

Notes on PCA (EOF) Computation

Ruixin Yang

This document describes the details of PCA computation with real examples by Matlab. Matlab commands are in regular (Arial) font.

Step 1: Define a 5X3 (rowXcol) matrix.

```
D = [1 2 4 4 3; 3 4 6 6 2; 5 6 7 8 5];
```

```
D = D';
```

```
D =
```

```
    1    3    5
    2    4    6
    4    6    7
    4    6    8
    3    2    5
```

```
%%*** compute the mean and take the anomaly only.
```

```
B =mean(D,2);
```

```
[m,n] = size(D)
```

```
for i = 1:m
```

```
    D(i,:) = D(i, :)-B(i);
```

```
end
```

```
B =
```

```
    3.0000
    4.0000
    5.6667
    6.0000
    3.3333
```

```
D =
```

```
   -2.0000         0    2.0000
   -2.0000         0    2.0000
   -1.6667    0.3333    1.3333
   -2.0000         0    2.0000
   -0.3333   -1.3333    1.6667
```

```
*** Right now, D is of m(=5) row with m spatial locations
```

```
***                and n(=3) column with n observations at different time.
```

Please note that we use the common notation used in Emery and Thomson so that each row is a time series for one spatial location (one variant) and each column is one time observation.

Please also note that we have taken the temporal means out from each time series. Therefore, the mean along each row is zero.

Step 2: Define the covariance matrix.

```
fac = 1/n;
```

```
C = fac*D*D';
```

```
C =
```

```
    2.6667    2.6667    2.0000    2.6667    1.3333
    2.6667    2.6667    2.0000    2.6667    1.3333
    2.0000    2.0000    1.5556    2.0000    0.7778
    2.6667    2.6667    2.0000    2.6667    1.3333
    1.3333    1.3333    0.7778    1.3333    1.5556
```

Step 3: Compute eigenvalues and eigenvectors of the con-variance matrix via the *eig* function.

Excerpt from the Matlab manual:

$[V,D] = \text{eig}(A)$ produces matrices of eigenvalues (D) and eigenvectors (V) of matrix A , so that $A*V = V*D$. Matrix D is the canonical form of A --a diagonal matrix with A 's eigenvalues on the main diagonal. Matrix V is the modal matrix--its columns are the eigenvectors of A .

```

%
[E,L]=eig(C);
evu=diag(L);

L =
    10.1917         0         0         0         0
         0     0.0000         0         0         0
         0         0     0.9194         0         0
         0         0         0    -0.0000         0
         0         0         0         0    -0.0000

E =
    0.5108    0.8551    0.0885    0.0064   -0.1176
    0.5108   -0.3143    0.0885   -0.2544   -0.2999
    0.3793   -0.2588    0.3118   -0.5828    0.8592
    0.5108   -0.3143    0.0885    0.7579   -0.3344
    0.2708   -0.0647   -0.9377   -0.1457    0.2148

```

Please note that the eigenvector from this function is normalized. That is, $E'E=I$.

```

%*** Check the ortho-normal condition of the eigenvectors.
CH1 = E'*E; %Remember that the columns are eigenvectors.

```

```

CH1 =
    1.0000   -0.0000   -0.0000    0.0000   -0.0000
   -0.0000    1.0000   -0.0000    0.0075   -0.1375
   -0.0000   -0.0000    1.0000   -0.0000    0.0000
    0.0000    0.0075   -0.0000    1.0000   -0.7099
   -0.0000   -0.1375    0.0000   -0.7099    1.0000

```

It should be noted that there are three zero eigenvalues. Therefore, among the eigenvectors associated with such degenerate eigenvalues, there is no guarantee on the orthogonal condition among them. For eigenvectors with different eigenvalues, the normal conditions are always satisfied as shown by the first and the third eigenvectors columns.

Please also note that EE' is not equal to I

```

>> E'*E'
ans =
    1.0139    0.0336   -0.1047    0.0441   -0.0262
    0.0336    0.5222    0.1932    0.2751    0.0483
   -0.1047    0.1932    1.3860   -0.4263    0.0965
    0.0441    0.2751   -0.4263    1.0538   -0.1066
   -0.0262    0.0483    0.0965   -0.1066    1.0241

```

The eigenvalues are not in the desired order.

```

%**** Sort the eigen-values to arrange them from the largest to the smallest.
[evu,ind]=sort(-evu);
evu=-evu;
for i=1:m
    pspatial(:,i)= E(:,ind(i));
end
E = pspatial

```

Step 4: Compute the EOF coefficients (time series).

```
***Compute the time series EOF coefficients
```

```
A = E'*D;
```

```
A =
```

```
-3.7874    -0.2346     4.0219
-0.7381     1.3542    -0.6161
-0.0000    -0.0000    -0.0000
 0.0000     0.0000     0.0000
 0.0000     0.0000    -0.0000
```

```
*** Each row of A (mxn) is a corresponding time series and each time series is a zero-mean TS.
```

```
for i = 1:m
```

```
    mean(A(i,:))
```

```
end
```

```
ans =
```

```
0
```

```
ans =
```

```
3.7007e-17
```

```
ans =
```

```
-2.7938e-16
```

```
ans =
```

```
2.6485e-16
```

```
ans =
```

```
8.6214e-17
```

Step 5: Check the results.

```
L2 = fac*A*A';
```

L2 should be the same as L as L2 is defined temporal average of the coefficients. Since we sorted the eigenvalues and eigenvector, the new result reflects such change.

```
L2 =
```

```
10.1917    -0.0000    -0.0000    -0.0000    -0.0000
-0.0000     0.9194     0.0000    -0.0000     0.0000
-0.0000     0.0000     0.0000    -0.0000     0.0000
-0.0000    -0.0000    -0.0000     0.0000    -0.0000
-0.0000     0.0000     0.0000    -0.0000     0.0000
```

Moreover, one can reconstruct original matrix (before the covariance matrix) by

```
DN = E*A;
```

```
DN =
```

```
-2.0000     0.0000     2.0000
-2.0000    -0.0000     2.0000
-1.6667     0.3333     1.3333
-2.0000    -0.0000     2.0000
-0.3333    -1.3333     1.6667
```

By definition, the total variance of original signal is the sum of temporal variances at each location. This total variance is also equal to the trace of the covariance matrix as well as the sum of all eigenvalues

```
%
```

```
total_v = sum(var(D',1)); %*** total variance
```

```

trace_v = trace(C);

total_v =
    11.1111
trace_v =
    11.1111

sum(evu)
ans = 11.1111

```

Consider the EOF as an approximation, each approximate component contains a certain percentage of the total variance.

```
no_EOF = 2; % There are only two meaningful EOFs.
```

```

for i=1:no_EOF
    a=A(i,:); %%* time series
    e=E(:,i); %%* spatial vector
    ci= e*a; %%* approximation component i
    sum(var(ci',1))
    D_appx(:, :, i) = ci;
end

```

```
ans =
    10.1917
```

```
ans =
    0.9194
```

```

D1 = D_appx(:, :, 1);
D2 = D_appx(:, :, 2);

```

```

D1 =
    -1.9347    -0.1198     2.0545
    -1.9347    -0.1198     2.0545
    -1.4365    -0.0890     1.5255
    -1.9347    -0.1198     2.0545
    -1.0255    -0.0635     1.0890
D2 =
    -0.0653     0.1198    -0.0545
    -0.0653     0.1198    -0.0545
    -0.2302     0.4223    -0.1921
    -0.0653     0.1198    -0.0545
     0.6921    -1.2698     0.5777

```

```

>> D1+D2
ans =
    -2.0000     0.0000     2.0000
    -2.0000    -0.0000     2.0000
    -1.6667     0.3333     1.3333
    -2.0000    -0.0000     2.0000
    -0.3333    -1.3333     1.6667

```

Since each mode of D_j is decomposed by $e*a$ format already. If we perform EOF on the approximated matrix, we will obtain one component only with the total variance being the same as that from the component.

```

C1 = fac*D1*D1';
[E1, L1] = eig(C1);
diag(L1)
ans =
     0
 10.1917
 -0.0000
  0.0000 + 0.0000i
  0.0000 - 0.0000i

```

Discussion:

In all above computations, we assume the original data are temporal anomaly data. Therefore, the variance can be computed by $\text{sum}(\text{component}^{**2})$. Since for EOF, both the spatial patterns (eigenvectors) and the EOF coefficients are orthogonal to each other, their contribution can be added up. The approximation components also satisfy such conditions.

Using D'D

When $m > n$, it is better to use $D'D$ instead of the regular covariance matrix DD' for solving the eigen-value problem.

```

C2 = fac*D'D; %%It doesn't make sense to multiply this factor here.
      %%We may consider the factor are associated with D (1/sqrt(n))

```

```

C2 =
  4.9630 -0.0370 -4.9259
 -0.0370  0.6296 -0.5926
 -4.9259 -0.5926  5.5185

```

```
[A2,L2]=eig(C2);
```

```

A2 =
 -0.5774 -0.4444 -0.6849
 -0.5774  0.8154 -0.0424
 -0.5774 -0.3710  0.7274

```

```

L2 =
  0.0000     0     0
     0  0.9194     0
     0     0 10.1917

```

IT is very clear that the eigenvalues are the same from $D'D$ and DD' .

```

%
evu2=diag(L2);
[evu2,ind]=sort(-evu2);
evu2=-evu2;

for i=1:n
    etemporal(:,i)= A2(:,ind(i));
end
F = etemporal;
F =
 -0.6849 -0.4444 -0.5774
 -0.0424  0.8154 -0.5774
  0.7274 -0.3710 -0.5774

```

Please note that in A2, each column is an eigenvector. And only after transposing the matrix composed with eigenvectors, we will say each row is a time series, the coefficient of an EOF. However, in this case, each time series is normalized already by the function. Therefore, we can consider the time series as a coordinate axis and project the original data onto it to get the spatial pattern. In this case, the spatial pattern will carry the unit.

By definition (Theorem 13.2.5), we know that the EOF = D multiplied by the eigenvectors of D'D

```
*** Compute EOF from the eigenvectors of the transpose matrix
```

```
P2 = D*F;
```

```
P2 =
```

```
 2.8246  0.1470 -0.0000
 2.8246  0.1470 -0.0000
 2.0972  0.5179  0.0000
 2.8246  0.1470 -0.0000
 1.4972 -1.5573 -0.0000
```

```
**Checking the projection relationships.
```

```
%
```

```
V1 = P2'*P2
```

```
V1 =
```

```
30.5751  0.0000 -0.0000
 0.0000  2.7582  0.0000
-0.0000  0.0000  0.0000
```

```
V2= n* evu2
```

```
V2 =
```

```
30.5751
 2.7582
 0.0000
```

```
** Normalize the EOF as commonly defined.
```

```
E2 = P2;
```

```
for i=1:n
```

```
    vec_tmp = P2(:,i);
```

```
    fac2 = sum(vec_tmp.*vec_tmp);
```

```
    if fac2 > 0
```

```
        E2(:,i) = P2(:,i)/sqrt(fac2);
```

```
    end
```

```
end
```

```
E2 =
```

```
 0.5108  0.0885 -0.3714
 0.5108  0.0885 -0.3714
 0.3793  0.3118  0.7428
 0.5108  0.0885 -0.3714
 0.2708 -0.9377 -0.1857
```

```
***Compute the time series EOF coefficients
```

```
AN2 = E2'*D;
```

```
AN2 =
```

```
-3.7874 -0.2346  4.0219
-0.7381  1.3542 -0.6161
```

1.0523 0.4952 -1.5475

We can also check the approximation of D by P and F.

```
%
FT = F';
for i=1:no_EOF
    a=FT(i,:); %%* time series
    e=P2(:,i); %%* spatial vector
    ci= e*a; %%* approximation component i
    sum(var(ci',1))
    D_appx2(:,:,i) = ci;
end

ans =
    10.1917

ans =
    0.9194

D1 = D_appx2(:,:,1);
D2 = D_appx2(:,:,2);

D1 =
    -1.9347    -0.1198     2.0545
    -1.9347    -0.1198     2.0545
    -1.4365    -0.0890     1.5255
    -1.9347    -0.1198     2.0545
    -1.0255    -0.0635     1.0890
D2 =
    -0.0653     0.1198    -0.0545
    -0.0653     0.1198    -0.0545
    -0.2302     0.4223    -0.1921
    -0.0653     0.1198    -0.0545
    0.6921    -1.2698     0.5777

C1 = fac*D2*D2';
[E1, L1] = eig(C1);
diag(L1)
ans =
     0
    0.9194
   -0.0000 + 0.0000i
   -0.0000 - 0.0000i
   -0.0000
```

Please note:

1. The eigenvector lambda can be multiply by a sign. When we use the eigenvector to compute the coefficients, we will get the same opposite sign (see below).
2. In this example, matrix D is of rank two only. The third eigenvectors are degenerated. We should not compare it with other computation because this vector does not make any sense to the problem. The same conclusion applies to the time series coefficients above.

Using SVD

```
%*****
```

```
%*** Now, we use SVD to compute the EOFs.
```

```
Excerpt from the Matlab manual:
```

*[U,S,V] = svd(X) produces a diagonal matrix S of the same dimension as X, with nonnegative diagonal elements in decreasing order, and unitary matrices U and V so that $X = U*S*V'$.*

[U,S,V] = svd(X,0) produces the "economy size" decomposition. If X is m-by-n with $m > n$, then svd computes only the first n columns of U and S is n-by-n.

```
[U,S,V] = svd(D,0)
```

```
U =
```

```
-0.5108 -0.0885 0.2359  
-0.5108 -0.0885 0.3288  
-0.3793 -0.3118 -0.8670  
-0.5108 -0.0885 0.1939  
-0.2708 0.9377 -0.2167
```

```
S =
```

```
5.5295 0 0  
0 1.6608 0  
0 0 0.0000
```

```
V =
```

```
0.6849 0.4444 0.5774  
0.0424 -0.8154 0.5774  
-0.7274 0.3710 0.5774
```

```
****
```

Theoretically, we can prove that $s^{*2} = \lambda$. However, in DD' eigenvalue problems, we multiply a factor 1/n for the temporal means. Therefore, the relationship between the eigenvalues and the singular values are

```
%
```

```
evu3=diag(S);  
evu4 = fac*evu3.*evu3;
```

```
evu4 =
```

```
10.1917  
0.9194  
0.0000
```

```
AN3 = V*S;
```

```
AN3 = AN3'
```

```
AN3 =
```

```
3.7874 0.2346 -4.0219  
0.7381 -1.3542 0.6161  
0.0000 0.0000 0.0000
```

Notes:

1. With SVD, we obtain eigenvalues (variance distributions), eigenvectors (EOFs) and EOF coefficient time series all together. E is given by U as shown above. Eigenvalues are

computed. The last part is for the computation of the time series. Since S is a diagonal matrix, the computation cost is not as high as those with a full matrix.

2. Same as before, eigenvector and its associated coefficient may distribute a sign arbitrarily. If we have the same λ , E and A , we should get all other conclusion without repeating the checking processes.