ĐẠI HỌC QUỐC GIA HÀ NỘI
TRƯỜNG ĐẠI HỌC KHOA HỌC TỰ NHIÊN

TIỂU LUẬN

Phương pháp số cho đại số tuyến tính
Một số phương pháp lặp giải hệ phương trình tuyến tính

Đại học Quốc gia Hà Nội
Đại học Khoa học Tự nhiên
Khoa Toán cơ tin

Giảng viên: Phan Trung Hiếu

Học viên: Nguyễn Mạnh Linh

Mục lục

Tóm tắt

Tiểu luận này trình bày một số phương pháp lặp giải hệ phương trình tuyến tính lớn bằng cách sử dụng kĩ thuật tối thiểu hóa phần dư tại mỗi bước lặp (minimal residual (MR)). Ma trận của hệ tuyến tính được xem xét cả dạng tổng quát (sử dụng phương pháp GMRES (Generalized minimal residual method)) và dạng đối xứng xác định dương (sử dụng phương pháp MINRES (Minimal residual method)). Không gian con Krylov được xem xét để tìm nghiệm gần đúng của hệ. Bài này cũng giới thiệu về phương pháp lặp Arnoldi và Lanczos. Cuối cùng ta sử dụng ngôn ngữ lập trình Python để minh họa các thuật giải và so sánh hiệu năng của chúng.

1 Phương pháp lặp đơn giản

Ma trận M hệ phương trình tuyến tính $Ax = b$, ta cùng tìm một ý tưởng tự nhiên để giải gần đúng nghiệm của phương trình này. Ma trận M được chọn sao cho $M^{-1}A$ theo một nghĩa nào đó gần đúng với ma trận đơn vị. $M^{-1}(b - Ax_k)$ có thể được dùng làm xấp xỉ lỗi $A^{-1}b - x_k$ với nghiệm gần đúng x_k

2 Tổng quan

Xem xét hệ phương trình tuyến tính $Ax = b$ trong đó $A \in \mathbb{C}^{m \times m}$ và $b \in \mathbb{C}^m$.

Nghiệm chính xác của hệ $x_* = A^{-1}b$. Dương nhiên việc tính ma trận nghịch đảo A^{-1} hoặc dùng phương pháp khử Gauss là rất tốn chi phí. Ta biết rằng độ phức tạp của thuật toán khi sử dụng phương pháp khủ Gauss là $O(m^3)$. Để có cái nhìn trực quan về độ phức tạp này, ta có thể xem qua bảng sau

Năm	m	m^3
1950	20	2×10^3
1965	200	2×10^6
1980	2000	2×10^9

Khi kích thước của ma trận tăng lên, số phép tính cũng tăng lên rất nhanh. Cùng với sự phát triển của phần cứng chúng ta cũng tính toán được với những ma trận với kích thước lớn hơn rất nhiều so với những năm 1950. Nhưng ta cũng thấy rằng, khi kích thước của ma trận là hàng nghìn thì số phép tính đã là hàng tỉ. Với những bài toán lớn như mô phỏng thời tiết hay thiên hà trong thiên văn học, phần cứng máy tính không thể đuổi kịp để tính toán cũng như lưu trữ.

Điều này đòi hỏi chúng ta cần tìm ra những phương pháp tốt hơn nhằm giảm độ phức tạp của thuật toán. Tiểu luận này giới thiệu 1 vài phương pháp với độ phức tạp $O(m^2)$.

Phương pháp lặp cho phép chúng ta tiến gần đến nghiệm chính xác của phương trình qua mỗi bước lặp. Đến một lúc nào đó phần dư là đủ chấp nhận được, ta có nghiệm gần đúng của phương trình.

Gọi x_n là nghiệm gần đúng của phương trình tại bước lặp thứ n. Ta định nghĩa phần dư

$$r_n = b - Ax_n \tag{2.1}$$

Khi $r_n < \epsilon$ (là một sai số nhỏ chấp nhận được) ta dừng lại và thu được nghiệm gần đúng x_n

Trong bài này chúng ta sẽ tìm nghiệm $x_n \in \kappa_n$, trong đó κ_n là không gian con Krylov

Không gian con Krylov

Cho ma trận $A \in \mathbb{C}^{m \times m}$ và vector $b \in \mathbb{C}^m$.

Dãy Krylov được định nghĩa là tập các vectors $b, Ab, A^2b, ...$

Không gian con Krylov là không gian được sinh bởi các vectors này

Ma trận Krylov

$$K_n = \begin{bmatrix} b & | & Ab & | & A^2b & ... & | & A^{n-1}b \end{bmatrix} \tag{2.2}$$

Trở lại bài toán, ta muốn tìm nghiệm x_n trong không gian κ_n hay nói các khác $x_n \in range(K_n)$. Ta có thể sử dụng phân tích QR cho ma trận Krylov và đặt $x_n = Q_n y$ trong đó Q_n là ma trận unitary.

Bài toán cực tiểu hóa r_n đưa về việc cực tiểu hóa $\|AQ_n y - b\|$

Chúng ta sẽ tiếp cận bài toán với việc giải tổng quát với phương pháp GMRES và sử giải bài toán ma trận đối xứng như một trường hợp riêng với phương pháp MINRES. Trước hết, thuật giải Arnoldi sẽ được giới thiệu sau đây.

3 GMRES

3.1 Thuật giải Arnoldi

Phương pháp GMRES sử dụng quá trình Gram-Schmitz chỉnh sửa để thiết lập một cơ sở trực chuẩn cho không gian Krylov $span\{r_0, Ar_0, ..., A^k r_0\}$. Quá trình Gram-Schmitz được áp dụng cho không gian này được gọi là phương pháp *Arnoldi* Giải thuật này nhằm phân tích $A = QHQ^*$ hoặc $AQ = QH$, trong đó Q là ma trận trực chuẩn (unitary) còn H là ma trận Hessenberg[1].

Gọi Q_n (ma trận $m \times n$) là ma trận được tạo thành bởi n cột đầu tiên của ma trận Q

$$Q_n = \begin{bmatrix} q_1 & | & q_2 & | & q_3 & ... & | & q_n \end{bmatrix} \tag{3.1}$$

Gọi $\tilde{H}_n$ là phần trên bên trái $(n+1) \times n$ của ma trận H. $\tilde{H}_n$ đương nhiên cũng là 1 ma trận Hessenberg.

Ta có phương trình:

$$AQ_n = Q_{n+1}\tilde{H}_n \tag{3.2}$$

Ta có thể viết riêng cho cột thứ n của phương trình như sau:

$$Aq_n = h_{1n}q_1 + ... + h_{nn}q_n + h_{n+1,n}q_{n+1} \tag{3.3}$$

Có nghĩa là q_{n+1} có thể tìm được từ n vectors $q_1, q_2, ..., q_n$. q_{n+1} được tính theo phương pháp lặp bằng giải thuất dưới đây (thuật giả Arnoldi)

Arnoldi Algorithm

```
Given q_1 with ||q_1|| = 1
For j = 1, 2, ...
    q̃_{j+1} = A q_j
    For i = 1, ..., j
        h_ij = ⟨q̃_{j+1}, q_j⟩
        q̃_{j+1} ← q̃_{j+1} − h_ij q_i
    h_{j+1,j} = ||q̃_{j+1}||
    q_{j+1} = q̃_{j+1} / h_{j+1,j}
```

[1] *Ma trận H vuông kích thước $n \times n$ được gọi là ma trận upper Hessenberg nếu $h_{ij} = 0$ với mọi i, j mà $i > j + 1$*

Trong đó $\langle v, u \rangle$ là tích vô hướng của 2 vectors v và u.

Chúng ta có thể tìm được ma trận Q với các ngôn ngữ lập trình bậc cao như MATLAB một cách tương đối dễ đàng. Ma trận A ở đây chỉ xuất hiện trong tích Aq_j và MATLAB cũng tích hợp sẵn chương trình nhân ma trận. Tuy nhiên trong bài này, tác giả sử dụng ngôn ngữ lập trình pyhthon cũng là một ngôn ngữ bậc cao hiện đại và có nhiều bộ thư viện giúp ta tính toán với ma trận tương đối dễ dàng và tường minh như *numpy* chẳng hạn.