Solution of EECS 600 HW #1

Yi Yang

1. **Proof:** The collection of infinite number of sequences which converges to zero is a vector space. In order to prove this statement, we should see whether it satisfies 5 addition and 5 scalar multiplication conditions:

   Addition:
   Closure: $\forall \{x_n\}_{n=1}^{\infty}, \{y_n\}_{n=1}^{\infty} \in$ X, and we want to see $\{x_n + y_n\}_{n=1}^{\infty} \in$ X .
   $\forall\ \epsilon = \epsilon_1 + \epsilon_2 > 0, \epsilon_1 > 0, \epsilon_2 > 0$, there exists $N > 0$, such that $|x_n + y_n| < \epsilon$ for all $n > N$.
   From the convergence of sequences $\{x_n\}$ and $\{y_n\}$, we know the following statements are true:
   $\forall \epsilon_1, \epsilon_2 > 0$, there exists N_1 and N_2 that are both greater than 0, such that $|x_n| < \epsilon_1$ for all $n > N_1$ and $|y_n| < \epsilon_2$ for all $n > N_2$, we choose $N = \max\{N_1, N_2\}$ so that we know the convergence statement satisfies for the sum of two sequences .
   Commutativity, Associativity: can be deduced from commutativity and associativity of the additions of real numbers .
   Additive identity: $\exists\ \{0\}_{n=1}^{\infty}$ such that $\forall\ \{x_n\}_{n=1}^{\infty} \in$ X, $\{0 + x_n\}_{n=1}^{\infty} = \{x_n\}_{n=1}^{\infty} \in$ X .
   Additive inverse: additive inverse of two infinite sequences of real numbers is a infinite sequences filled with 0 which satisfies our condition .
   Scalar Multiplication:
   Closure: $\forall\ \epsilon = \frac{\epsilon_1}{|a|} > 0,\ a \in \mathbb{R}$, there exists $N > 0$, such that $|\frac{x_n}{a}| < \frac{\epsilon_1}{|a|}$ for all $n > N$.
   Associativity, First Distributivity, Second Distributivity and application of multiplicative identity can be deduced from properties of real numbers and closure conditions.

3(a) **Solution:** The matlab code for both versions is attached below.

```
%Problem 3
A = [8 4 2 1; 4 8 4 2; 2 4 8 4; 1 2 4 8];
m = 4;
%A = rand(m); AA = A'*A;
A1 = A;
t = cputime;
for k = 1:m-2
    x = A1(k+1:m,k);
    v = x;
    v(1) = v(1) + sign(x(1))*norm(x,2);
    v = v/norm(v,2);
    A1(k+1:m,k:m) = A1(k+1:m,k:m) - 2*v*(v'*A1(k+1:m,k:m));
    A1(1:m,k+1:m) = A1(1:m,k+1:m) - 2*(A1(1:m,k+1:m)*v)*v';
end
```

```
dt = cputime - t;
A2 = A;
t1 = cputime;
for k = 1:m-2
    x = A2(k+1:m, k);
    v = x;
    v(1) = v(1) + sign(x(1))*norm(x,2);
    v = v/norm(v,2);
    A2(k+1,k) = -sign(x(1))*norm(x,2);
    A2(k,k+1) = -sign(x(1))*norm(x,2);
    A2(k+2:m,k) = 0; A2(k,k+2:m) = 0;
    B = 2*(A2(k+1:m,k+1:m)*v - (v'*A2(k+1:m,k+1:m)*v)*v)*v';
    for j = 1:m-k
        A2(k+j,k+1:k+j) = A2(k+j,k+1:k+j) - B(j,1:j) - B(1:j,j)';
        A2(k+1:k+j,k+j) = A2(k+j,k+1:k+j);
    end
    %A2(k+1:m,k+1:m) = A2(k+1:m,k+1:m) - B - B';
end
dt1 = cputime - t1;
```

For the general case, we can get A is:

$$A = \begin{pmatrix} 8.000000000000000 & -4.582575694955841 & 0 & 0 \\ -4.582575694955841 & 12.571428571428569 & 3.886134431067267 & -0.000000000000001 \\ 0 & 3.886134431067267 & 7.666409266409263 & -1.188992612745301 \\ 0 & -0.000000000000001 & -1.188992612745300 & 3.762162162162162 \end{pmatrix},$$

For the real symmetric case:

$$A = \begin{pmatrix} 8.000000000000000 & -4.582575694955840 & 0 & 0 \\ -4.582575694955840 & 12.571428571428562 & 3.886134431067272 & 0 \\ 0 & 3.886134431067272 & 7.666409266409261 & -1.188992612745300 \\ 0 & 0 & -1.188992612745300 & 3.762162162162163 \end{pmatrix}.$$

3(b) The cputime for the two methods are listed below.

m	general algorithm	symmetric
100	0	0.218401400000005
200	0.702004500000001	1.310408400000000
300	0.826805299999997	3.915625100000000
400	2.090413400000003	5.007632099999995
500	4.040425900000003	8.751656100000005
600	7.971651099999988	14.086890300000007
700	14.367692099999999	21.746539400000003
800	24.133354699999984	31.746203500000007
900	36.410633399999995	43.664679899999982
1000	52.853138800000011	59.997984599999995

Table 1: CPU time.

From this table, we can see the general case will cost less cputime to run the code. This is because Matlab will spend extra time to run the inner loops.

4(a) **Solution:** Matlab code is listed below.

```
A = [ 8, 4, 2, 1; 4, 8, 4, 2; 2, 4, 8, 4; 1, 2, 4, 8];
N=10; v = [.5; .5; .5; .5]; eig(A)
lambda1 = zeros(N,1);
for k = 1:N
    w = A*v; v = w/norm(w,2);
    lambda1(k) = v'*A*v;
end
mu = 0; v = [.5; .5; .5; .5]; lambda2 = zeros(N,1);
for k = 1:N
    w = (A-mu*eye(4))\v; v = w/norm(w,2);
    lambda2(k) = v'*A*v;
end
v = [.5; .5; .5; .5]; lambda3 = zeros(N+1,1);
lambda3(1) = v'*A*v;
for k = 2:N+1
    w = (A-lambda3(k-1)*eye(4))\v; v = w/norm(w,2);
    lambda3(k) = v'*A*v;
end
disp([(1:N)', lambda1, lambda2, lambda3(2:N+1)])
for k = 1:N
    [Q, R] = qr(A); A = R*Q;
    E = max(max(abs(A-diag(diag(A)))));
    disp(E);
end
```

The results at each step are

Iteration Number	Power method	inverse iteration	Rayleigh quotient
1	16.672131147540984	14.399999999999999	16.684615384615384
2	16.683819628647210	7.135135135135138	16.684658438426489
3	16.684602322430148	4.554938956714761	16.684658438426489
4	16.684654684513276	4.331664534989951	16.684658438426489
5	16.684658187307214	4.316434838256361	16.684658438426489
6	16.684658421627791	4.315414702695186	16.684658438426489
7	16.684658437302737	4.315346454398972	16.684658438426489
8	16.684658438351320	4.315341888880738	16.684658438426489
9	16.684658438421469	4.315341583468829	16.684658438426489
10	16.684658438426155	4.315341563038205	16.684658438426489

4(b) **Solution:** The maximum magnitude of the off-diagonal elements at each step is

Iteration Number	maximum magnitude of off-diagonal element
1	4.289906995109921
2	2.659558449077073
3	1.363329400421120
4	0.727713493424353
5	0.394169759053324
6	0.252862471417952
7	0.178747433831389
8	0.125365640161673
9	0.087544529634242
10	0.060995911929791

and the final matrix is

$$A^{(10)} = \begin{pmatrix} 16.684650870697748 & 0.008106965374877 & 0.000018128314708 & 0.000000256400932 \\ 0.008106965374876 & 7.999920721769701 & 0.017888947809789 & 0.000253015404780 \\ 0.000018128314708 & 0.017888947809787 & 4.312593875744196 & 0.060995911929791 \\ 0.000000256400932 & 0.000253015404780 & 0.060995911929791 & 3.002834531788369 \end{pmatrix}.$$