韓國數學敎育學會誌 시리즈 E
<數學敎育 論文集>
제 20집 제 4호, 2006. 11. 575-586

J. Korea Soc. Math. Ed. Ser. E:
Communications of Mathematical Education
Vol. 20 No 4, Nov. 2006. 575-586

실습을 통한 수축방법의 효과적인 이해

이 규 봉 (배재대학교)

1. 서 론

수치해석학에서 고유치문제를 지도하면 처음 나오는 내용이 거듭제곱방법(power method)이다. 이 방법은 0이 아닌 임의의 초기벡터에 주어진 행렬을 거듭해서 곱하면, 곱하여진 벡터는 주어진 행렬의 고유치 중 절대값이 가장 큰 고유치에 대응하는 고유벡터로 수렴한다는 이론에 근거한 것이다. 따라서 행렬과 벡터의 곱으로만 주어진 행렬의 고유치 중 절대값이 가장 큰 고유치와 이에 대응하는 고유벡터를 구할 수 있다.

역거듭제곱방법(inverse power method)은 거듭제곱방법을 응용한 방법이다. 주어진 행렬을 이 행렬의 고유치가 아닌 상수만큼 이동한 후 역행렬을 취하면, 이 행렬의 고유벡터는 원래 주어진 행렬의 고유벡터와 같다는 사실에 이론적 근거가 있다. 따라서 이동된 행렬의 역행렬에 거듭제곱방법을 적용하면 주어진 상수에 가장 가까운 고유치에 대응하는 고유벡터로 수렴하여 주어진 상수에 가장 가까운 고유치를 구할 수 있다.

수축방법(deflation)은 거듭제곱방법이나 역거듭제곱방법으로 구하여진 고유치와 고유벡터를 이용하여 주어진 행렬의 다른 고유치와 고유벡터를 구하는 방법이다. 이론적으로는 수축방법을 거듭 이용하여 모든 고유치와 고유벡터를 구할 수 있다. 그러나 행렬이 매우 크면 마무리오차로 인해 정확한 고유치를 구하는 것에는 효과적이지 못하다.

고유치문제에 관련된 수치해석학을 강의할 때 이 부분을 언급만 하며 지나간다. 그러나 매트랩 같은 응용프로그램을 이용하면 이 부분을 설명하는데 매우 효과적일 수 있다. 따라서 본 논문에서는 매트랩을 이용하여 거듭제곱방법과 역거듭제곱방법 그리고 수축방법까지의 이론적인 연관성을 실습을 통하여 효과적으로 가르치고 학생들로 하여금 스스로 깨달을 수 있게 하는 것에 대하여 서술한다.

2장에서는 거듭제곱방법과 역거듭제곱방법 그리고 수축방법의 이론적인 관련성을 설명하며, 3장에서는 매트랩을 이용한 실습에 대하여 서술한다. 그리고 4장에서 마무리하고자 한다.

---

* ZDM 분류 : N35
* MSC2000 분류 : 97U70
* 주제어 : 고유치, 거듭제곱방법, 역거듭제곱방법, 수축방법

2. 본 론

고유치방정식의 해법은 정사각행렬 A가 주어졌을 때 $Ax=\lambda x$를 만족하는 상수 λ와 벡터 x를 구하는 것이다. 이 상수 λ를 A의 고유치라 하고 0이 아닌 벡터 x를 고유치 λ에 대응하는 A의 고유벡터라고 한다.

크기가 n인 행렬은 중복을 포함하여 복소수 범위에서 n개의 고유치를 갖는다. 그러나 그에 대응하는 일차독립인 고유벡터는 많아야 n개이다. 한편 서로 다른 n개의 고유치를 가지면 그에 대응하는 n개의 고유벡터는 일차독립이다. n차 대칭행렬은 실수 고유치를 갖으며 서로 직교하는 n개의 고유벡터를 갖는다. 따라서 임의의 n차 벡터는 n개의 일차독립인 고유벡터의 일차결합으로 만들어진다.

Gerschgorin 원판 정리에 의하면 고유치는 행렬의 대각성분을 중심으로 적당한 거리 안에 있어야 한다. 따라서 행렬의 고유치의 분포를 짐작할 수 있다. 이를 이용하여 역거듭제곱방법에서 필요한 이동변수 α를 택하면 이에 가까운 고유치를 구할 수 있게 된다(이규봉, 2003)

가. 거듭제곱방법

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 하자. 고유벡터가 일차독립이므로 x_0는 적당한 상수 $\alpha_1, \alpha_2, \cdots, \alpha_n$이 존재하여 다음과 같이 고유벡터의 일차결합으로 나타난다.

$$x_0=\alpha_1 u_1+\alpha_2 u_2+\cdots+\alpha_n u_n,\ \alpha_1\neq 0$$

$A^k u_i=\lambda_i^k u_i$, $i=1,2,\cdots,n$이므로

$$\begin{aligned} x_k &= A^k(\alpha_1 u_1+\alpha_2 u_2+\cdots+\alpha_n u_n) \\ &= \alpha_1\lambda_1^k u_1+\alpha_2\lambda_2^k u_2+\cdots+\alpha_n\lambda_n^k u_n \\ &= \lambda_1^k[\alpha_1 u_1+\alpha_2(\frac{\lambda_2}{\lambda_1})^k u_2+\cdots+\alpha_n(\frac{\lambda_n}{\lambda_1})^k u_n] \end{aligned}$$

이다. 여기서 $|\frac{\lambda_2}{\lambda_1}|,\cdots,|\frac{\lambda_n}{\lambda_1}|<1$이므로 k가 커짐에 따라 $(\frac{\lambda_i}{\lambda_1})^k$은 0으로 수렴하게 된다. 즉 충분히 큰 k에 대하여 x_k는 λ_1에 대응하는 고유벡터에 가깝게 된다.

$\beta_k=\frac{x_k^TAx_k}{x_k^Tx_k}$라 하고 $x_k^Tu_i=\gamma_i,\ i=1,2,\cdots,n$라고 하면

$$\beta_k=\lambda_1\frac{\alpha_1\gamma_1+\alpha_2(\frac{\lambda_2}{\lambda_1})^{k+1}\gamma_2+\cdots+\alpha_n(\frac{\lambda_n}{\lambda_1})^{k+1}\gamma_n}{\alpha_1\gamma_1+\alpha_2(\frac{\lambda_2}{\lambda_1})^k\gamma_2+\cdots+\alpha_n(\frac{\lambda_n}{\lambda_1})^k\gamma_n}$$

이고 $1>|\frac{\lambda_2}{\lambda_1}|\geq\cdots\geq|\frac{\lambda_n}{\lambda_1}|$이므로 β_k는 $(\frac{\lambda_2}{\lambda_1})^k$가 0에 수렴하는 만큼 빨리 λ_1에 수렴한다. 즉, $\beta_k=\lambda_1+O(\left(\frac{\lambda_2}{\lambda_1}\right)^k)$이다. 따라서 $|\lambda_1|$과 $|\lambda_2|$의 차이가 클수록 $|\frac{\lambda_2}{\lambda_1}|$이 작아지므로 더욱 빨리 수렴한다. 만약 $\alpha_1=0$이면 이론적으로는 β_k가 λ_2에 수렴할 것이나 마무리오차로 인하여 실질적으로는 그 경우도 λ_1에 수렴한다. 이 방법을 거듭제곱방법이라고 한다.

k가 커짐에 따라 x_k는 u_1에 수렴하고 β_k는 λ_1에 수렴한다. 만일 행렬 A가 대칭이면 고유벡터가 서로 수직이므로 $\gamma_i=\lambda_1^k\alpha_i\left(\frac{\lambda_i}{\lambda_1}\right)^k$이다. 따라서 $\beta_k=\lambda_1+O[\left(\frac{\lambda_2}{\lambda_1}\right)^{2k}]$가 되어 더욱 빨리 수렴한다.

나. 역거듭제곱방법

거듭제곱방법을 이용하면 절대값이 가장 큰 고유치에 수렴한다. 행렬을 이동하면 이동한 만큼 고유치도 변한다. 이를 이용하여 임의의 상수에 가장 가까운 고유치를 구할 수 있다.

상수 α에 대하여 $Ax=\lambda x$이면 $(A-\alpha I)x=(\lambda-\alpha)x$가 된다. α가 A의 고유치가 아니면 행렬 $A-\alpha I$는 정칙이고 역행렬이 존재한다. 따라서

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

이다. 역행렬 $(A-\alpha I)^{-1}$의 고유치는 $(\lambda-\alpha)^{-1}$가 되고 대응하는 고유벡터는 λ에 대응하는 A의 고유벡터와 같다. 따라서 행렬 $B=(A-\alpha I)^{-1}$에 거듭제곱방법을 적용하면 $x_k=\dfrac{B^k x_0}{\|B^k x_0\|}$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 고유치로 표현된다. 따라서 x_k는 α에 가장 가까운 고유치에 대응하는 고유벡터 u_J로 수렴한다. 여기서 J는 $|\lambda_J-\alpha|=\min_i|\lambda_i-\alpha|$을 만족하는 첨자이다. 그러므로 β_k는 α에 가장 가까운 A의 고유치 λ_J로 수렴한다(Trefethen, 1997).

이와 같이 $(A-\alpha I)^{-1}$에 거듭제곱방법을 적용하는 방법을 역거듭제곱방법이라 한다. 이 방법에서 $x_{k+1}=(A-\alpha I)^{-1}x_k$은 동치인 연립방정식을 풀면 된다. 즉

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

상수 α가 복소수이면 그에 가까운 복소 고유치를 구하는데 매우 효과적이다. 거듭제곱방법에서 이미 언급한 것처럼 α가 고유치 λ_J에 가까울수록 그리고 λ_{J-1}과 λ_{J+1}이 λ_J로부터 멀리 분포될수록 더욱 빨리 수렴된다.

다. 수축방법

거듭제곱방법 또는 역거듭제곱방법을 이용하여 대칭행렬 A의 고유치 λ_ℓ와 그에 대응하는 고유벡터 u_ℓ를 구했으면 다음 정리에 의해 또 다른 고유치와 그에 대응하는 고유벡터를 구할 수 있다(Johnson, 1982).

정리. n차 대칭행렬 A에 대하여 $Au_i=\lambda_i u,\ i=1,\cdots,n$이라고 하자. 단 $\|u_i\|=1$. 그러면 행렬 $B\equiv A-\lambda_\ell u_\ell u_\ell^T$는 λ_ℓ을 제외한 A의 고유치와 0을 갖는다. 즉

$$Bu_i=\lambda_i u_i,\ i\neq\ell,\ Bu_\ell=0$$

이론적으로 이 정리는 대칭행렬의 모든 고유치와 고유벡터를 구할 수 있게 해준다. 거듭제곱방법을 이용하여 λ_1과 이에 대응하는 단위고유벡터 u_1를 구한 다음, 행렬 $A-\lambda_1 u_1 u_1^T$를 구한다. 이

행렬에 다시 거듭제곱방법을 적용하면 그 다음 번 고유치 λ_2와 고유벡터 u_2를 구할 수 있다. 이렇게 구하는 방법을 수축방법(deflation)이라고 한다. 이 방법을 거듭 이용하면 이론적으로는 모든 고유치를 구할 수 있으나 실질적으로는 오차가 클 수 있다. 그러므로 구하여진 근사값을 역거듭제곱방법의 이동변수 α로 사용하여 더욱 좋은 근사값을 구할 수 있다.

3. 실습을 통한 효과적인 이해

이 장에서는 각 방법의 알고리즘을 서술하고 매트랩을 이용한 프로그램을 제시한 후 이 프로그램을 실행하여 이론적인 사실을 확인한다.

거듭제곱방법의 알고리즘과 매트랩 프로그램 powermethod.m은 다음과 같다.

알고리즘 : 거듭제곱방법	powermethod.m
초기벡터 u	function [u,beta]=powermethod(A,u,kmax)
for k=1,2,...,kmax	for k=1:kmax
$y = Au$	y=A*u;
$\beta = u^T y$	beta=u'*y;
$u = y/\|\|y\|\|$	u=y/norm(y);
end	end

실습으로 확인할 필요가 있는 것은, 첫 째 가장 큰 고유치로 정말 수렴하는가? 둘 째, $|\lambda_1|$과 $|\lambda_2|$의 차이가 클수록 빨리 수렴하는가? 그리고 초기벡터 x_0에서 $\alpha_1 = 0$이어도 λ_1에 수렴하는가? 이다.

실습 1. $A = \begin{pmatrix} 9 & 4 & 4 & 8 \\ 13 & 0 & 4 & 8 \\ 5 & 0 & -4 & 0 \\ 6 & 8 & 8 & 12 \end{pmatrix}$에 대하여 $A_0 = A + I, A_1 = A + 10I, A_2 = A + 10^2 I$ 그리고 $A_3 = A + 10^3 I$ 라고 하자. A의 고유치가 24, -4,-4, 1이므로 A_0의 고유치는 25, -3, -3, 2, A_1의 고유치는 34, 11, 6, 6, A_2의 고유치는 124, 101, 96, 96, A_3의 고유치는 1024, 1001,996, 996이 된다. powermethod.m에 $\frac{|\lambda_1 - \lambda_1^{(k)}|}{|\lambda_1|} < 10^{-8}$이면 프로그램을 중지하는 과정을 삽입한 후 나온 반복횟수의 값 k은 각각의 경우 다음과 같다.

<표 1> $|\frac{\lambda_2}{\lambda_1}|$와 반복횟수의 비교

	A	A_0	A_1	A_2	A_3
$\lvert\frac{\lambda_2}{\lambda_1}\rvert$	0.1667	0.12	0.3235	0.8145	0.9775
k	11	9	13	56	440

따라서 가장 큰 고유치 λ_1으로 수렴함을 확인할 수 있고, 또한 $|\lambda_1|$과 $|\lambda_2|$의 차이가 클수록 빨리 수렴하는 것이 확인된다.

실습 2. $A=\begin{pmatrix}6&4&4&1\\4&6&1&4\\4&1&6&4\\1&4&4&6\end{pmatrix}$의 고유치는 $\lambda=15,5,5,-1$이고 이에 대응하는 고유벡터는 각각 $u_1=\begin{pmatrix}1\\1\\1\\1\end{pmatrix}, u_2=\begin{pmatrix}1\\4\\-4\\1\end{pmatrix}, u_3=\begin{pmatrix}-4\\1\\-1\\4\end{pmatrix}, u_4=\begin{pmatrix}-1\\1\\1\\-1\end{pmatrix}$이다. $x_0=u_2+u_3+u_4=\begin{pmatrix}-4\\6\\-4\\2\end{pmatrix}$일 때 반복횟수$k$에 따른 고유치 β_k는 다음과 같다.

<표 2> $a_1=0$인 경우 반복횟수에 따른 고유치의 비교

k	30	35	40	45	50
β_k	5.0000	5.0046	14.6439	15.0000	15.0000

따라서 초기벡터 x_0에서 $a_1=0$이어도 초기 반복에는 λ_2로 수렴하는 듯 하지만, 좀 더 반복하면 이론대로 λ_1에 수렴함을 알 수 있다.

역거듭제곱방법의 알고리즘과 매트랩 프로그램 inv_power.m은 다음과 같다.

알고리즘 : 역거듭제곱방법
초기벡터 u, 상수 α
for k=1,2,...,kmax
  $(A-\alpha I)y=u$의 해를 구함
  $u=y/\|y\|$
  $\beta=x^TAx$
 end

```
inv_power.m
function [u,beta]=inv_power(A,u,mu,kmax)
B=A-eye(n)*mu;
for k=1:kmax
    u=B\u;
    u=u/norm(u);
    beta=u'*(A*u);
end
```

실습으로 확인할 필요가 있는 것은, 첫 째 α에 가장 가까운 고유치로 정말 수렴하는가? 둘 째, α가 고유치에 가까울수록 더욱 빨리 수렴할까? 그리고 복소고유치는 구해질까? 이다.

실습 3. $A=\begin{pmatrix}9&4&4&8\\13&0&4&8\\5&0&-4&0\\6&8&8&12\end{pmatrix}$의 고유치는 $\lambda_1=24,\lambda_2=-4,\lambda_3=-4,\lambda_4=1$이다. 따라서 $(A-\alpha I)^{-1}$의 고유치는 $\frac{1}{\lambda_i-\alpha}$, $i=1,2,3,4$이다. $\alpha=25,50,100$일 때는 $\lambda=\lambda_1$으로, $\alpha=-5,-10,-20$일 때는 $\lambda=\lambda_2$로 그리고 $\alpha=2,5,10$일 때는 $\lambda=\lambda_4$로 수렴한다. 각각의 경우 $\frac{|\lambda-\lambda^{(k)}|}{|\lambda|}<10^{-8}$이 되는 반복횟수의 값 k는 다음과 같다.

<표 3> $|\lambda-\alpha|$와 반복횟수의 비교

α	25	50	100	-5	-10	-20	2	5	10
$\|\lambda-\alpha\|$	1	26	76	1	6	16	1	4	9
k	5	22	51	10	27	60	13	27	45

따라서 α에 가장 가까운 고유치로 수렴하고 α가 고유치에 가까울수록 더욱 빨리 수렴하는 것을 확인할 수 있다.

실습 4. $\begin{pmatrix}-2&-2&-9\\-1&1&-3\\1&1&4\end{pmatrix}$ 의 고유치는 $\lambda_1=1+i,\lambda_2=1-i,\lambda_3=1$이다. $\alpha=1+0.5i,1+2i,2+2i$일 때 $\frac{|\lambda_1-\lambda^{(k)}|}{|\lambda_1|}<10^{-8}$이 되는 반복횟수의 값 k 은 각각 다음과 같다.

<표 4> $|\lambda_1 - \alpha|$와 반복횟수의 비교

α	$1+0.5i$	$1+2i$	$2+2i$
$\lvert\lambda_1-\alpha\rvert$	0.5	1	$\sqrt{2}$
k	17	26	39

복소고유치가 있는 경우 α를 복소수로 사용하면 복소고유치로 수렴하고, 복소고유치의 경우도 역시 α에 가장 가까운 고유치로 수렴하며 α가 고유치에 가까울수록 더욱 빨리 수렴하는 것을 확인할 수 있다.

수축방법의 알고리즘과 여러 고유치를 구하도록 수정한 프로그램 deflation.m은 다음과 같다.

알고리즘 : 수축방법

초기벡터 x_0

A의 고유치와 단위고유벡터 (λ, u)

$B = A - \lambda u u^T$

λ=거듭제곱방법(B, x_0)

(λ, u)=역거듭제곱방법(A, x_0,λ)

```
deflation.m
function [eigvl,u]=deflation(A,x0,kmax,no)
[lam,u]=powermethod(A,x0,kmax);
B=A; eigvl(1)=lam;
for k=2:no
    B=B-lam*u*u';
    [lam,u]=powermethod(B,x0,kmax);
    [lam,u]=inv_power(A,x0,lam,kmax);
    eigvl(k)=lam;
end
```

실습으로 확인할 필요가 있는 것은, 첫 째 수축된 행렬 $A - \lambda_\ell u_\ell u_\ell^T$의 고유치는 λ_ℓ을 제외한 A의 고유치와 0인가? 둘 째, 수축된 행렬에 거듭제곱방법을 적용하여 구한 고유치를 역거듭제곱방법의 이동변수로 사용하면 더 정확한 고유치가 되는가? 그리고 수축방법을 거듭 사용하여 모든 고유치를 구할 수 있는가? 이다.

실습 5. $A = \begin{pmatrix} 6 & 4 & 4 & 1 \\ 4 & 6 & 1 & 4 \\ 4 & 1 & 6 & 4 \\ 1 & 4 & 4 & 6 \end{pmatrix}$의 고유치는 $\lambda = 15, 5, 5, -1$이고 이에 대응하는 고유벡터는 각각 $u_1 = \begin{pmatrix} 1 \\ 1 \\ 1 \\ 1 \end{pmatrix}, u_2 = \begin{pmatrix} 1 \\ 4 \\ -4 \\ 1 \end{pmatrix}, u_3 = \begin{pmatrix} -4 \\ 1 \\ -1 \\ 4 \end{pmatrix}, u_4 = \begin{pmatrix} -1 \\ 1 \\ 1 \\ -1 \end{pmatrix}$이다. 매트랩으로 수축행렬 $B = A - \lambda_\ell u_\ell u_\ell^T$의 고유

치와 고유벡터를 다음과 같이 확인할 수 있다. 단 u_ℓ은 단위벡터이다.

```
>> A=[6 4 4 1;4 6 1 4;4 1 6 4;1 4 4 6];
>> [Ua,Ka]=eig(A)
Ua =
   -0.5000    0.5567   -0.4359    0.5000
    0.5000    0.4359    0.5567    0.5000
    0.5000   -0.4359   -0.5567    0.5000
   -0.5000   -0.5567    0.4359    0.5000
Ka =
   -1.0000         0         0         0
         0    5.0000         0         0
         0         0    5.0000         0
         0         0         0   15.0000
>> B=A-15*U(:,4)*U(:,4)';
>> [Ub,Kb]=eig(B)
Ub =
    0.5000   -0.5000    0.0968    0.7005
   -0.5000   -0.5000   -0.7005    0.0968
   -0.5000   -0.5000    0.7005   -0.0968
    0.5000   -0.5000   -0.0968   -0.7005
Kb =
   -1.0000         0         0         0
         0   -0.0000         0         0
         0         0    5.0000         0
         0         0         0    5.0000
```

이 결과에서 Ua의 첫 번째 벡터와 Ub의 첫 번째 벡터 그리고 Ua의 네 번째 벡터와 Ub의 두 번째 벡터는 고유벡터로서 같다. 또한 Ua의 두 번째 벡터와 세 번째 벡터가 생성하는 공간과 Ub의 세 번째 벡터와 네 번째 벡터가 생성하는 공간은 서로 같으므로 축소행렬의 고유벡터도 같다. 따라서 수축된 행렬 $A-\lambda_\ell u_\ell u_\ell^T$의 고유치는 λ_ℓ을 제외한 A의 고유치와 0이고 고유벡터도 같음을 확인할 수 있다.

실습 5. 대칭행렬

$$A=\begin{bmatrix} 52 & -4 & 4 & 7 & -9 & 6 & 4 & 0 & -6 & -2 \\ -4 & 31 & 3 & -7 & -2 & 5 & 3 & 1 & 1 & 0 \\ 4 & 3 & 31 & -3 & 3 & -9 & 2 & 0 & -3 & 0 \\ 7 & -7 & -3 & 46 & -5 & 3 & -3 & 6 & 2 & 0 \\ -9 & -2 & 3 & -5 & 36 & 2 & 1 & 6 & 2 & 1 \\ 6 & 5 & -9 & 3 & 2 & 57 & -7 & 7 & 7 & 4 \\ 4 & 3 & 2 & -3 & 1 & -7 & 37 & -4 & 0 & -4 \\ 0 & 1 & 0 & 6 & 6 & 7 & -4 & 44 & -5 & 5 \\ -6 & 1 & -3 & 2 & 2 & 7 & 0 & -5 & 28 & 1 \\ -2 & 0 & 0 & 0 & 1 & 4 & -4 & 5 & 1 & 17 \end{bmatrix}$$

의 고유치는 69.1547, 61.6673, 49.2791, 45.2859, 35.6923, 33.1971, 29.2709, 21.3923, 19.0389, 15.0214이다. deflation.m을 이용하여 다른 나머지를 확인해 보자. 여기서 x0는 성분이 모두 1인 벡터이다.

```
>> [lam,u]=powermethod(A,x0,50);
>> lambda=deflation(A,lam,u,x0,40,2)
lambda =   69.1547   61.6673
>> lambda=deflation(A,lam,u,x0,40,3)
lambda =   69.1547   61.6673   49.2791
>> lambda=deflation(A,lam,u,x0,40,4)
lambda =   69.1547   61.6673   49.2791   45.2859
>> lambda=deflation(A,lam,u,x0,40,5)
lambda =   69.1547   61.6673   49.2791   45.2859   35.6923
>> lambda=deflation(A,lam,u,x0,40,6)
lambda =   69.1547   61.6673   49.2791   45.2859   35.6923   33.1971
>> lambda=deflation(A,lam,u,x0,40,7)
lambda =   69.1547   61.6673   49.2791   45.2859   35.6923   33.1971   29.2709
```

실습 5에서 수축된행렬에 거듭제곱방법을 적용하여 구한 고유치를 역거듭제곱방법의 이동변수로 사용하여 고유치를 구하였다. 그리고 이 방법을 거듭 사용하여 모든 고유치를 구할 수 있음을 보였다.

지금까지의 실습에서 행렬을 일일이 만드는 것은 매우 성가신 일이다. 대칭인 강대각지배행렬을 생성하는 아래와 같은 gen_dom_mat.m을 사용하면 매우 큰 행렬에서 다양하게 실습을 할 수 있다.

```
function [A,x]=gen_dom_mat(n,k)
% 입력 : n 행렬의 크기, k 성분의 범위
% 출력 : A [-k,k]의 정수로 이루어진 n차 대칭행렬, x 성분이 모두 1인 열벡터
A=2*k*rand(n,n)-k;
A=round((A+A')/2);
for i=1:n
    sum=0;
    for j=1:n
        sum=sum+abs(A(i,j));
    end
    A(i,i)=sum; x(i)=1;
end
x=x';
```

3. 결 론

수치해석학은 수학의 여러 분야 중에서 실습이 필요한 과목이다. 이 특징을 살려 수치해석학 교육을 해야 한다. 수학적으로 증명된 이론이나 알고리즘을 컴퓨터를 이용하여 실습하고 그 결과가 옳음을 확인하는 것은 교육적으로 매우 바람직하다.

고유치를 구하는 첫 번 단계인 거듭제곱방법, 역거듭제곱방법 그리고 수축방법에서 배우는 알고리즘과 수학적인 사실이 옳음을 실습을 통하여 보여주어야 한다. 실습하다 보면 이론대로 나오지 않을 수도 있다. 그러면 왜 그런지 파악하여야 한다. 예를 들어, 거듭제곱방법은 가장 큰 고유치가 나오도록 되어 있으나 충분한 반복을 하지 않으면 다른 고유치가 구해지는 경우가 있다. 이 경우 충분히 더 반복하면 예상한대로 가장 큰 고유치에 수렴함을 보일 수 있다.

이론과 실습이 서로 조화를 이루어 수치해석학을 교육하면 수학의 필요함을 더욱 강하게 학생들에게 보여줄 수 있다.

참 고 문 헌

이규봉 (2003). 실습하며 배우는 수치해석학, 서울: 경문사

Johnson, L. W. & Riess, R. D. (1982). *Numerical Analysis*, Addison-Wesley

Trefethen, L. N. & Bau, D. (1997). *Numerical Linear Algebra*, SIAM, Philadelphia

Effective Teaching of Deflation using Computer Practice

Gyou Bong Lee

Dept. of Applied Math., Paichai University, Daejeon 302-175, Korea

gblee@pcu.ac.kr

Both theory and experiment are very important parts in sciences. Especially in mathematics, theory seems to be very important, but experiment or practice doesn't. Numerical analysis of many parts in mathematics needs practice in computer. In this paper, I suggest that computer-practicing in teaching power method, inverse power method and deflation to calculate eigenvalues and eigenvectors is good in understanding the theory. It also makes students sure that mathematics is helpful.

* ZDM Classification : N35
* 2000 Mathematics Subject Classification : 97U70
* Key Words : eigenvalue, power method, inverse power method, deflation