VLSI DPS HW#1

電機四 **4109064119** 陳柏淳

1. Last square optimization problem

```
% Problem 1. Last square optimization problem
A = [    15     -13     20    -8;
          -5     -15     -4     -4;
         -17     16       -2     9;
          10    -19     -14    -15;
          -7      8       -7     15;
          14     10      -8     -17;
          -5     -3       16     -2;
          13     -5      -10    -19];
b = [13; 10; -15; 9; 3; 18; 3; 20];
```

a) Pseudo inverse

- Code

```
% (a) Pseudo inverse
x_pseudo    = pinv(A) * b;
```

- Result

x1 = 0.4638

x2 = -0.1005

x3 = -0.0716

x4 = -0.4137

(b) QR decomposition

- Code

```
% (b) QR decomposition
[Q, R] = qr(A);
y = Q' * b;
R1 = R(1:4, 1:4);
y1 = y(1:4);
x_QR = inv(R1) * y1; % R * x_b = Q' * b
```

- Result

x1 = 0.4638

x2 = -0.1005

x3 = -0.0716

x4 = -0.4137

(c) Compare if a) and b) yield the same result?

Yes, both a) and b) yield the same result.

2. Eigen decomposition

- Code

```
% Problem 2. Eigen decomposition
M = [ --2    16    -6     -16    3     15    -6    -19;
        16   -17    10      -2    7     8     3      5;
        --6   10    15      -1   -15   -18    9     -8;
       --16   -2     -1     9     0     0     0      18;
         3     7    -15     0     14    19   -12     11;
        15     8    -18     0     19    10   -8     -17;
        -6     3     9      0    -12    -8    15     20;
       -19     5     -8     18    11   -17    20     20];

% Use iterative
convergence_threshold = 1e-4;
[D_iter, V_iter, sweeps] = eig_iterative(M, convergence_threshold);

% Use eig()
[V, D] = eig(M);

disp('Eigenvalue matrix D Using Iterative:');
disp(D_iter);
disp('Eigenvalue matrix V Using Iterative:');
disp(V_iter);

disp('The numbers of sweeps:');
disp(sweeps);

disp('Eigenvalue matrix D Using eig():');
disp(D);
disp('Eigenvalue matrix V Using eig():');
disp(V);

function [D_iter, V_iter, sweeps] = eig_iterative(M, Convergence_Threshold)
    M_tiled = M;
    convergence = false;
    sweeps = 0;

    % Find D
    while ~convergence
        [Q, R] = Given_QR(M_tiled);
        M_new = R * Q;

        convergence = (det(Q*M_new*Q') - (det(diag(diag(Q*M_new*Q'))))) /
                      (det(M_tiled)) < Convergence_Threshold;

        % convergence = (det((Q*M_new*Q') - diag(diag(Q*M_new*Q')))) /
                        (det(M_tiled)) < Convergence_Threshold;

```

```
47              M_tiled = M_new;
48              sweeps = sweeps + 1;
49          end
50
51          % Find V
52          D_iter = M_tiled;
53          n = size(D_iter, 1);
54          V_iter = zeros(n);
55          % Compute eigenvectors for each approximate eigenvalue from D_iter
56          for i = 1:n
57              lambda = D_iter(i, i);
58              v = rand(n, 1);
59              for k = 1:10 % 10 iterations for refinement
60                  v = (M - lambda * eye(n)) \ v; % inverse iteration
61                  v = v / norm(v); % Normalize
62              end
63              V_iter(:, i) = v;
64          end
65      end
66
67      function [Q, R] = Given_QR(M)
68          [n, m] = size(M);
69          Q = eye(m);
70          R = M;
71          for i = 1 : n-1
72              for j = i+1 : m
73                  x = R(:, i);
74                  q_t = Givens(x, i, j);
75                  Q = Q * q_t';
76                  R = q_t * R;
77              end
78          end
79      end
80
81      function R = Givens(x, i, j)
82          r = sqrt(x(i)^2 + x(j)^2);
83          cost = x(i) / r;
84          sint = x(j) / r;
85
86          R = eye(length(x));
87          R(i, i) = cost;
88          R(i, j) = sint;
89          R(j, i) = -sint;
90          R(j, j) = cost;
91      end
92
```

- Result

Eigenvector matrix D Using Iterative:

67.1862	0.0000	-0.0000	0.0000	0.0000	0.0000	-0.0000	-0.0000
0.0000	46.8120	-0.2845	-0.0000	0.0000	-0.0000	-0.0000	0.0000
-0.0000	-0.2845	-36.9128	-0.0000	0.0000	-0.0000	-0.0000	0.0000
-0.0000	-0.0000	-0.0000	-26.0043	0.0000	0.0000	0.0000	-0.0000
-0.0000	0.0000	0.0000	0.0000	16.4684	0.0000	-0.0000	0.0000
-0.0000	0.0000	0.0000	0.0000	0.0000	-11.0351	-0.0000	-0.0000
-0.0000	0.0000	0.0000	0.0000	0.0000	-0.0000	6.0581	0.0000
-0.0000	0.0000	0.0000	0.0000	0.0000	-0.0000	-0.0000	1.4275

Eigenvector matrix V Using Iterative:

-0.3839	-0.1511	-0.4687	0.3638	0.3729	0.5802	-0.0097	0.0447
-0.0728	-0.0006	0.7047	-0.1987	0.4103	0.3837	0.3740	0.0562
0.2682	-0.5003	-0.3288	-0.1075	0.1175	-0.2141	0.7059	-0.0092
0.2528	0.3313	-0.0014	0.3618	-0.3726	0.2220	0.3308	0.6308
-0.2731	0.5323	0.0063	0.4116	0.1844	-0.3783	0.3964	-0.3720
-0.4867	0.2184	-0.2156	-0.5016	0.1834	-0.2840	0.0533	0.5479
0.4086	-0.0333	0.0664	0.3131	0.6480	-0.3281	-0.3033	0.3315
0.4827	0.5337	-0.3531	-0.4083	0.2290	0.2993	0.0181	-0.2199

Sweep times: 38

Eigenvector matrix D Using eig():

-36.9137	0	0	0	0	0	0	0
0	-26.0043	0	0	0	0	0	0
0	0	-11.0351	0	0	0	0	0
0	0	0	1.4275	0	0	0	0
0	0	0	0	6.0581	0	0	0
0	0	0	0	0	16.4684	0	0
0	0	0	0	0	0	46.8129	0
0	0	0	0	0	0	0	67.1862

Eigenvector matrix V Using eig():

-0.4687	0.3638	-0.5802	-0.0447	0.0097	-0.3729	-0.1511	-0.3839
0.7047	-0.1987	-0.3837	-0.0562	-0.3740	-0.4103	-0.0006	-0.0728
-0.3288	-0.1075	0.2141	0.0092	-0.7059	-0.1175	-0.5003	0.2682
-0.0014	0.3618	-0.2220	-0.6308	-0.3308	0.3726	0.3313	0.2528
0.0063	0.4116	0.3783	0.3720	-0.3964	-0.1844	0.5323	-0.2731
-0.2156	-0.5016	0.2840	-0.5479	-0.0533	-0.1834	0.2184	-0.4867
0.0664	0.3131	0.3281	-0.3315	0.3033	-0.6480	-0.0333	0.4086
-0.3531	-0.4083	-0.2993	0.2199	-0.0181	-0.2290	0.5337	0.4827

- Verification

利用迭代方法計算 eigen value 主要取決於迭帶次數與收斂條件，由於題目給的收斂條件過於寬鬆，導致達到收斂條件後後的結果與 eig()的結果相去甚遠。因此我將收斂公式從 det($Q\widetilde{M}Q^t$ – diag($Q\widetilde{M}Q^t$))) / det(M)修改為(det($Q\widetilde{M}Q^t$) – det(diag($Q\widetilde{M}Q^t$))) / det(M)，在 sweeps 38 次後就可以滿足收斂條件，達到與 eig()相同的結果，而 eig()函式在計算過程中會將 eigen value 由小至大排列，因此顯示矩陣內的元素位置會略有不同。

如果使用原始題目給的收斂條件去迭代的話，僅 sweep 3 次就會達到收斂條件了，此結果與 eig()有很大的落差，使用原始收斂條件(det($Q\widetilde{M}Q^t$ – diag($Q\widetilde{M}Q^t$))) / det(M)) (code:44 行)得到的結果如下。

Eigenvector matrix D Using Iterative (original converged condition):

62.5493	20.4652	-1.1407	-2.8255	-0.6693	-0.7815	-0.2282	0.0021
20.4652	-32.0278	-6.1344	-1.5885	1.7020	1.5989	0.2942	-0.0024
-1.1407	-6.1344	46.5752	1.5508	-0.8664	-1.4046	-0.1330	0.0028
-2.8255	-1.5885	1.5508	-24.6506	4.3164	5.3311	0.5658	-0.0050
-0.6693	1.7020	-0.8664	4.3164	7.7065	11.8474	2.1704	-0.0290
-0.7815	1.5989	-1.4046	5.3311	11.8474	-3.8638	-0.1568	0.0256
-0.2282	0.2942	-0.1330	0.5658	2.1704	-0.1568	6.2839	0.0059
0.0021	-0.0024	0.0028	-0.0050	-0.0290	0.0256	0.0059	1.4274

Eigenvector matrix V Using Iterative (original converged condition):

0.3839	0.4302	-0.1511	0.3638	-0.0097	0.0377	-0.0097	0.0447
0.0728	-0.6814	-0.0006	-0.1987	0.3740	0.0533	0.3740	0.0562
-0.2682	0.3379	-0.5003	-0.1075	0.7059	-0.0034	0.7059	-0.0092
-0.2528	-0.0346	0.3313	0.3618	0.3308	0.6296	0.3308	0.6308
0.2731	-0.0472	0.5323	0.4116	0.3964	-0.3657	0.3964	-0.3720
0.4867	0.2645	0.2184	-0.5016	0.0533	0.5515	0.0533	0.5479
-0.4086	-0.0972	-0.0333	0.3131	-0.3033	0.3341	-0.3033	0.3315
-0.4827	0.3920	0.5337	-0.4083	0.0181	-0.2234	0.0181	-0.2199

Sweep times: 3