng lưu và độ sâu tại biên hạ lưu, chúng ta hoàn toàn xác định được giá trị *const* trong công thức trên. Kết quả thực hiện nhận được từ sơ đồ tích phân *Adams-Bashforth* bậc ba:

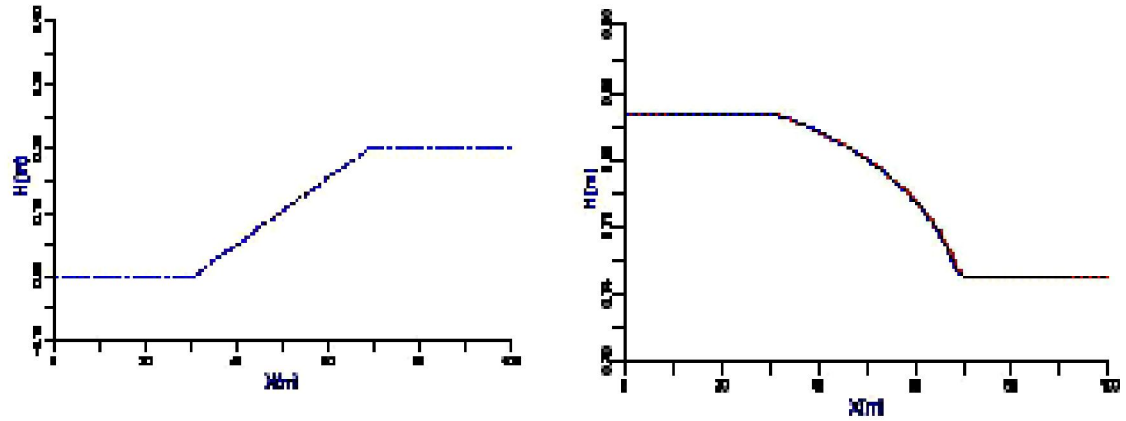
a- Dòng chảy dừng tới hạn qua điểm gồ: địa hình đáy có một điểm gồ (Hình 4 trái) được xác định bởi phương trình $z = 0.2 - 0.5(x-10)^2$



Hình 4. Địa hình đáy (trái) và độ sâu nhận được từ mô phỏng trong so sánh với nghiệm giải tích bài toán dòng chảy dừng tới hạn qua điểm gồ

Trong mô phỏng này độ dài kênh được lấy bằng $25m$, rộng $0.5m$ với độ phân giải $0.25m$. Lưu lượng và độ sâu cho trước lần lượt là $4.42m^3/s$ và $2m$. Lời giải nhận được tiến tới trạng thái dừng sau $50s$ với bước thời gian $0.02s$ và bộ lọc thực hiện 5 lần tại mỗi bước thời gian. Kết quả được cho trên Hình 4 (phải)

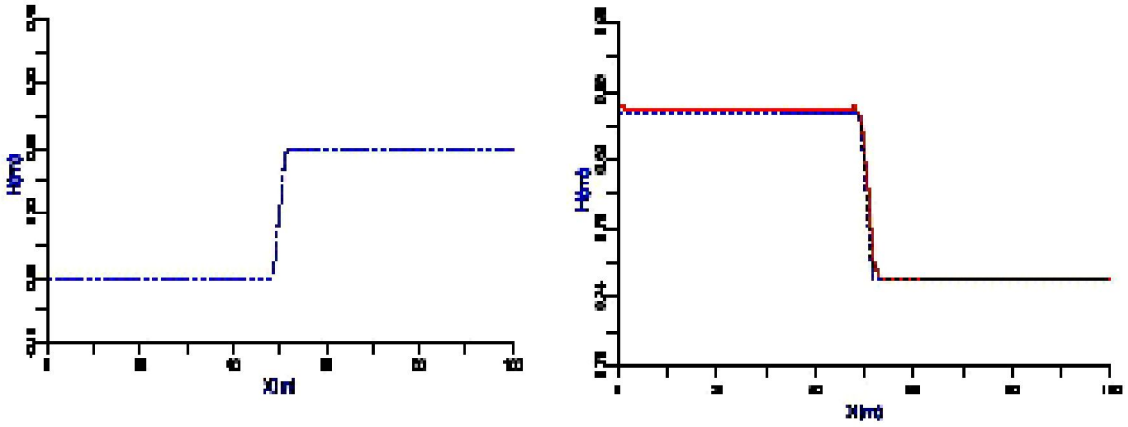
b- Dòng chảy dừng tới hạn qua địa hình dốc: bài toán được mô tả bởi [6]. Độ cao địa hình đáy được cho trên Hình 5 (trái)



Hình 5. Địa hình đáy (trái) và độ sâu nhận được từ mô phỏng trong so sánh với nghiệm giải tích bài toán dòng chảy dừng tới hạn qua địa hình dốc

Kênh thực hiện mô phỏng dài $100m$, rộng $2m$ với bước lưới $1m$. Lưu lượng tại biên trái là $1m^3/s$. Độ sâu tại biên phải là $0.5m$. Với bước thời gian $0.1s$, nghiệm số đạt tới trạng thái dừng khoảng sau $500s$. Bộ lọc vẫn được thực hiện giống như trong bài toán trước đó. Hình 5 cho ta kết quả mô phỏng bài toán này.

c- Dòng chảy dừng tới hạn qua địa hình gián đoạn: trong bài toán này mọi điều kiện biên và điều kiện ban đầu giống như trong bài toán trước đó. Điểm khác biệt là độ dốc địa hình bây giờ gần như 90^0 (Hình 6).



Hình 6. Địa hình đáy (trái) và độ sâu nhận được từ mô phỏng trong so sánh với nghiệm giải tích bài toán dòng chảy dừng tới hạn qua địa hình dốc

Mọi kết quả mô phỏng nhận được từ ba bài toán trên cho thấy sự trùng hợp rất tốt giữa nghiệm giải tích với nghiệm số. Điều này đảm bảo cho ta tin tưởng vào cách xử lý các số hạng *grdient* độ sâu cũng như các số hạng bình lưu của mô hình.

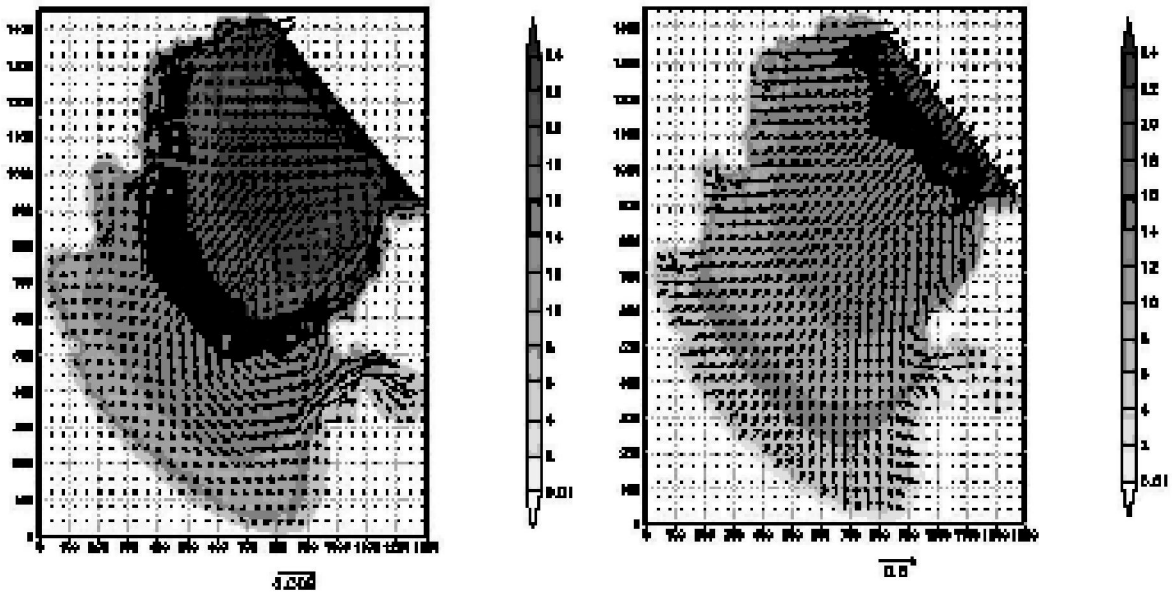
3.2. Bài toán sóng thủy triều

Trong kiểm thử này chúng tôi áp dụng mô hình đã thực hiện vào mô phỏng sóng triều tại các vùng cửa biển. Vấn đề đặt ra là ta phải dự báo mực nước cũng như dòng chảy tại các vùng ven biển với biên độ sóng triều cho trước tại các biên mở. Trong những vùng này không chỉ quy luật biến đổi của dòng chảy theo không gian và thời gian mà cả trạng thái khô ướt cũng trở nên quan trọng. Vùng được lựa chọn tính toán ở đây là Đà Nẵng.

Độ sâu trung bình qua đo đạc sẽ được nội suy về một lưới đều với bước lưới $100m$ cho ta điều kiện địa hình đáy của vùng khảo sát. Tại các biên mở sóng triều với biên độ $0.3m$, chu kỳ một ngày đêm và pha ban đầu bằng không được cho trước. Tại các vùng biên vuông góc với đường bờ chúng tôi sử dụng điều kiện biên phát xạ. Các vùng biên còn lại đều được xem là vùng biên đóng. Dòng chảy ban đầu được cho bằng không. Ta chỉ sử dụng nghiệm thu được sau khoảng thời gian tích phân một ngày đêm. Những kết quả nhận được trước khoảng này được xem là nghiệm tạm thời. Độ sâu ướt cực đại được lấy bằng $1cm$. Kết quả tích phân với hai chu kỳ triều cực đại triều dâng và triều rút cho Đà Nẵng được cho trên Hình 7.

Hình 7. Kết quả dự báo mực nước và dòng chảy tại vịnh Đà Nẵng khi triều dâng (trái) và khi triều rút (phải)

Nhìn chung về mặt định tính các kết quả thể hiện khá tốt song cần được thẩm định thông



qua các số liệu đo đạc thực tế. Tính toán sóng thủy triều tại vịnh Đà Nẵng với các thông số điều kiện biên và điều kiện ban đầu như trên trên máy tính với bộ vi xử lý 3GHz thời gian tích phân ba ngày là 5 giờ 30 phút với bước thời gian 1 giây. Để thử nghiệm chương trình tính toán sau khi đã được song song hóa chúng tôi đã thực hiện tính toán song song trên máy tính IBM chia sẻ nhớ với hai bộ vi xử lý tốc độ 3GHz. Kết quả với cùng thời gian tích phân ba ngày và bước thời gian 1 giây thời gian tính toán bây giờ rút xuống còn 4 giờ. Như vậy với hai bộ vi xử lý thời gian tính toán đã giảm được một phần tư so với khi chỉ tính toán với một bộ vi xử lý.

Theo logic thông thường với hai bộ vi xử lý, ta có thể hy vọng thời gian tính toán giảm xuống còn một nửa. Tuy nhiên thực tế ta chỉ giảm được một phần tư. Có điều này là do khi thực hiện tích phân tại mỗi bước thời gian bộ lọc cần được thực hiện một số lần nhằm loại bỏ các sóng nhiễu xuất hiện khi tích phân. Quá trình đồng bộ giữa các lần lọc này sẽ làm giảm đáng kể hiệu năng tính toán. Khi giảm số lần lọc đi, hiệu năng tính toán sẽ được tăng lên nhưng kết quả sẽ chứa các dao động nhiễu sinh ra, dẫn đến những sai lệch trong kết quả.

Nhằm tăng cường hơn nữa hiệu năng tính toán, chúng ta có thể thử nghiệm chương trình tính toán trên các máy tính chia sẻ nhớ với nhiều bộ vi xử lý hơn. Chắc chắn hiệu năng tính toán sẽ tăng lên đáng kể khi có nhiều bộ vi xử lý hơn. Tuy nhiên do các máy tính chia sẻ nhớ có giá thành khá cao và không phổ biến tại Việt Nam nên hướng song song hóa trên hệ chia sẻ nhớ không nhận được nhiều hỗ trợ như hướng song song hóa trên hệ phân tán nhớ. Tiếp cận theo hướng thứ hai này, bài toán kỹ thuật sẽ trở nên khá đơn giản khi ta có thể tận dụng tất cả các máy tính trong một mạng máy tính. Hiệu năng tính toán bây giờ sẽ phụ thuộc nhiều vào tốc độ truyền thông giữa các máy tính.

4. KẾT LUẬN

Các thử nghiệm cho thấy một sự tương thích giữa kết quả mô phỏng so với kết quả nhận

được từ nghiệm giải tích, từ đo đạc thực địa hay từ các mô hình khác. Ta có thể khẳng định hiệu năng của mô hình trong mô phỏng các bài toán gián đoạn. Khi lựa chọn sơ đồ bán ẩn mô hình nên sử dụng kỹ thuật sai phân đạo hàm theo hướng. Nếu sơ đồ tích phân là sơ đồ *Adams-Bashforth* bậc ba thì cả hai kỹ thuật sai phân không gian đều cho kết quả như nhau. Giữa hai phương pháp tích phân, sơ đồ *Adams-Bashforth* bậc ba cho kết quả chính xác hơn.

Hầu như sóng nhiễu đều xuất hiện trong kết quả với bất kỳ phương pháp tích phân nào. Điều này có nghĩa rằng chúng bắt nguồn từ kỹ thuật xấp xỉ đạo hàm không gian của ta. Do đó sơ đồ lọc loại bỏ các sóng ngắn này là cần thiết trong mô phỏng. Khi đó câu hỏi liệu các đại lượng bảo toàn có được bảo toàn hay không sau khi lọc sẽ được đặt ra. Có thể thấy vấn đề này không quá nghiêm trọng với các thử nghiệm đã thực hiện. Tuy nhiên ta cần nghiên cứu kỹ thêm trong tương lai. Tính bảo toàn cũng được đặt ra khi ta sử dụng hướng tiếp cận độ sâu ướt cực tiểu trong xử lý khô ướt. Với độ sâu này được lấy bằng $10^{-6}m$ có thể xem tác động của nó tới tính bảo toàn là nhỏ.

Thử nghiệm song song hóa chương trình áp dụng vào một bài toán cụ thể xác định sóng triều tại vịnh Đà Nẵng cho thấy hiệu năng tính toán đã được tăng lên. Tuy nhiên khi chuyển từ tính toán tuần tự sang tính toán song song (áp dụng cho hai máy), thời gian tính toán chỉ rút ngắn đi một phần tư. Với số bộ vi xử lý tăng lên, hiệu năng tính toán sẽ cao hơn. Hướng song song hóa trên hệ phân tán nhớ có ưu điểm lớn về mặt kỹ thuật và cần được tiến hành thử nghiệm trong tương lai.

TÀI LIỆU THAM KHẢO

[1] F Alcrudo, PG Navarro, A high-resolution Godunov-type scheme in finite volumes for the 2D shallow water equations, *International Journal for Numerical Methods in Fluids* **16** (1993) 489–505.

[2] F Alcrudo, PG Navarro, Computing two dimensional flood propagation with a high resolution extension of McCormack's method, *Proceedings on Modelling of Flood Propagation over Initially Dry Areas*, American Society of Civil Engineers, Milan, Italy, 1994 (3–17).

[3] K Anastasiou, CT Chan, Solution of the 2D shallow water equations using the finite volume method on unstructured triangular meshes, *International Journal for Numerical Methods in Fluids* **24** (1997) 1225–1245.

[4] RJ Fennema, MH Chaudhry, Explicit methods for 2D transient free surface flows, American Society of Civil Engineers, *Journal of Hydraulic Engineering* **116** (1990) 1013–1034.

[5] C.A.R. Hoare, *Communicating sequential processes*, Prentice Hall International, 1985.

[6] M. Horritt, Development and testing of a simple 2D finite volume model of sub-critical shallow water flow, *International Journal for Numerical Methods in Fluids* **44** (2004) 1231–1255.

[7] K. Hu, CG Mingham, DM Causon, Numerical simulation of wave overtopping of coastal structures using the non-linear shallow water equations, *Coastal Engineering* **41** (2000) 433–465.

[8] PA Madsen, OR Sorensen, HA Schoffer, Surf zone dynamics simulated by a Boussinesq type model. Part I: Model description and cross-shore motion of regular waves, *Coastal Engineering* (32) (1997) 255–287.

[9] KD Nguyen, "2D shallow water models using unstructured finite volumes methods", University of Caen, France, 2002.

[10] Tran GL, Nguyen MS, Le VC, Calculation of the horizontal two dimensional unsteady flows by the method of characteristics, *Vietnam Journal of Mechanics* **25** (2003) 49–64.

[11] P Welch, "Process oriented design for Java - Concurrency for all", University of Kent at Canterbury, England. 2002.

[12] P Welch, J Aldous, J. Foster, "CSP networking for Java (JCSP.net)", University of Kent at Canterbury, England. 2002.

Nhận bài ngày 17 - 8 - 2006