CS450_HW3_XiaowenLin

xlin11@illinois.edu

4.3 p208

(a) $f(x) \approx |A - \lambda E| = | \begin{bmatrix} 1 & 4 \\ 1 & 1 \end{bmatrix}$-$\begin{bmatrix} \lambda & 0 \\ 0 & \lambda \end{bmatrix} | = (1-\lambda) \times (1-\lambda) - 4 \times 1 =$

$\lambda^2 - 2\lambda - 3$

(b) $f(x) = (\lambda - 3) \times (\lambda + 1) = 0, \lambda = 3 or - 1$

(c) 3 and -1 are the eigenvalues of A

(d)

$x = \begin{bmatrix} x_1 \\ x_2 \end{bmatrix}$

For $\lambda = 3$, $A - \lambda E = \begin{bmatrix} 1-3 & 4 \\ 1 & 1-3 \end{bmatrix} = \begin{bmatrix} -2 & 4 \\ 1 & -2 \end{bmatrix} \rightarrow \begin{bmatrix} 1 & -2 \\ 0 & 0 \end{bmatrix}$

$x_1 - 2x_2 = 0$, so $x_1 = 2x_2 . x = \begin{bmatrix} 0.5 \\ 1 \end{bmatrix}$

For $\lambda = -1$, $A - \lambda E = \begin{bmatrix} 1+1 & 4 \\ 1 & 1+1 \end{bmatrix} = \begin{bmatrix} 2 & 4 \\ 1 & 2 \end{bmatrix} \rightarrow \begin{bmatrix} 1 & 2 \\ 0 & 0 \end{bmatrix}$

$x_1 + 2x_2 = 0$, so $x_1 = -2x_2 . x = \begin{bmatrix} -0.5 \\ 1 \end{bmatrix}$

(e)$x_1 = Ax_0 = \begin{bmatrix} 1 & 4 \\ 1 & 1 \end{bmatrix} \begin{bmatrix} 1 \\ 1 \end{bmatrix} = \begin{bmatrix} 5 \\ 2 \end{bmatrix}$

(f)based the Normalized Power Iteration method,

$x_0 = \begin{bmatrix} 1 \\ 1 \end{bmatrix}$

for k = 1, 2, 3, ...30

$y_k = Ax_{k-1}$

$x_k = y_k / \|y_k\|_\infty$

end

We obtain the following sequence

k	xk[1]	xk[2]	xk[1]/xk[2]
1	5.000000000000	2.000000000000	2.500000000000
2	2.600000000000	1.400000000000	1.857142857143
3	3.153846153846	1.538461538462	2.050000000000
4	2.951219512195	1.487804878049	1.983606557377
5	3.016528925620	1.504132231405	2.005494505495
6	2.994520547945	1.498630136986	1.998171846435
7	3.001829826167	1.500457456542	2.000609756098
8	2.999390429747	1.499847607437	1.999796789270
9	3.000203231379	1.500050807845	2.000067741498
10	2.999932260796	1.499983065199	1.999977420010
11	3.000022580245	1.500005645061	2.000007526720
12	2.999992473308	1.499998118327	1.999997491100
13	3.000002508903	1.500000627226	2.000000836301
14	2.999999163700	1.499999790925	1.999999721233
15	3.000000278767	1.500000069692	2.000000092922
16	2.999999907078	1.499999976769	1.999999969026
17	3.000000030974	1.500000007744	2.000000010325
18	2.999999989675	1.499999997419	1.999999996558
19	3.000000003442	1.500000000860	2.000000001147
20	2.999999998853	1.499999999713	1.999999999618
21	3.000000000382	1.500000000096	2.000000000127
22	2.999999999873	1.499999999968	1.999999999958
23	3.000000000042	1.500000000011	2.000000000014
24	2.999999999986	1.499999999996	1.999999999995
25	3.000000000005	1.500000000001	2.000000000002
26	2.999999999998	1.500000000000	1.999999999999
27	3.000000000001	1.500000000000	2.000000000000
28	3.000000000000	1.500000000000	2.000000000000
29	3.000000000000	1.500000000000	2.000000000000
30	3.000000000000	1.500000000000	2.000000000000

A will ultimately converge into $\begin{bmatrix} 3 \\ 1.5 \end{bmatrix}$

(g) Repeating the process above, using Rayleigh quotient, the value of the Rayleigh quotient at each iteration is shown below:

k	lamda	xk[1]	xk[2]
1	3.500000000000	1.428571428571	0.571428571429
2	2.724137931034	1.363471971067	0.734177215190
3	3.087155963303	1.392926331855	0.679476259441
4	2.970206631427	1.384022015884	0.697730107181
5	3.009857314018	1.387089821555	0.691644787734
6	2.996705721104	1.386078367069	0.693673254150
7	3.001097159168	1.386416754606	0.692997097700
8	2.999634176098	1.386304096193	0.693222483220
9	3.000121929741	1.386341664264	0.693147354712
10	2.999959355468	1.386329143270	0.693172397548
11	3.000013548035	1.386333317123	0.693164049936
12	2.999995483973	1.386331925860	0.693166832473
13	3.000001505341	1.386332389617	0.693165904961
14	2.999999498220	1.386332235031	0.693166214132
15	3.000000167260	1.386332286560	0.693166111075
16	2.999999944247	1.386332269384	0.693166145427
17	3.000000018584	1.386332275109	0.693166133976
18	2.999999993805	1.386332273201	0.693166137793
19	3.000000002065	1.386332273837	0.693166136521
20	2.999999999312	1.386332273625	0.693166136945
21	3.000000000229	1.386332273695	0.693166136804
22	2.999999999924	1.386332273672	0.693166136851
23	3.000000000025	1.386332273680	0.693166136835
24	2.999999999992	1.386332273677	0.693166136840
25	3.000000000003	1.386332273678	0.693166136838
26	2.999999999999	1.386332273678	0.693166136839
27	3.000000000000	1.386332273678	0.693166136839
28	3.000000000000	1.386332273678	0.693166136839
29	3.000000000000	1.386332273678	0.693166136839
30	3.000000000000	1.386332273678	0.693166136839

So eigenvalue converges to 3.

(h) Using inverse iteration ultimately converges into 2.

(i) Using inverse iteration with shift=2, the eigenvalue turns out to be 3.

(j) Using QR iteration, A is converged into triangular. QR iteration is implemented in two-stages. For non-symmetric, the final matrix will be triangular.

4.24 p209

(a)

P and Q are full ranked $n \times n$ matrixes.

According to the probelm, $P_1AQ_1 = \begin{bmatrix} 1 & o \\ o & O \end{bmatrix}$

If two nonzero vectors multiply, the rank of the result will be one. So $P_2u_1v_1^TQ_2 = \begin{bmatrix} 1 & o \\ o & O \end{bmatrix}$.

So $P_1AQ_1 = P_2u_1v_1^TQ_2$, assume $P = P_1^{-1}P_2, Q = Q_1^{-1}Q_2$

Thus, $A = Pu_1v_1^TQ$

$Pu_1$is still a vector, assume $u = Pu_1$.

$v_1^TQ$is still a vector, assume $v^T = v_1^TQ$.

So $A = uv^T$.

(b)

$Au = uv^Tu = u(v^Tu) = (v^Tu)u = (v^Tu)^Tu = u^Tvu$. So u^Tv is a eigenvalue.

(c)

Because A has a rank of one, A has only one nonzero eigenvalue. So the other eigenvalues are zeros.

(d)

Assume $x = a_1v_1 + a_2v_2 + \cdots + a_nv_n$,

$v_1, v_2, \cdots v_n$ are eigenvectors of A. $v_1$is the eigenvector for the nonzero eigenvalue.

$Ax = a_1Av_1 + a_2Av_2 + \cdots + a_nAv_n = a_1\lambda_1v_1 + a_2\lambda_2v_2 + \cdots + a_n\lambda_nv_n = a_1\lambda_1v_1$

So it needs only one time of iteration.

4.31 p210

(a)According to Rayleight iteration,

$\lambda = \frac{x^TQx}{x^Tx}$

$\lambda^T = \frac{x^TQ^Tx}{x^Tx}$

$\lambda^2 = \lambda \times \lambda^T = \frac{x^TQxx^TQ^Tx}{x^Txx^Tx} = \frac{x^TQQ^Tx}{x^Tx} = \frac{x^Tx}{x^Tx} = 1$

So $|\lambda| = 1$

(b)

$(QQ^T)x = \lambda x \Rightarrow \lambda = \frac{x^TQQ^Tx}{x^Tx} = 1$

$Q = USV, Q^T = V^TSU^T$

$QQ^T = USVV^TSU^T = USSU^T = E$

$\because UU^T = E$

$\therefore (QQ^T)$ and$(SS)$are similar. So (SS) has the same eigenvalues as (QQ^T), 1.

$\because S = diag(a_1, a_{2,}a_3 \dots a_r \dots a_n)$

$\therefore SS = diag(a_1^2, a_2^2, a_3^2, \dots a_r^2, \dots a_n^2)$ has eigenvalues, 1.

For diagonal matrix, entries are equal to eigenvalues.

So singular values of rotogonal matrix are ± 1.

```
function cp04_02

A = [2 3 2; 10 3 4; 3 6 1];
x0 = [0 0 1]';

v = x0;
m = 0;
l = 0;
for(k=1:5000)
    y = A*v;
    m = norm(y,Inf);
    v = y/m;

    if(abs(m - l)<1.0e-6)
        l = m;
        break;
    else
        l = m;
    end

end

x1 = v;
l1 = l;
disp('(a)');
disp('eigenvalue');
fprintf('%d\n',round(l1));
disp('eigenvector');
disp(x1);

%(b)

[q,r] = qr(x1);
H = q;
seudoB = H*A*inv(H);
B = seudoB(2:3,2:3);


v = x0(2:3);
m = 0;
l = 0;
for(k=1:5000)
    y = B*v;
    m = norm(y,Inf);
    v = y/m;

    if(abs(m - l)<1.0e-5)
        [trash,I] = max(abs(y'),[],2);
        l = y(I);
        s = k;
        v = y(I)/abs(y(I))*y/m;
```

```
        break;
    else
        [trash,I] = max(abs(y'),[],2);
        l = y(I);
        v = y(I)/abs(y(I))*y/m;
    end


end

x2 = v;
l2 = l;
disp('(b)');
disp('eigenvalue');
fprintf('%d\n',round(l2));
disp('eigenvector');
disp(x2);

%(c)
disp('(c)')
[v1_c,l1_c] = eig(A)
[v2_c,l2_c] = eig(B)
disp('The eigenvalues are the same with results from (a) and (b).')
disp('The eigenvector corresponding to  (a) is');
disp(v1_c(:,1));
disp('the normalized result is');
disp(v1_c(:,1)/norm(v1_c(:,1),Inf));
disp('which is the same as result of (a).')

disp('The eigenvector corresponding to  (b) is');
disp(v2_c(:,1));
disp('the normalized result is');
disp(v2_c(:,1)/norm(v2_c(:,1),Inf));
disp('which is the same as result of (b).')

disp('The results are the same, while with library ruotine, the eigenvectors are not
normalized by norm(x,Inf).')
```

```
>> cp04_02
(a)
eigenvalue
11
eigenvector
    0.5000
    1.0000
    0.7500

(b)
eigenvalue
-3
eigenvector
   -0.7183
    1.0000

(c)

v1_c =

    0.3714    0.1826   -0.0000
    0.7428    0.3651   -0.5547
    0.5571   -0.9129    0.8321


l1_c =

   11.0000         0         0
         0   -2.0000         0
         0         0   -3.0000


v2_c =

   -0.5834    0.3633
    0.8122   -0.9317


l2_c =

   -3.0000         0
         0   -2.0000

The eigenvalues are the same with results from (a) and (b).
The eigenvector corresponding to  (a) is
    0.3714
    0.7428
    0.5571

the normalized result is
    0.5000
```

```
    1.0000
    0.7500

which is the same as result of (a).
The eigenvector corresponding to  (b) is
   -0.5834
    0.8122

the normalized result is
   -0.7183
    1.0000

which is the same as result of (b).
The results are the same, while with library ruotine, the eigenvectors are not normalized
by norm(x,Inf).
>>
```

```
function cp04_03
%(a)
A = [6 2 1; 2 3 1; 1 1 1];
x0 = [0 0 1]';
u = 2;

v = x0;
m = 0;
l = 0;
for(k=1:1000)
    y = (A-u*eye(3,3))\v;
    m = norm(y,Inf);
    v = y/m;

    if(abs(m - l)<1.0e-5)
        l = 1/m + u;
        break;
    else
        l = m;
    end
end

disp('(a)');
disp('eigenvector');
disp(v);
disp('eigenvalue');
disp(l);
%(b)
disp('(b)');
[v_b,l_b]=eig(A)
disp('Normalized eigenvector is');
disp(v_b(:,2)/norm(v_b(:,2),Inf));
disp('The results obtained from (a) is the second colume in eigenvectors and
eigenvalues.');
disp('Because the second dominant eigenvalue is most close to 2.');
```

```
>> cp04_03
(a)
eigenvector
   -0.6069
    1.0000
    0.3469

eigenvalue
    2.1331

(b)

v_b =

   -0.0432   -0.4974   -0.8664
   -0.3507    0.8196   -0.4531
    0.9355    0.2843   -0.2098


l_b =

    0.5789         0         0
         0    2.1331         0
         0         0    7.2880

Normalized eigenvector is
   -0.6069
    1.0000
    0.3469

The results obtained from (a) is the second colume in eigenvectors and eigenvalues.
Because the second dominant eigenvalue is most close to 2.
>>
```

```
function cp04_04

A = [6 2 1; 2 3 1; 1 1 1];
x0 = [0 0 1]';
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
v = x0;
m = 0;
l = 0;

for(k=1:1000)
    y = A*v;
    m = (y'*v)/(v'*v);
    v = y/m;

    if(abs(m - l)<1.0e-6)
        l = m;
        v = y/m/norm(y/m,Inf);
        break;
    else
        l = m;
    end
end
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
[v_lib,l_lib] = eig(A);
disp('using A and x0 used in cp04_03');
disp('According to the program for 4.4, eigenvector is');
disp(v);
disp('eigenvalue is');
disp(l);
disp('With library method, the eigenvalue is');
disp(l_lib(3,3));
disp('With library method, the eigenvector is');
disp(v_lib(:,3));
disp('The normalized positive result is');
disp(abs(v_lib(:,3)/norm(v_lib(:,3),Inf)));
disp('These values are the same as those obtained from library routine. This program
works.');
```

```
>> cp04_04
using A and x0 used in cp04_03
According to the program for 4.4, eigenvector is
    1.0000
    0.5229
    0.2422

eigenvalue is
    7.2880

With library method, the eigenvalue is
    7.2880

With library method, the eigenvector is
   -0.8664
   -0.4531
   -0.2098

The normalized positive result is
    1.0000
    0.5229
    0.2422

These values are the same as those obtained from library routine. This program works.
>>
```