TẠP CHÍ KHOA HỌC TÍNH TOÁN VÀ ĐIỀU KHIỂN Tập II (1986), Số 1 (27–30)

MỘT SỐ THUẬT TOÁN VÀ CHƯƠNG TRÌNH GIẢI HỆ PHƯƠNG TRÌNH ĐẠI SỐ TUYẾN TÍNH VỚI MA TRẬN THƯA CỠ LỚN

NGUYỄN CÔNG ĐIỀU
HOÀNG VĂN LAI
NGUYỄN LÊ THU

I – ĐẶT VẤN ĐỀ

Trong quá trình giải các bài toán vật lý bằng các phương pháp số (sai phân, phần tử hữu hạn, v.v...) chúng ta thường dẫn đến một hệ phương trình đại số tuyến tính

$$Ax = b \tag{1}$$

Với A là ma trận cỡ lớn. Trong thực tế cỡ của A có thể đạt tới hàng nghìn hoặc hàng vạn. Đối với các ma trận có cỡ như vậy, các thuật toán cổ điển không còn thích hợp nữa vì nó đòi hỏi quá nhiều bộ nhớ của MTĐT và thời gian để thực hiện thuật toán.

Từ những năm 1970 trở lại đây, lý thuyết giải hệ phương trình đại số tuyến tính thưa cỡ lớn có bước phát triển mới. Trên cơ sở các thuật toán sẵn có, người ta đã cải biên đi để triệt để lợi dụng các tính chất thưa của ma trận. Trong bài này chúng tôi mô tả một số phương pháp lưu trữ ma trận A sao cho tiết kiệm bộ nhớ và giới thiệu một số thuật toán giải chúng. Sau cùng sẽ giới thiệu bộ chương trình đã được thiết lập theo các thuật toán đó theo nguyên tắc đơn thể (module).

II – CÁCH LƯU TRỮ MA TRẬN THƯA

Giả sử A là ma trận vuông cỡ n×n. Khái niệm thưa là một khái niệm tương đối. Thông thường có thể hiểu được khi số lượng phần tử khác không khá nhỏ so với số lượng phần tử của ma trận A. Do đó nếu chúng ta chỉ cần lưu trữ các phần tử khác không của A thì có thể sẽ tiết kiệm được khá nhiều ô nhớ của máy tính. Phương pháp đầu tiên ta mô tả là phương pháp lưu trữ động.

Phương pháp lưu trữ động lần đầu tiên được Tenning và Tuff [1] đề ra, sau đó được nhiều tác giả khác sử dụng. Nó thuận tiện khi ma trận có dạng băng hay các phần tử khác không nằm tập trung gần đường chéo chính.

Nếu ta ký hiệu

$$\lambda(i) = \min \{ j : 1 \leqslant j \leqslant i \leqslant n ; \quad a_{ij} \neq 0 \} \tag{2}$$

$$\beta(i) = \min \{ i : 1 \leqslant i \leqslant j \leqslant n ; \quad a_{ij} \neq 0 \} \tag{3}$$

Với ma trận A bất kỳ ta có phân tích

$$A = L + U$$

trong đó L là phần tam giác dưới, còn U là phần tam giác trên của A. Việc lưu trữ dưới dạng profil được tiến hành riêng biệt đối với L và U (nếu A là đối xứng chỉ cần lưu trữ L là đủ).

Để tiến hành lưu trữ L ta cần hai mảng một chiều. Mảng đầu chứa các phần tử của A theo từng hàng bắt đầu từ phần tử đầu tiên khác không của hàng đó $a_{i,\ \lambda(i)}$ cho đến phần tử đường chéo a_{ii}. (Đối với ma trận U ta lưu trữ theo từng cột bắt đầu từ $a_{\beta(j),\ j}$ đến a_{jj}) mảng thứ hai nhằm chỉ ra vị trí của các phần tử đường chéo nằm trong mảng thứ nhất.

Phương pháp lưu trữ thứ hai là phương pháp động. Trong phương pháp này chỉ lưu trữ các phần tử khác không. Nó cần ba mảng một chiều. Mảng đầu để chứa các phần tử khác không của A theo thứ tự bất kỳ. Mảng thứ hai và ba để chỉ vị trí hàng và cột của phần tử khác không ở mảng đầu tiên. Như vậy để lưu trữ toàn bộ ma trận A ta cần 3×M ô nhớ, trong đó M là số lượng khác không của A.

III — CÁC THUẬT TOÁN.

Các thuật toán để giải hệ (1) với các ma trận thưa bao gồm hai loại: loại thuật toán lặp và thuật toán chính xác. Dưới đây chúng tôi chỉ đưa ra một số thuật toán lặp. Các thuật toán này tuy không cho lời giải chính xác nhưng có một loạt ưu điểm: lợi dụng tốt tính thưa của ma trận và đơn giản vì các phép tính chỉ bao gồm hai loại: nhân véctơ với ma trận và nhân vô hướng hai véc tơ. Điều này cho phép giảm số lượng phép tính đến tối thiểu vì các phép tính chỉ thực hiện với các phần tử khác không của ma trận. Điều này hoàn toàn thích hợp với cách lưu trữ động. Các ma trận A dưới đây thường phải được giả thiết là đối xứng và xác định dương.

1. Phương pháp Gradient liên hợp.

Thuật toán này được đề ra từ những năm 50 do Hestens và Stifel nêu lên lần đầu. Áp dụng cho hệ (1) chúng được viết

$\bar{x}_0$ cho trước (thông thường cho $\bar{x}_0 = 0$)

$\bar{r}_0 = \bar{v}_0 = \bar{b} - A\bar{x}_0$

$k = 0$

$M_k = (\bar{r}_k / \bar{r}_k) / (A\bar{v}_k / \bar{v}_k)$

$\bar{x}_{k+1} = \bar{x}_k + M_k \bar{v}_k$

$\bar{r}_{k+1} = \bar{r}_k - M_k A\bar{v}_k$

$L_{k+1} = (\bar{r}_{k+1}, \bar{r}_{k+1}), (\bar{r}_k / \bar{r}_k)$

$\bar{v}_{k+1} = \bar{r}_{k+1} + L_{k+1} \bar{v}_k$

nếu $\| \bar{r}_{k+1} \| > \varepsilon \| \bar{1} \|$ thì k=k+1 và chuyển lên (i)

Stop.

Thật ra đây là thuật toán đúng cho nghiệm chính xác sau nhiều nhất là n lần lặp. Tuy nhiên do sai số tính toán, nó trở thành phương pháp lặp và hội tụ sau số bước lặp lớn hơn n nhiều.

Nếu ký hiệu $\chi(A) = \| A \| \| A^{-1} \|$ là số điều kiện của ma trận A. Người ta có nhận xét rằng nếu $\chi(A) \approx 1$ thì phương pháp hội tụ rất nhanh, sau khoảng n/3 bước lặp [1].

2, Phương pháp Gradient liên hợp có chỉnh lý trước.

Từ lúc xuất hiện phương pháp gradient liên hợp đã thu hút sự chú ý của nhiều người vì nó khá đơn giản. Nhưng dần dần nó bị lãng quên đi vì áp dụng trong thực tế nó hội tụ rất chậm. Nguyên nhân như ta thấy ở phần trên là do không phải đối với ma trận nào $\chi(A)$ cũng xấp xỉ gần bằng 1. Muốn áp dụng phương pháp này có hiệu quả, cần phải cải tiến nó, chỉnh lý ma trận A sao cho số điều kiện của nó gần 1. Mãi cho đến năm 1977 mới có kết quả đầu tiên theo hướng này [2]. Nhưng khi phương pháp được đề ra, nó liền được áp dụng rất rộng rãi, nhất là trong phương pháp phần tử hữu hạn [3] và còn dùng để giải các bài toán phi tuyến [4]. Ngày nay người ta càng có thêm những cải biến mới. Chúng tôi trình bày dưới đây hai cách chỉnh lý.

Thay vì giải (1) ta dẫn đến hệ tương đương

$$K^{-1} A\bar{x} = K^{-1} b \qquad (4)$$

Nếu K^{-1} gần A^{-1} theo nghĩa nào đó thì rõ ràng $\chi(K^{-1} A) \approx 1$. Vấn đề ở đây là chọn K sao cho (i) có profil giống A và (ii) dễ nghịch đảo.

Có hai phương pháp lựa chọn

– Chọn theo phương pháp choleski không hoàn chỉnh; ta cần tìm K dưới dạng

$$K = \tilde{L}\,\tilde{L}^T \qquad (5)$$

trong đó

$$\tilde{l}_{ij} = \begin{cases} a_{ij} - \sum_{k=1}^{j-1} \tilde{l}_{ik}\,\tilde{l}_{jk} / \tilde{l}_{jj} & \text{nếu } a_{ij} \neq 0 \\ 0 & \text{nếu } a_{ij} = 0 \end{cases}$$

$$\tilde{l}_{ii} = \left(a_{ii} - \sum_{k=1}^{i-1} \tilde{l}_{ik}^2 \right)^{\frac{1}{2}}$$

Rõ ràng là K thỏa mãn hai điều kiện (i) và (ii). Qua công thức trên cho thấy K biểu diễn gần như phân tích Choleski của A. Cách thứ hai là theo phương pháp giảm dư trên. Ta biểu diễn

$$A = E + D + F$$

E là phần tam giác trên, F là phần tam giác dưới, D là phần đường chéo của ma trận A.

Khi đó ta chọn $\tilde{L}$ theo công thức sau:

$$K = \tilde{L}\,\tilde{L}^T = (D + \omega E)\, D^{-1} (D + \omega F)$$
$$= (D^{1/2} + \omega D^{-1/2} F)\,(D^{1/2} + \omega E D^{-1/2}) \qquad (6)$$

ω là thông số giảm dư và $\omega \in [0,1]$. Chưa có cách chỉ ra ω tối ưu sao cho phương pháp hội tụ nhanh nhất. Các kết quả thí nghiệm số của chúng tôi cho phép chỉ ra rằng $\omega \in [0{,}8 \div 1{,}2]$ là tốt nhất. Nó cũng trùng với kết quả của [1].

Khi đó thuật toán Gradient liên hợp có chỉnh lý trước được viết:

$\bar{x}_0 = 0$

$\bar{r}_0 = \bar{b}$

$\bar{Z}_0 = \bar{P}_0 = K^{-1} \bar{r}_0$

$k = 0$

$M_k = (\bar{Z}_k, \bar{Z}_k)/(K^{-1} A \bar{P}_k, \bar{P}_k)$

$\bar{x}_{k+1} = \bar{x}_k + M_k \bar{P}_k$

$\bar{r}_{k+1} = \bar{r}_k - M_k A \bar{P}_k$

$\bar{Z}_k = K^{-1} \bar{r}_{k+1}$ (7)

$L_{k+1} = (\bar{Z}_{k+1}, \bar{Z}_{k+1}) / (\bar{Z}_k, \bar{Z}_k)$

$\bar{P}_{k+1} = \bar{Z}_{k+1} + L_{k+1} \bar{P}_k$

Nếu $\|\bar{r}_{k+1}\| \geqslant \varepsilon \|b\|$ thì $k = k + 1$ chuyển lên (i)

Stop

Trong thuật toán trên có hai lần phải giải hệ $K.\bar{Z} = \bar{r}$, nhưng điều này dễ thực hiện vì $K = \tilde{L}\,\tilde{L}^T$.

3. Phương pháp lặp Richarson – Chebysev

Đây là thuật toán đơn giản và có tốc độ hội tụ nhanh nhất trong lớp giải lặp hai lớp. Giả sử ma trận A thỏa mãn điều kiện

$$0 < m_1 E \leqslant A \leqslant m_2 E \qquad (8)$$

m_1, m_2 có thể coi như giới hạn phổ trên và dưới của A. Khi đó quá trình lặp được xây dựng như sau [5]

$$\bar{x}_{k+1} = \bar{x}_k - T_k (A \bar{x}_k - b) \qquad (9)$$

T_K là tham số lặp được tính theo công thức

$$T_k = T_0 / (1 + \rho_0 M_k) \qquad (10)$$

$$T_0 = \frac{2}{m_1 + m_2}; \quad \rho_0 = \frac{1 - \xi}{1 + \xi}; \quad \xi = \frac{m_1}{m_2}$$

M_K là nghiệm của đa thức Chebysev bậc n.

Thứ tự chọn nghiệm của đa thức Chebysev có ảnh hưởng quan trọng đến sự ổn định của thuật toán. Khi đó M_k phải theo công thức

$$M_k = -\cos \frac{2\theta_k - 1}{2n}$$

Trong đó $\{\theta_k\}_{k=1}^{n}$ là thứ tự chọn nghiệm. Thuật toán tìm θ_k được trình bày trong [5].

Ngoài ra trong công thức còn cần phải tìm m_1 và m_2. Thuật toán này có thể tìm thấy trong [6].

Ngoài các phương pháp lặp đã trình bày ở trên, còn một lớp phương pháp giải đúng mà chúng tôi sẽ trình bày tiếp ở bài sau.

IV – BỘ CHƯƠNG TRÌNH

Dựa trên các thuật toán trên, chúng tôi xây dựng một bộ chương trình nhằm mục đích có thể giải hiệu quả được các phương trình đại số tuyến tính với ma trận thưa cỡ lớn. Các chương trình này được viết dưới dạng đơn thể (module) bằng ngôn ngữ FORTRAN.IV các module này có thể dùng độc lập với nhau, hoặc có thể dùng chung trong các chương trình mẫu khác. Các thông số vào hay ra có cùng bản chất được mô tả bằng các tham số giống nhau. Trong từng trường hợp cụ thể của bài toán cần lựa chọn những chương trình (module) sao cho hiệu quả có thể lớn nhất. Dưới đây là các module chính.

– CPROF – chương trình giải bằng phương pháp Choleski. Ma trận được lưu trữ dưới dạng profil.

– GNSP 1 Giải bằng phương pháp Gauss, khi A đối xứng về vị trí, không đối xứng về giá trị.

– GCMNS giải bằng phương pháp Gradient liên hợp khi A không đối xứng.

– GCMSP phương pháp Gradient liên hợp khi A đối xứng.

– SOSILT phương pháp kicharson – Chebysev.

– GRALH phương pháp Gradient liên hợp có chỉnh lý trước bằng phương pháp Choleski không hoàn chỉnh.

– PCGSOR phương pháp Gradient liên hợp có chỉnh lý trước bằng phương pháp giảm dư trên.

V – KẾT LUẬN

Các chương trình trên đã được thử nghiệm trên máy ODRA 1304, MINSK 32 và IBM 360140. Chúng cũng được cài đặt trên máy vi tính MICRAN. Các thí dụ đều cho kết quả thỏa đáng. Đối với phương pháp Gradient liên hợp khi n nhỏ cho kết quả rất tốt, nhưng khi n lớn nó hội tụ rất chậm. Nhưng khi dùng phương pháp Gradient liên hợp có chỉnh lý cho kết quả khả quan. Thông thường chỉ cần 5,6 lần lặp là cho kết quả thỏa đáng. Trong hai chương trình GRALH và PCGSOR thì GRALM tỏ ưu thế rõ rệt bởi nó cho kết quả tốt sau số lần lặp ít hơn và phương pháp tỏ ra không nhạy cảm với việc cho giả trị lặp đầu tiên. Các phương pháp giải đúng cho kết quả phù hợp.

TÀI LIỆU THAM KHẢO

1. Peronnet A., Cours D.E.A. 1981 – 1982.

2. Meijerink Vandevorst, An iterative solution method for linear system of which the coefficient matrix is a symmetric M – matrix, Mathematis of computation. V. 31, N° 1371, 1977.

3. R.S. Chwarez, The method of conjugate gradients in Finite element application Z.A.M.P. V. 30, F. 2, 1979.

4. Glowinski et all, An efficient preconditionned conjugate gradient method applied to nonlinear problems in fluid dynamics. IRIA 1979.

5. Samarski Nicolaev, Phương pháp giải phương trình lưới (tiếng Nga) M. 1978.

6. Marchuk G.I., Phương pháp số (tiếng Nga), M. 1980.

ABSTRACT

SOME ALGORITHMS AND A PACKAGE FOR LARGE SPARSE MATRICES

In this paper some direct and iterative processes used in linear algebra for large sparse matrices will be considered. Most of out attention will be concentrated on storage schemes to economise the memory of a computer. The algorithms will be modified that they can use will sparse structure of matrices Finally the package will be exposed and test of this package give the good results.