Journal of Natural Sciences
Pai Chai University, Korea
Vol. 16, No. 1 : 49-54, 2005

역거듭제곱방법의 비교

이 규 봉
배재대학교 자연과학대학 전산정보수학과

Comparison of Two Algorithms of Inverse Power Method

Gyou-Bong Lee
Department of Applied Mathematics, Pachai University

요 약

행렬의 고유치를 계산하는 방법을 가르칠 때 역거듭제곱방법을 소개한다. 이 방법의 알고리즘을 이론적인 면에서 이해하기 쉬운 알고리즘과 계산적인 면에서 효과적인 알고리즘을 비교한다.

Abstract

When teaching on the computing of eigenvalues of a matrix, we introduced the inverse power method. We compared the algorithm which is good for theoretical point of view with that is useful for computational point of view.

Keywords: eigenvalues, eigenvectors, power method, inverse power method

I. 서 론

고유치를 계산하는데 가장 알기 쉽고 고전적인 방법은 거듭제곱방법이다. 이 방법은 0이 아

Corresponding Author : Gyou-Bong Lee, Department of Applied Mathematics, PaiChai University, Daejeon, 302-735, Korea. Tel. : +82-42-520-5608, E-mail : gblee@pcu.ac.kr

닌 임의의 초기벡터에 주어진 행렬을 거듭해서 곱해주면 그 벡터는 그 행렬의 고유치 중 절대값이 가장 큰 고유치에 대응하는 고유벡터로 수렴한다는 이론에 근거한 것이다. 그러므로 이 수렴하는 벡터를 이용하여 주어진 행렬의 고유치 중 절대값이 가장 큰 고유치를 구할 수 있다.

역거듭제곱방법은 거듭제곱방법을 응용한 방법으로, 그 근거는 주어진 행렬을 이 행렬의 고유치가 아닌 상수만큼 이동하고 역행렬을 취하면 이동된 역행렬의 고유벡터는 원래 주어진 행렬의 고유벡터와 같다는 사실이다. 따라서 이 역행렬에 거듭제곱방법을 이용하면 주어진 상수에 가장 가까운 고유치를 구할 수 있다.

이 논문에서는 이동한 행렬의 역행렬의 고유치와 원래의 행렬의 고유치를 비교하여 구하는 방법과 이동된 역행렬의 고유벡터는 원래 주어진 행렬의 고유벡터와 같다는 사실에 의하여 고유치를 구하는 방법에 대하여 가르칠 때와 실지 계산할 때의 장단점을 비교하고자 한다.

II장에서 역거듭제곱방법에 대한 두 가지 알고리즘을 설명하고 III장에서는 계산결과를 제시하며 IV장에서 결과를 비교한다.

Ⅱ. 알고리즘의 비교

행렬 A가 다음과 같은 고유치를 갖고 그에 대응하는 고유벡터 $u_1, u_2, \cdots, u_n$이 일차독립이라고 하자.

$$|\lambda_1| > |\lambda_2| \geq \cdots \geq |\lambda_n|$$

임의의 초기벡터 x_0에 대하여 $x_k = A^k x_0$, $k = 1,2,\cdots$이라고 하면 x_k는 λ_1에 대응하는 고유벡터에 수렴한다. 따라서 $\beta_k = \frac{x_k^T A x_k}{x_k^T x_k}$는 λ_1에 수렴한다. 만일 $x_k = \frac{A^k x_0}{||A^k x_0||}$로 정규화시키면 $\beta_k = x_k^T A x_k$는 λ_1에 수렴한다. 이 방법을 거듭제곱방법이라고 한다. 이 방법은 $|\lambda_1|$과 $|\lambda_2|$의 차이가 클수록, 즉 $|\frac{\lambda_2}{\lambda_1}|$이 작을수록 더욱 빨리 수렴한다. 이 방법의 알고리즘은 다음과 같이 두 가지로 나타낼 수 있다.

알고리즘 1.1	알고리즘 1.2
임의의 벡터 $x^{(0)}$	임의의 벡터 $x^{(0)}$

for k=1,2,...
$y = Ax_{k-1}$
$\beta_k = x_{k-1}^T y$
$x_k = y/||y||$
end

for k=1,2,...
$y = Ax_{k-1}$
$x_k = y/||y||$
$\beta_k = x_k^T Ax_k$
end

알고리즘 1.1과 1.2에서 x_k는 서로 같다. 따라서 알고리즘 1.1의 β_k는 x_{k-1}에 대응하는 근사치이고 알고리즘 1.2의 β_k는 x_k에 대응하는 근사치로서 결국 같은 값으로 수렴한다. 그러나 연산의 수에 있어서는 매 단계에서 알고리즘 1.1은 하나의 행렬과 벡터의 곱이 있으나, 알고리즘 1.2에서는 두 개의 행렬과 벡터의 곱이 있다. 그러므로 알고리즘 1.2는 알고리즘 1.1이 필요한 연산의 거의 두 배가 더 필요하다. 그러므로 계산적인 면에서는 알고리즘 1.1이 더 효과적임을 알 수 있다.

상수 α에 대하여 $Ax = \lambda x$이면 $(A-\alpha I)x = (\lambda-\alpha)x$가 된다. 즉, A의 고유치가 λ이면 행렬 $A-\alpha I$의 고유치는 $\lambda-\alpha$가 되고 대응하는 고유벡터는 λ에 대응하는 A의 고유벡터와 같다. 따라서 이동된 행렬 $B = A-\alpha I$에 거듭제곱방법을 응용하면 $x_k = \frac{B^k x_0}{||B^k x_0||}$는 A의 고유벡터 u_J로 수렴한다. 여기서 $|\lambda_J - \alpha| = \max_i |\lambda_i - \alpha|$이다.

α가 A의 고유치가 아니면 행렬 $A-\alpha I$는 0인 고유치가 없으므로 정칙이고 역행렬이 존재한다. 따라서

$$(A-\alpha I)^{-1}x = \frac{1}{\lambda-\alpha}x$$

이므로 역행렬 $(A-\alpha I)^{-1}$의 고유치는 $(\lambda-\alpha)^{-1}$가 되고 대응하는 고유벡터는 λ에 대응하는 A의 고유벡터와 같다. 역으로 μ가 $(A-\alpha I)^{-1}$의 고유치이면

$$(A-\alpha I)^{-1}u = \mu u$$

인 벡터 $u \neq 0$가 있고 이 방정식은 $Au = (\frac{1}{\mu}+\alpha)u$와 동치이므로 $\frac{1}{\mu}+\alpha$는 행렬 A의 고유치

가 된다. 그러므로 μ가 $(A-\alpha I)^{-1}$의 고유치이고 λ가 A의 고유치이면 관계식 $\lambda=\frac{1}{\mu}+\alpha$이 성립한다. 행렬 $(A-\alpha I)^{-1}$의 고유치 중 그 절대값이 가장 큰 고유치를 μ라고 하면 $\mu=(\lambda-\alpha)^{-1}$가 되는 A의 고유치 λ가 있다. μ가 가장 크므로 이 λ는 A의 고유치 중 α에 가장 가까운 고유치이다. 따라서 행렬 $(A-\alpha I)^{-1}$에 거듭제곱방법을 이용하여 구하여진 근사값을 $\tilde{\mu}$라고 하면 $\frac{1}{\tilde{\mu}}+\alpha$은 α에 가장 가까운 A의 고유치의 근사값이 된다[알고리즘 2.1, 이].

한편 행렬 $B=(A-\alpha)^{-1}$에 거듭제곱방법을 적용하면 $x_k=\frac{B^k x_0}{\|B^k x_0\|}$는 B의 고유치 중 절대값이 가장 큰 고유치에 대응하는 고유벡터로 수렴한다. B의 고유치는 $(\lambda_i-\alpha)^{-1}, i=1,2,\cdots,n$이므로 절대값이 가장 큰 고유치는 α에 가장 가까운 A의 고유치이다. 따라서 $x_k=\frac{B^k x_0}{\|B^k x_0\|}$는 그 고유치에 대응하는 고유벡터 u_J로 수렴한다. 즉 J는 $|\lambda_J-\alpha|=\min_i |\lambda_i-\alpha|$을 만족하는 첨자이다. 그러므로 $\beta_k=x_k^T A x_k$는 α에 가장 가까운 A의 고유치 λ_J로 수렴한다[알고리즘 2.2, Trefethen].

이와 같이 $(A-\alpha I)^{-1}$에 거듭제곱방법을 적용하는 방법을 역거듭제곱방법이라 한다. 이 방법에서 $(A-\alpha I)^{-1}x_k=x_{k+1}$을 계산하기 위하여 역행렬 $(A-\alpha I)^{-1}$을 직접 구할 필요는 없고 동치인 연립방정식을 풀면 된다. 즉,

$$x_{k+1}=(A-\alpha I)^{-1}x_k \Leftrightarrow (A-\alpha I)x_{k+1}=x_k$$

알고리즘 2.1

임의의 벡터 x_0, 상수 α

for k=1,2,...

$(A-\alpha I)y=x_{k-1}$의 해를 구함

$\beta=x_{k-1}^T y$, $\beta_k=1/\beta+\alpha$

$x_k=y/\|y\|$

end

알고리즘 2.2

임의의 벡터 x_0, 상수 α

for k=1,2,...

$(A-\alpha I)y=x_{k-1}$의 해를 구함

$x_k=y/\|y\|$

$\beta_k=x_k^T A x_k$

end

알고리즘 2.1과 2.2에서 x_k는 서로 같으나 알고리즘 2.1의 β는 x_{k-1}에 대응하는 $(A-\alpha I)^{-1}$의 고유치의 근사치이고 β_k는 α에 가장 가까운 A의 고유치이다. 알고리즘 2.2의 β_k는 x_k에 대응하는 α에 가장 가까운 A의 고유치의 근사치이다. 그러므로 두 알고리즘의 β_k는 같은 값으로 수렴한다. 그러나 연산의 수에 있어서는 거듭제곱방법과 마찬가지로 매 단계에서 알고리즘 2.1은 하나의 연립방정식의 해를 구해야 하나, 알고리즘 2.2에서는 하나의 연립방정식의 해와 하나의 행렬과 벡터의 곱이 더 필요하다. 그러므로 연산의 수에 있어서는 알고리즘 2.1이 알고리즘 2.2보다 작아 더 효율적이다. 알고리즘 2.2는 이론적으로 알고리즘 2.1보다 이해하기가 더 쉽다.

두 알고리즘에서 k가 커짐에 따라 β_k는 모두 α에 가장 가까운 A의 고유치 λ에 수렴하고 x_k는 그에 대응하는 고유벡터에 수렴한다.

Ⅲ. 계산결과

크기가 n인 다음 행렬 A의 고유치는 $\lambda_k = 1-4r\sin^2\frac{k\pi}{2(n+1)}, k=1,2,\cdots,n$으로 알려져 있다[Johnson].

$$A=\begin{bmatrix} 1-2r & r & 0 & \cdots & \cdots & \cdots & 0 \\ r & 1-2r & r & 0 & \cdots & \cdots & 0 \\ 0 & r & \ddots & \ddots & \cdots & \cdots & 0 \\ \vdots & & \ddots & & & & \vdots \\ \vdots & & \ddots & & & \ddots & r \\ 0 & \cdots & & 0 & r & 1-2r & \end{bmatrix}$$

MATLAB 6.5를 이용하여 알고리즘 1.1과 알고리즘 1.2의 계산속도(초)를 비교하였다. $r=-1$일 때 그 결과는 <표 1>과 같다.

<표 1> 거듭제곱방법의 비교

	알고리즘 1.1	알고리즘 1.2	알1.1/알1.2
$n=300, \text{imax}=100$	0.079	0.125	0.632
$n=500, \text{imax}=200$	0.125	0.250	0.5

알고리즘 2.1과 알고리즘 2.2의 경우 $r=-1, \alpha=0$일 때 그 결과는 <표 2>와 같다.

<표 2> 역거듭제곱방법의 비교

	알고리즘 2.1	알고리즘 2.2	알2.1/알2.2
$n=300, imax=100$	1.390	1.4530	0.9566
$n=500, imax=200$	10.250	10.391	0.9864

Ⅳ. 결 론

거듭제곱방법과 역거듭제곱방법으로 행렬의 고유치를 구하는 알고리즘 1.1과 1.2 그리고 알고리즘 2.1과 2.2는 각각 동일하다. 알고리즘 1.1과 1.2의 경우 예상한대로 계산속도는 두 배정도로 나왔다. 역거듭제곱방법의 경우 그 비율이 0.5에 가깝지 않은 이유는 연립방정식을 계산하는 복잡도는 $O(n^3)$인 반면 행렬과 벡터의 곱을 계산하는 복잡도는 $O(n^2)$으로 비록 행렬과 벡터의 곱이 추가되도 그 효과가 작기 때문이다.

역거듭제곱방법의 경우 알고리즘 2.1에 비하여 알고리즘 2.2는 이론이 간단하고 알고리즘을 이해하는 것이 더욱 수월하다. 한편 알고리즘의 복잡도는 알고리즘 2.1이 알고리즘 2.2보다 비교적 적다. 계산결과를 보면 알고리즘 2.1이 알고리즘 2.2보다 더 빠르게 계산되어지나 그 차이는 거듭제곱방법의 경우만큼 크지는 않다. 따라서 거듭제곱방법의 경우 계산적인 면에서는 알고리즘 1.1이 훨씬 효과적이나 역거듭제곱방법의 경우 그 차이는 인정하나 충분히 효과적이라 볼 수는 않는다.

따라서 거듭제곱방법의 경우 알고리즘 1.1을 설명하고, 역거듭제곱방법은 이론적으로도 이해가 쉬운 알고리즘 2.2를 설명하는 것이 더욱 효과적이다.

참고문헌

이규봉. 2003. 실습하며 배우는 수치해석학. 경문사, 서울, 463pp

Johnson, L.W. & Riess, R.D. 1982. Numerical Analysis. Addison-Wesley, 563pp

Trefethen, L.N. & Bau, D. 1997. Numerical Linear Algebra. SIAM, Philadelphia, 361pp