2008 년 2 학기 HW#6

# 수치해석기초

## HW#7 : **Eigenvalue Problems for 3-D Particle Diffusion**

원자핵공학과

2003-12491

이 원 재

## A. Matrix Eigenvalue Problem

1. **Construct Matrix B using the MATLAB function inv(A)*S. Determine the eigenvalues and the normalized eigenvectors of B using the MATLAB script [U,LAM]=eig(B). Determine the dominance ratio of this system as well.**

문제 풀이를 위해 다음과 같은 MATLAB 함수를 작성한다.

In file lesc3d.m

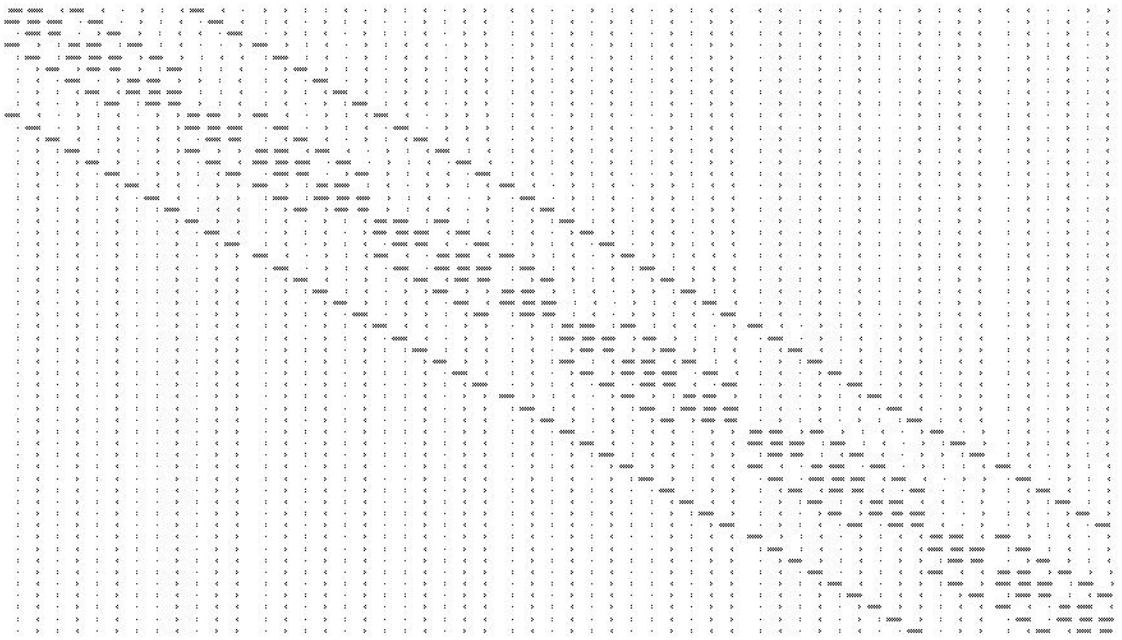
```
function [A,S] = lesc3d(m,a,D)
%Linear Eigenvalue System Constructor 3-D
%The input parameter 'm' is the number of grids for x direction
%the second input parameter 'a' is physical size in x direction
%the third input parameter 'D' is diffusion coefficient
%this function returns (m*n*k x m*n*k) matrix A and S

a=a; %takes input parameter
b=a;
c=2*a;
D=D;%diffiusion coefficient 1.0
sigA=0.10; %absorption cross section
sigS=0.12; %reaction cross section
m=m; %x축 mesh 개수
n=m; %y축 mesh 개수
k=2*m; %z축 mesh 개수
N=m*n; %k번째 plane의 총 mesh 개수
Nk=N*k; %총 mesh 개수
hx=a/m; %x축 mesh 크기
hy=b/n; %y축 mesh 크기
hz=c/k; %z축 mesh 크기
hx2=hx*hx;
hy2=hy*hy;
hz2=hz*hz;
B2=sigA/D; %B^2 = sigA/D , Buckling
A=sparse(Nk,Nk);
S=sparse(Nk,Nk);
```

```
for (i=1:Nk)
        A(i,i)=B2;
%뒤따르는 if 문은 두개씩 짝을 이루고 있는데 그 두 if 문이 동시에 참이되지 못한다.
%즉, 첫번째 if 가 사실이면 두번째 if 문은 참이 될 수 없다. (세가지 경우 존재,TF,FT,FF)
따라서 첫번째 if문이 사실이어서 +2/hz2를 해주면 자동으로 그 다음 if 문에서 +1/hz2를
해주기 때문에 null flux 조건이 만족된다.
%% 만약 첫번째 if가 거짓이고, 두번째 if 가 참이면 그래도 첫번째 if 문의 else 이하에
서 +1/hz2를 해주고 두번째 if 문에서 +2/hz2를 해주기 때문에 결과적으로 +3/hz2 의 null
flux 조건을 만족시킨다. 마찬가지로 두개다 거짓일때 +2/hz2가 만족 된다.
        if(i<=N) %첫번째 plane 즉, z=0 인 block에서는 그보다 위쪽에 접근 못한다.
                A(i,i)=A(i,i)+2/hz2; %z=0에 대해서 null flux
        else
                A(i,i-N)=-1/hz2; %top cell
                A(i,i)=A(i,i)+1/hz2; %z에 대해서 일반적인경우
        end
        if(i>N*(k-1))
                %마지막 plane z=c 인 block 이면 bottom에 접근 할 수 없다.
                A(i,i)=A(i,i)+2/hz2; %z=c에서 null flux
        else
                A(i,i+N)=-1/hz2; %bottom cell
                A(i,i)=A(i,i)+1/hz2;
        end
        j=fix(i/N);%현재 몇번째 plane에 있는지 알아봄
        if(i>N*j && i<=N*j+m) %y=0 에 있으면 위쪽(북쪽)에 접근할 수 없다.
                A(i,i)=A(i,i)+0/hy2;%y=0에서 zero current
        else %그렇지 않으면 접근할 수 있음
                A(i,i-m)=-1/hy2;%북쪽 셀
                A(i,i)=A(i,i)+1/hy2;
        end
        if((i>N*j+(n-1)*m) || mod(i,N)==0 ) %y=b에 있으면 (남쪽)에 접근할 수 없다.
                A(i,i)=A(i,i)+2/hy2; %y=b에서 null flux
        else %그렇지 않으면 접근 할 수 있다
                A(i,i+m)=-1/hy2;%남쪽 셀
                A(i,i)=A(i,i)+1/hy2;
        End
        if(mod(i,m)==1) %x=0에 있으면 왼쪽에 접근할 수 없다.
```

```
            A(i,i)=A(i,i)+0/hx2; %x=0에서 zero current
        else
            A(i,i-1)=-1/hx2; %왼쪽(서쪽) 셀
            A(i,i)=A(i,i)+1/hx2;
        end
        if(mod(i,m)==0) %오른쪽 끝에 있으면 더이상 오른쪽은 없다.
            ;
            A(i,i)=A(i,i)+2/hx2; % x=a에서 null flux
        else
            A(i,i+1)=-1/hx2; %오른쪽(동쪽) 셀
            A(i,i)=A(i,i)+1/hx2;
        end

        S(i,i)=sigS/D; %RHS 의 matrix construction

end
```

우선 이 함수로 만들어진 행렬 A가 septa diagonal matrix로 잘 만들어져 있는지 확인해본다.

```
>> [A,S]=lesc3d(3,100,1);
>> full(A)
```



이제 실제 문제에 맞도록 grid 개수를 주고 m=4, n=4 ,k=8, B 매트릭스를 구한 후 B의 eigenvector를 구해보면

```
>> [A,S]=lesc3d(4,100,1);
>> B=inv(A)*S;
>> [U,LAM]=eig(full(B));
>> LAM

LAM =

  Columns 1 through 7

    1.1913         0         0         0         0         0         0
         0    1.1831         0         0         0         0         0
         0         0    1.0108         0         0         0         0   :
         0         0         0    1.0129         0         0         0   :
         0         0         0         0    1.1573         0         0   :
         0         0         0         0         0    1.0189         0
         0         0         0         0         0         0    1.1633
                                      ....
```

Dominance ratio는 다음과 같이 구할 수 있다.

```
>> L=sort(max(LAM),'descend');
L =

   Columns 1 through 10

    1.1913    1.1831    1.1712    1.1712    1.1712    1.1633    1.1633  ….

>> Dominance_ratio=L(2)/L(1)

Dominance_ratio =
    0.9932

>>
```

고유값은 가장 큰 값이 1.1913 이고 그 다음이 1.1831이어서 이 둘을 나누면 dominace ratio를 1.1831로 구할 수 있다. 한편, 세번째로 큰 고유값이 1.1712 인데 비슷한 값이 3개나 존재하는 것을 확인 할 수 있다.

**2. Use the regular power method to find the maximum eigenvalue and the corresponding eigenvector of B. Terminate the iteration when the change in the eigenvalue between two successive values is smaller than $1.0\times10^{-6}$. Normalize the eigenvector and compare it with the reference solution obtained from Step 1.**

다음과 같은 MATLAB 함수를 작성한다.

In file pm.m

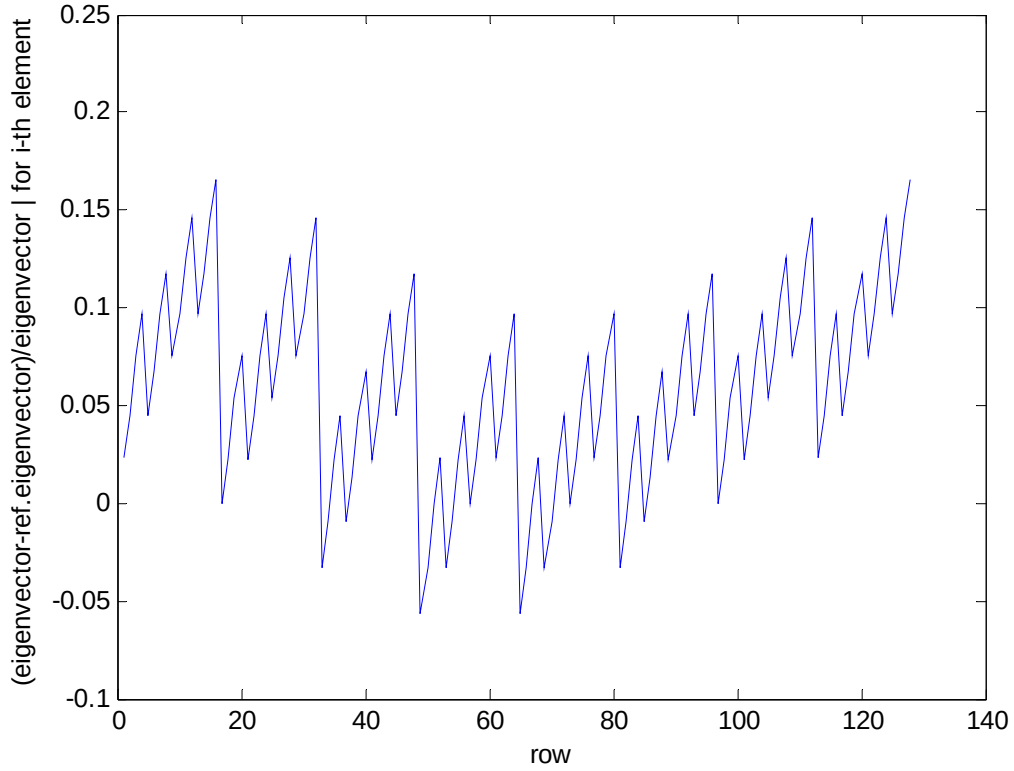
```
function [lamdak,xk] = pm(B)
%Power Method
epsilon=1.0e-6;
maxIter=1000;
n=length(B);
xk=ones(n,1);
xk_1=ones(n,1); %initial guess of eigenvector
lamdak=1;
lamdak_1=1; %initial guess of eigenvalue
```

```
for (k=1:maxIter)
    xk_1=xk_1./lamdak_1;   %scaling
        xk=B*xk_1;             %matrix-vector multiplication
        lamdak=(xk'*xk)/(xk'*xk_1); %obtaining new eigenvalue
        if (abs(lamdak-lamdak_1)<epsilon)
            break;
        end
        xk_1=xk;
        lamdak_1=lamdak;
end
xk=xk/lamdak; %마지막으로 scaling 시켜줌
x_k=x_k/norm(x_k);   %normalization
```

```
>> [l_pm,x_pm]=pm(B);
>> l_pm

l_pm =

      1.1913

>>
```

위에서 구한 제 1 고유값과 같은 결과를 내는 것을 알 수 있다. 고유벡터를 비교하기 위해 다음을 수행해 본다.

```
>> for (i=1:length(x_pm)) grph(i,1)=i;grph(i,2)=(x_pm(i)-U(i,1))/x_pm(i); end
>> plot(grph(:,1),grph(:,2));
>> xlabel('row')
>> ylabel('(eigenvector-ref.eigenvector)/eigenvector | for i-th element')
```




```
>> norm(x_pm-U(:,1))

ans =

    0.0371

>>
```

eig( )로 구한 eigenvector와 비교해 볼 때 완전히 같지는 않은 것을 알 수 있다.
이는 아마도 power method 의 수렴 정도에 의존할 수 있다. 따라서 수렴 조건을 1.E-10 까지로 바꾼 다음에 차이를 조사해 보면 (pm.m 내의 수렴 조건 바꾼후 )

```
>> [l_pm,x_pm]=pm(B);
>> norm(x_pm-U(:,1))

ans =
```

```
   3.7245e-004
>>
```

수렴 조건을 좀더 엄밀하게 하면 고유 벡터가 eig( )로 구한 고유벡터와 비슷해 지는 것을 확인 할 수 있다.

**B. Inverse Power Method**

**1. Implement the inverse power method. For checking the convergence of the inner iteration, use the relative error norm of the two successive iterate vectors and set the convergence criterion to $1.0\times10^{-6}$. For checking the convergence of the outer iteration, use also the relative norm of the two successive eigenvectors. Make sure that you store the old eigenvector to a separate variable such as *phid* and use the current eigenvector as the initial guess to begin each inner iteration. Set the convergence criterion of the power iteration to $1.0\times10^{-5}$.**

다음과 같은 함수를 작성한다.
In file ipm.m

```
function [lamdak,xk] = ipm(A,S)
A=full(A); %sparse matrix 자료형은 참조하는데 시간이 오래걸림
iepsilon=1.e-6;oepsilon=1.e-5;
maxIter=1000;
n=length(A);
xk=ones(n,1);xk_1=ones(n,1); %initial guess of eigenvector
lamdak=1;lamdak_1=1; %initial guess of eigenvalue
xk_1=xk_1./lamdak_1;   %initial scaling
for (k=1:maxIter)
        %이 위치에서 xk_1은 이미 스케일링 되어 있는 상태
      phi=xk_1;
      phid=xk_1;
        %%%%해 벡터 초기화%%%%%%%%%%%%%%%%%%%%%%%%%%
    for (i=1:n)
        b(i)=xk_1(i)*S(i,i);%이미 스케일링 하면서 lamdak_1로 나누기가 포함되어 있음
    end
    for (j=1:maxIter) %inner iteration
```

```
            for (i=1:n)
                suml=0.0;
                sumu=0.0;
                for (m=1:i-1)
                    suml=suml+A(i,m)*phi(m);
                end
                for (m=i+1:n)
                    sumu=sumu+A(i,m)*phid(m);
                end
                phi(i)=(b(i)-suml-sumu)/A(i,i);
            end
            rho_tilda=norm(phi-phid)/norm(phi); %오차의 상대값
            if( rho_tilda < iepsilon)
                break;
            end
            phid=phi; %예전것에 현재 나온 것을 대입
        end         %matrix-vector multiplication
        xk=phi;
        lamdak=(xk'*xk)/(xk'*xk_1); %obtaining new eigenvalue
        xk=xk./lamdak;%scaling
        if (   norm(xk-xk_1)/norm(xk) < oepsilon )
            break;
        end
        xk_1=xk; %스케일링 되어 있음
        lamdak_1=lamdak;
end
```

```
>> [l_ipm,x_ipm]=ipm(A,S);
>> l_ipm
l_ipm =

      1.1913
>> norm(x_ipm-U(:,1))

ans =
   5.8024e-004
```

>>
여기서 주의 해야할 점은 매트릭스 A를 매개변수로 넘겨주고나서 매트릭스의 원소를 참조하려 할 때, full 매트릭스로 해야 한다는 점이다. Sparse 매트릭스를 넘기면 메모리는 아낄수 있지만 내부 루프에서 인덱스로 매트릭스 원소를 참조하는데 연산시간이 오래 걸리게 된다.

**2. During the outer iteration, evaluate the true relative error by subtracting the normalized eigenvector from the reference eigenvector you obtained from A.1. Plot both the true error and pseudo error vs. iteration on a semilog graph. Explain why you would see large difference in the two curves.**

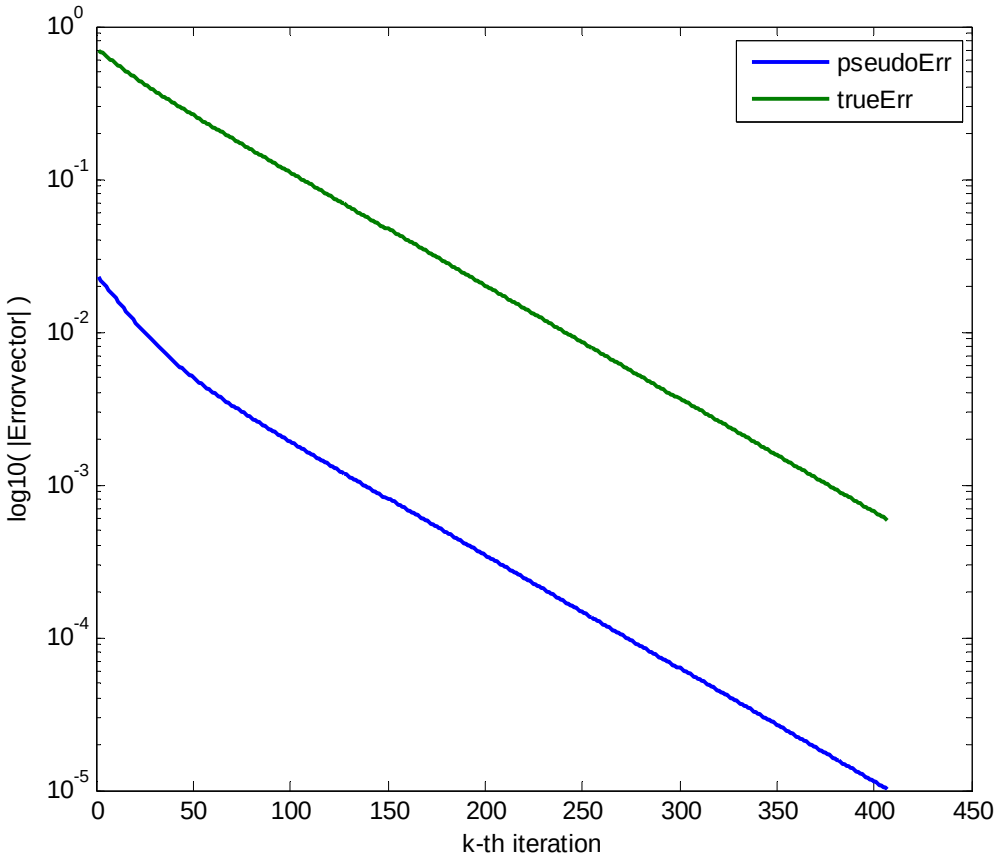
다음과 같은 함수를 작성한다.

In file hw7b2.m

```
function [lamdak,xk] = hw7b2(A,S)
B=inv(A)*S;
[interU,interLAM]=eig(full(B));
refEigVec=(interU(:,1));

A=full(A);
graphmat=zeros(1,3);
iepsilon=1.e-6;
oepsilon=1.e-5;

%%….. 중간 생략

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
xknorm=xk/norm(xk); %스케일링 된 벡터를 normalize
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

xk_1=xk; %스케일링 되어 있음
lamdak_1=lamdak;

%%%%%5%%graph%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
trueErrNorm=norm(xknorm-refEigVec);

graphmat(k,1)=k;
```

```
graphmat(k,2)=pseudoErr;
graphmat(k,3)=trueErrNorm;

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
end

semilogy(graphmat(:,1),(graphmat(:,2)),graphmat(:,1),(graphmat(:,3)),'LineWidth',2);
legend('pseudoErr','trueErr')
xlabel('k-th iteration')
ylabel('log10( |Errorvector| )')
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
```

>> hw7b2(A,S);
>>




위의 그래프에서 볼 수 있듯이 true eigenvector로 간주될 수 있는 U( : ,1)과 각 단계의 eigenvector를 비교한 오차가 pseudo 오차 보다 더욱 크게 나오는 것

을 확인 할 수 있다. 그 이유는 다음을 생각해 보면 간단하다. 결국 k번째 iteration에서의 벡터는 첫번째 고유벡터로 수렴해 가게 되는데, true 오차의 경우 첫번째 벡터와 고유벡터와의 차이가 초기 추정 벡터에 따라서 상당히 차이가 날 수 밖에 없다. 반면에 pseudo 오차의 경우 바로 전단계의 벡터로부터 변하는 양을 나타내기 때문에 더 적게 나타난다. 그래프로부터 확인해 보면 그 오차간의 차이는 두 오더정도 차이가 나는 것을 볼 수 있다.

**3. Estimate the dominance ratio of this system using the pseudo error ratios of power iteration. Compare this estimate with the one determined by the eigenvalue ratios in A.1.**

다음과 같은 함수를 작성한다.

```
function [lamdak,xk] = hw7b3(A,S)
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
B=inv(A)*S;
[interU,interLAM]=eig(full(B));
refEigVec=(interU(:,1));

A=full(A);
lam=sort(max(interLAM),'descend');
ref_dominance_ratio=lam(2)/lam(1)
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
graphmat=zeros(1,3);

iepsilon=1.e-6;
oepsilon=1.e-5;

maxIter=1000;
n=length(A);

xk=ones(n,1);
xk_1=ones(n,1); %initial guess of eigenvector
lamdak=1;
lamdak_1=1; %initial guess of eigenvalue
xk_1=xk_1./lamdak_1;    %scaling
```

```
pseudoErrk_1=NaN;
pseudoErrk_2=NaN;

for (k=1:maxIter)
%이 위치에서 xk_1은 이미 스케일링 되어 있는 상태
%matrix-vector multiplication
    phi=xk_1;
    phid=xk_1;
%%%%해 벡터 초기화%%%%%%%%%%%%%%%%%%%%%%%%
    for (i=1:n)
        b(i)=xk_1(i)*S(i,i);%이미 스케일링 하면서 lamdak_1로 나누기가 포함되어 있음
    end
    for (j=1:maxIter) %inner iteration
        for (i=1:n)
            suml=0.0;
            sumu=0.0;
                for (m=1:i-1)
                    suml=suml+A(i,m)*phi(m);
                end
                for (m=i+1:n)
                    sumu=sumu+A(i,m)*phid(m);
                end
                phi(i)=(b(i)-suml-sumu)/A(i,i);
        end
        rho_tilda=norm(phi-phid)/norm(phi); %오차의 상대값
        if( rho_tilda < iepsilon)
            break;
        end
        phid=phi; %예전것에 현재 나온 것을 대입
    End
%matrix-vector multiplication
    xk=phi;
    lamdak=(xk'*xk)/(xk'*xk_1); %obtaining new eigenvalue

    xk=xk./lamdak;%scaling
    pseudoErrk=norm(xk-xk_1);
```

```
    if ( pseudoErrk/norm(xk)< oepsilon )
        break;
    end

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
    graphmat(k,1)=k;
    graphmat(k,2)=pseudoErrk/pseudoErrk_1;
    graphmat(k,3)=pseudoErrk/pseudoErrk_2;  %sign에 의한 효과를 줄이기 위해.. 즉,
rho^2를 구하려 한다.

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
    xk_1=xk; %스케일링 되어 있음
    lamdak_1=lamdak;
    pseudoErrk_2=pseudoErrk_1;
    pseudoErrk_1=pseudoErrk;

end
plot(graphmat(:,1),graphmat(:,2),graphmat(:,1),graphmat(:,3));
xlabel('k');
ylabel('pseudoErr(k)/pseudoErr(k-1)');
legend('pseudoErrk/pseudoErrk-1','psuudoErrk/pseudoErrk-2');

estimated_dominance_ratio=graphmat(k-1,2)
sign_free_dominance_ratio=sqrt(graphmat(k-1,3))
```

실행시켜 보면

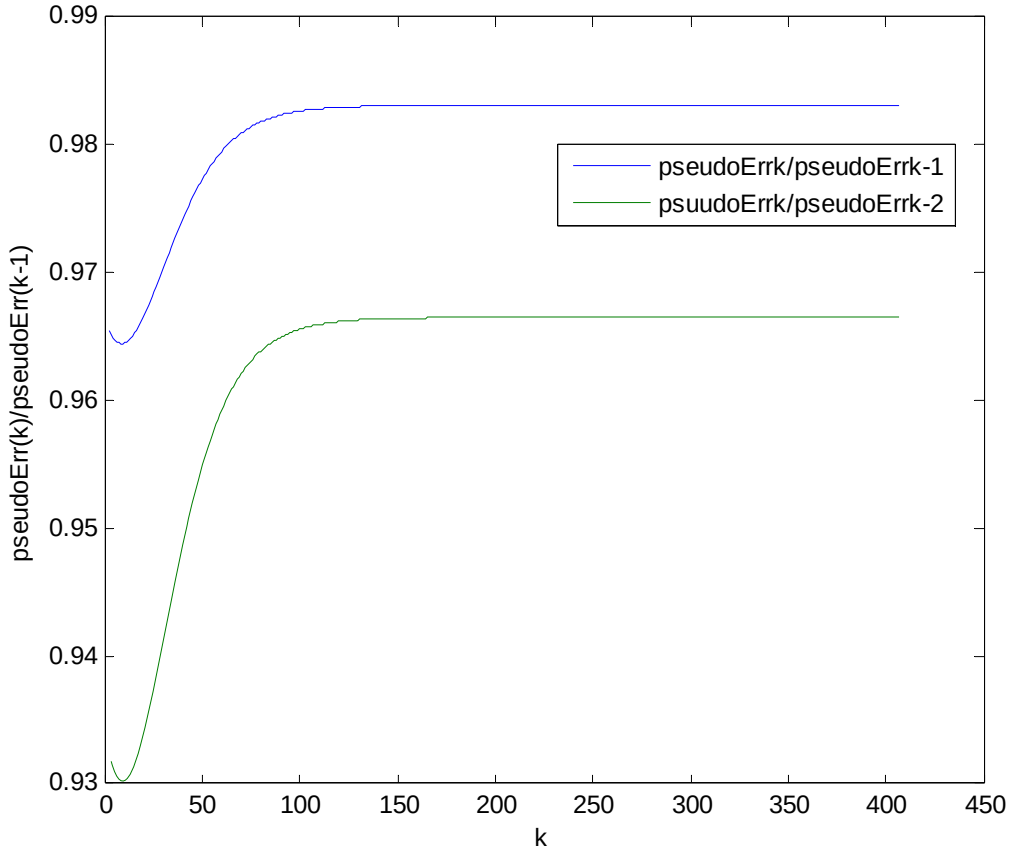
```
>> hw7b3(A,S);

ref_dominance_ratio =

    0.9932


estimated_dominance_ratio =
```

0.9831

sign_free_dominance_ratio =

0.9831

>>




위에서 보는 바와 같이 예상 $\sigma = \frac{\lambda_2}{\lambda_1}$ 즉, 0.9932로 나타나지 않고, 0.9831로 나타나는 것을 확인 할 수 있다. 이것은 고유벡터 내에 있는 부호의 영향일 수도 있기 때문에 그 영향을 없애고자 pseudo오차를 두개 차이로 해서 $\sigma^2$ 을 구하고 다시 제곱근을 구해보았지만 그래도 결과는 마찬가지로 나오는 것을 확인 할 수 있었다. 예상보다 수렴성은 좋지만 이렇게 나오는 원인을 크게

두 가지로 생각해 볼 수 있겠는데, 그 첫째는 B행렬의 고유값에서 3번째 고유값이 3개가 나오는 점을 생각해 볼 수 있다.

$$x^{(k)} = \frac{{\lambda_1}^k}{\prod {\lambda_1}^{(i)}}(c_1u_1 + c_2(\frac{\lambda_2}{\lambda_1})^k u_2 + c_3(\frac{\lambda_3}{\lambda_1})^k u_3 + c_4(\frac{\lambda_4}{\lambda_1})^k u_4 + c_5(\frac{\lambda_5}{\lambda_1})^k u_5 + ...)$$

$$\lambda_3 = \lambda_4 = \lambda_5, \sigma_2 = \frac{\lambda_2}{\lambda_1}, \sigma_3 = \frac{\lambda_3}{\lambda_1}, k >> 1$$

$$x^{(k)} = (c_1u_1 + c_2{\sigma_2}^k u_2 + (c_3u_3 + c_4u_4 + c_5u_5){\sigma_3}^k + ...)$$

위 와 같이 세번째 고유값에 가중치가 붙는 것을 확인할 수 있다. 실제로도 추정된 dominance ratio는 $\frac{\lambda_3}{\lambda_1}$에 더욱 가깝다는 것을 볼 수 있다.

```
>> L=sort(max(LAM),'descend');
>> L(3)/L(1)
ans =
    0.9831
>>
```

둘째로, B가 symmetric metric이기 때문에 나타나는 효과로 생각할 수 있다. 대칭행렬의 경우에 $u_i{}^t u_j = \delta_{ij}$

$$\lambda_1^{(k)} = \frac{< x^{(k)}, x^{(k)} >}{< x^{(k)}, \hat{x}^{(k-1)} >}$$

$$\lambda_1^{(k)} = \frac{< \frac{1}{\prod_{p=1}^{k-1} \lambda^p} \sum_i c_i {\lambda^k}_i u_i, \frac{1}{\prod_{p=1}^{k-1} \lambda^p} \sum_j c_j {\lambda^k}_j u_j >}{< \frac{1}{\prod_{p=1}^{k-1} \lambda^p} \sum_i c_i {\lambda^k}_i u_i, \frac{1}{\prod_{p=1}^{k-1} \lambda^p} \sum_j c_j {\lambda^{k-1}}_j u_j >}$$

$$\lambda_1^{(k)} = \frac{< \sum_i c_i {\lambda^k}_i u_i, \sum_j c_j {\lambda^k}_j u_j >}{< \sum_i c_i {\lambda^k}_i u_i, \sum_j c_j {\lambda^{k-1}}_j u_j >}$$

$$\lambda_1^{(k)} = \frac{\sum_i c^2{}_i \lambda_i^{2k}}{\sum_i c^2{}_i \lambda_i^{2k-1}}$$

$$\lambda_1^{(k)} = \frac{c_1^2 \lambda_1^{2k} + c_2^2 \lambda_2^{2k} + c_3^2 \lambda_3^{2k} + \ldots + c_n^2 \lambda_n^{2k}}{c_1^2 \lambda_1^{2k-1} + c_2^2 \lambda_2^{2k-1} + c_3^2 \lambda_3^{2k-1} + \ldots + c_n^2 \lambda_n^{2k-1}}$$

$$\lambda_1^{(k)} = \frac{\lambda_1^{2k}(c_1^2 + c_2^2(\frac{\lambda_2}{\lambda_1})^{2k} + \ldots)}{\lambda_1^{2k-1}(c_1^2 + c_2^2(\frac{\lambda_2}{\lambda_1})^{2k-1} + \ldots)}$$

$$\lambda_1^{(k)} = \frac{\lambda_1^{2k}(c_1^2 + c_2^2\sigma^{2k} + \ldots)}{\lambda_1^{2k-1}(c_1^2 + c_2^2\sigma^{2k-1} + \ldots)}$$

위와 같이 오차항이 dominance ratio의 제곱에 관련해서 변하는 점에서 그 근거를 찾아 볼 수 있다.

```
>> Dominance_ratio=L(2)/L(1)
Dominance_ratio =
    0.9932
>> Dominance_ratio^2
ans =
    0.9864
>>
```

0.9832와 비슷함을 알 수 있다.

**4. Discuss the advantage of the inverse power method over the regular (direct) power method of A.2.**

일반적인 power method로 eigenvalue와 eigenvector를 구해야 하는 경우 A행렬의 역행렬을 구해야 하는 과정이 필요하다. 그런데 역행렬 구하는 것 자체가 고유값이나 고유벡터를 구해야 하는 일만큼 어려운 일이다. 특히 A행렬의 사이즈가 큰 경우 총 연산 시간은 n^3으로 늘어나게 되므로 실제 문제의 풀이에는 별로 좋은 방법이 아니다. 반면에 inverse power method는 역행렬을 구할 필요 없이 일반적인 선형 시스템의 풀이법 (직접해법 또는 간접해법)을 도입하여 풀 수 있다. A 행렬을 쉽게 LU 분해 할 수 있는 경우에는 한번 LU 분해한 것에 대해 backsubstitution과 forward substitution을 한번씩 이용하여 정

해를 손쉽게 구해낼 수 있다. 아니면 간접해법을 사용해서 원하는 정확도 내에서 최소 연산으로 효율적으로 해를 구해낼 수 있는 장점이 있다.

## C. Physical Factors Affecting Dominance Ratio

**1. Increase and decrease the diffusion coefficient by a factor of 2 and 1/2.**

**2. Now increase and decrease the problem dimensions of the reference problem by a factor of 2 and 1/2. Keep the same number of meshes.**

위의 문제를 한꺼번에 풀기 위해서 다음과 같은 루틴을 작성해 본다.

```
function [dr,edr] = hw7c1c2()
%첫번째 리턴 값은 이론적인 dominance ratio를 담은 행렬
%두번째 리턴 값은 실제 pseudoerror vector비로 구한 dominace ratio 행렬
array_D=[0.5,1,2];
array_dimension=[50,100,200];
dr=zeros(length(array_D),length(array_dimension));
edr=dr;
for (i=1:length(array_D))
    for (j=1:length(array_dimension))
        [Aij,Sij]=lesc3d(4,array_dimension(j),array_D(i));
        [dummy1,dummy2,dr(i,j),edr(i,j)]=hw7b3(Aij,Sij);
    end
end
```

```
>> [dr,edr]=hw7c1c2()

dr =

    0.9865    0.9966    0.9991
    0.9737    0.9932    0.9983
    0.9502    0.9865    0.9966


edr =

    0.9670    0.9914    0.9978
    0.9369    0.9831    0.9957
```

0.8843 0.9670 0.9914

>>

표로 정리해 보면

| dr | a,b,c/2=50 | a,b,c/2=100 | a,b,c/2=200 |
|---|---|---|---|
| D=0.5 | 0.9865 | 0.9966 | 0.9991 |
| D=1 | 0.9737 | 0.9932 | 0.9983 |
| D=2 | 0.9502 | 0.9865 | 0.9966 |

| edr | a,b,c/2=50 | a,b,c/2=100 | a,b,c/2=200 |
|---|---|---|---|
| D=0.5 | 0.9670 | 0.9914 | 0.9978 |
| D=1 | 0.9369 | 0.9831 | 0.9957 |
| D=2 | 0.8843 | 0.9670 | 0.9914 |

**3. Discuss the dependence of the dominance ratio of the two physical factors.**

위의 결과를 살펴보면 확산계수가 증가할수록 수렴성이 좋아지는 것을 확인 할 수 있다. 또한 dimension의 크기가 증가할수록 수렴성이 나빠지는 것을 확인 할 수 있는데 이 효과의 크기는 확산계수의 변화에 의한 효과와 정확히 같은 것을 확인 할수 있다. 즉, 확산계수가 두배로 증가하는 동시에 dimension의 크기가 두배로 증가하면 dominance ratio의 크기변화는 없는 것을 알 수 있다.

주어진 확산방정식을 해석적으로 풀어보면

$$\frac{\partial^2\phi}{\partial x^2}+\frac{\partial^2\phi}{\partial y^2}+\frac{\partial^2\phi}{\partial z^2}+\frac{\frac{1}{\lambda}\sigma_s-\sigma_A}{D}\phi=0$$

$$B^2=\frac{\frac{1}{\lambda}\sigma_s-\sigma_A}{D}$$

$$\phi=X(x)Y(y)Z(z)$$

$$\frac{1}{X}\frac{d^2X}{dx^2}+\frac{1}{Y}\frac{d^2Y}{dy^2}+\frac{1}{Z}\frac{d^2Z}{dz^2}+B^2=0$$

$$B^2=B_x^{\,2}+B_y^{\,2}+B_z^{\,2}$$

$$\frac{d^2X}{dx^2} + B_x{}^2 X = 0$$

$$\frac{d^2Y}{dy^2} + B_y{}^2 Y = 0$$

$$\frac{d^2Z}{dz^2} + B_z{}^2 Z = 0$$

for given B.C.

$$\left.\frac{\partial\phi}{\partial x}\right|_{x=0} = 0, \;\; \phi(a,y,z) = 0, \;\; \left.\frac{\partial\phi}{\partial y}\right|_{y=0} = 0, \;\; \phi(x,b,z) = 0$$

$$\phi(x,y,0) = 0, \;\; \phi(x,y,c) = 0$$

$$X = A_x \cos B_x x$$

$$Y = A_y \cos B_y y$$

$$Z = A_z \sin B_z z$$

$$B_x a = \frac{n_x \pi}{2}, B_x a = \frac{n_x \pi}{2}, B_z c = n_z \pi$$

$$n_x, n_y = 1,3,5...$$

$$n_z = 1,2,3...$$

$$\therefore B^2 = (\frac{n_x \pi}{2a})^2 + (\frac{n_y \pi}{2b})^2 + (\frac{n_z \pi}{c})^2$$

$$\lambda = \frac{\sigma_S}{\sigma_A + DB^2} = \frac{\sigma_S}{\sigma_A + D((\frac{n_x \pi}{2a})^2 + (\frac{n_y \pi}{2b})^2 + (\frac{n_z \pi}{c})^2)}$$

실제로 계산해 보면

```
>> sigA=0.1;sigS=0.12;D=1;a=100;b=100;c=200;
>> nx=1;ny=1;nz=1;
>> sigS/(sigA+D*((nx*pi)^2/(2*a)^2+(ny*pi)^2/(2*b)^2+(nz*pi)^2/(c)^2))

ans =

    1.1912

>> L(1)

ans =
```

1.1913

>>
수치적으로 구한 값과 거의 일치하는 것을 확인 할 수 있다.
>> nx=1;ny=1;nz=2;
>> sigS/(sigA+D*((nx*pi)^2/(2*a)^2+(ny*pi)^2/(2*b)^2+(nz*pi)^2/(c)^2))

ans =

1.1825

>> L(2)

ans =

1.1831

>>
따라서 다음 식

$$\lambda = \frac{\sigma_S}{\sigma_A + DB^2} = \frac{\sigma_S}{\sigma_A + D((\frac{n_x\pi}{2a})^2 + (\frac{n_y\pi}{2b})^2 + (\frac{n_z\pi}{c})^2)}$$

의 유효함을 확인할 수 있고, 아울러

$$\frac{\lambda_2}{\lambda_1} = \frac{\sigma_A + D((\frac{\pi}{2a})^2 + (\frac{\pi}{2b})^2 + (\frac{\pi}{c})^2)|_{n=1}}{\sigma_A + D((\frac{\pi}{2a})^2 + (\frac{\pi}{2b})^2 + (\frac{2\pi}{c})^2)|_{n=2}}$$

을 얻는다. 위 식은 dominace ratio 와 같은데 여기서 D가 클수록 분자, 분모의 두번째 term이 우세하여 dominance ratio가 작아짐을 알 수 있다. 반면에 D가 작아지면 첫번째 term이 우세하여 dominace ratio가 1에 가까워 지게 된다. 마찬가지로 dimension이 커지면 두번째 term이 작아지고 첫번째 term의 비중이 커지게 되어 dominace ratio 가 1에 가까워진다.