Application Problems

Problem 3.7 (Latent Semantic Indexing)

```
load("HW03\HW1_DataSet\wordVecV.mat")
M = V;
M( M < 1 ) = 0;
M( M > 0 ) = 1;
S = vecnorm(M);
Mnorm = V./S;
```

```
[U,S,V] = svd(Mnorm)
```

```
U = 1651×1651
   -0.0023   -0.0106   -0.0019   -0.0201   -0.0031    0.0314    0.0015   -0.0186 ···
   -0.0047    0.0246   -0.0028   -0.0035    0.0119    0.0016    0.0091   -0.0077
   -0.0017    0.0009   -0.0011    0.0026   -0.0114    0.0069   -0.0014    0.0350
   -0.0114   -0.0040   -0.0391    0.0053   -0.0562    0.0207   -0.1346   -0.0638
   -0.0113   -0.0141    0.0042    0.0453   -0.0448   -0.0235   -0.0399   -0.0542
   -0.0023   -0.0106   -0.0019   -0.0201   -0.0031    0.0314    0.0015   -0.0186
   -0.0023   -0.0106   -0.0019   -0.0201   -0.0031    0.0314    0.0015   -0.0186
   -0.0029   -0.0084    0.0074    0.0042    0.0209   -0.0024   -0.0089    0.0075
   -0.0017    0.0009   -0.0011    0.0026   -0.0114    0.0069   -0.0014    0.0350
   -0.0045    0.0033   -0.0186    0.0127   -0.0265   -0.0053   -0.0680   -0.0226
     ⋮
S = 1651×10
    8.8275         0         0         0         0         0         0         0 ···
         0    2.4708         0         0         0         0         0         0
         0         0    2.0516         0         0         0         0         0
         0         0         0    1.9314         0         0         0         0
         0         0         0         0    1.7457         0         0         0
         0         0         0         0         0    1.6522         0         0
         0         0         0         0         0         0    1.5659         0
         0         0         0         0         0         0         0    1.4216
         0         0         0         0         0         0         0         0
         0         0         0         0         0         0         0         0
     ⋮
V = 10×10
   -0.2384    0.0341   -0.0363    0.0792   -0.3099    0.1779   -0.0335    0.7719 ···
   -0.2943    0.0604   -0.2810    0.1806   -0.3412   -0.0649   -0.7848   -0.2367
   -0.3180   -0.2908   -0.6076   -0.3131   -0.0798   -0.4527    0.2980    0.1123
   -0.3067    0.4894   -0.4545    0.3962    0.3190    0.3047    0.2805   -0.1483
   -0.2536   -0.3277   -0.0498   -0.4841   -0.0672    0.6483    0.0286   -0.3297
   -0.4357    0.5884    0.3470   -0.5081    0.0535   -0.2535   -0.0251   -0.0468
   -0.2380    0.0588    0.2250    0.0935   -0.3794    0.3095    0.1982    0.1532
   -0.2034   -0.0907    0.2573    0.3079   -0.4041   -0.2829    0.3213   -0.3700
   -0.4261   -0.3430    0.2496    0.1325    0.6033   -0.0643   -0.2315    0.1772
   -0.3574   -0.2970    0.2047    0.3043    0.0307   -0.0431    0.1343   -0.1042
```

Part A)

If since matrix M is the term-by-document matrix and we know that U and V are produced from MM' (document-to-document) and M'M (term-to-term) matricies we can imagine the vectors of U to represent how close the documents are together and V to represent how close the terms are together.

Part B)

```
k= 3;
U_k = U(:, 1:k)
```

```
U_k = 1651×3
   -0.0023   -0.0106   -0.0019
   -0.0047    0.0246   -0.0028
   -0.0017    0.0009   -0.0011
   -0.0114   -0.0040   -0.0391
   -0.0113   -0.0141    0.0042
   -0.0023   -0.0106   -0.0019
   -0.0023   -0.0106   -0.0019
   -0.0029   -0.0084    0.0074
   -0.0017    0.0009   -0.0011
   -0.0045    0.0033   -0.0186
     ⋮
```

```
% S_k = S(1:k, 1:k);
% V_k = V(:, 1:k);
% Mnorm_k = U_k * S_k * V_k';
```

Part C)

```
largest_elements = maxk(S(:), 10)
```

```
largest_elements = 10×1
    8.8275
    2.4708
    2.0516
    1.9314
    1.7457
    1.6522
    1.5659
    1.4216
    1.3984
    1.2598
```

Part d)

```
k= 9;
U_k = U(:, 1:k);
S_k = S(1:k, 1:k);
V_k = V(:, 1:k);
Mnorm_k = U_k * S_k * V_k';
[angles,i]=pdist2(Mnorm_k',Mnorm_k',"cosine","Smallest",2);
[min_angle,index] = mink(angles(2,:),1);
pair = i(2,index);
display([min_angle,index,pair])
```

```
    0.0717    9.0000   10.0000
```

09: Barack Obama

10: George W. Bush

Part e)

```
for k= 1:8
    U_k = U(:, 1:k);
    S_k = S(1:k, 1:k);
    V_k = V(:, 1:k);
    Mnorm_k = U_k * S_k * V_k';
    [angles,i]=pdist2(Mnorm_k',Mnorm_k',"cosine","Smallest",2);
    [min_angle,index] = mink(angles(2,:),1);
    pair = i(2,index);
    display([k,min_angle,index,pair])
end
```

```
     1     0     1     2

    2.0000    0.0000    9.0000   10.0000

    3.0000    0.0000    9.0000   10.0000

    4.0000    0.0063    9.0000   10.0000

    5.0000    0.0178    1.0000    2.0000

    6.0000    0.0325    1.0000    2.0000

    7.0000    0.0453    1.0000    7.0000

    8.0000    0.0536    9.0000   10.0000
```

The lowest value of k that still gives the same answer is 2.

The pair of similar documents for k-1 is:

01: B. J. Cole

02: Mary J. Blige

Problem 3.8 (Eigenfaces and l2 projection)

```
load("HW03\HW3_DataSet\yalefaces.mat")
figure;
imshow(M(:,:,1)/255)
```

N = number of images = 2414

d = number of pixels in image = 1024 = 32*32

x^(i) = a flat vector of an image = 1xd = 1x1024

x bar = average of all x^(i)s = average face

x^(i) bar = x^(i) - x bar = centered data

```
Mflat = reshape(M,1024,[])
```

```
Mflat = 1024×2414
    82    86    76    15    70    53   101    94   118    90   117   120   104 ···
    81    86    88    16    69    51   101    93   120    93   117   121   105
    72    74    80    22    48    37    79    69    90    84   104   110    97
    72    67    62    11    64    41    82    83    99    87    95   104    96
    40    47    60    23    46    50    54    58    69    55    68    77    62
    93    66    20     8   109   105    98   121   134   126    93   116   130
   119    87    22     7   138   138   116   145   152   161   117   144   159
   129   101    40     7   147   141   126   151   158   166   125   144   162
   135   118    75     6   136   122   138   152   168   160   142   153   163
   121   132   137     4   102    78   146   133   163   125   158   158   141
     ⋮
```

```
muX = sum(Mflat')' ./ 2414
```

```
muX = 1024×1
   62.3637
   62.4606
   57.5282
   53.5638
   49.6885
   52.2655
   59.1877
   63.2142
   67.4818
   73.6794
     ⋮
```

```
X = Mflat - muX
```

```
X = 1024×2414
   19.6363   23.6363   13.6363  -47.3637    7.6363   -9.3637   38.6363   31.6363 ···
   18.5394   23.5394   25.5394  -46.4606    6.5394  -11.4606   38.5394   30.5394
   14.4718   16.4718   22.4718  -35.5282   -9.5282  -20.5282   21.4718   11.4718
   18.4362   13.4362    8.4362  -42.5638   10.4362  -12.5638   28.4362   29.4362
   -9.6885   -2.6885   10.3115  -26.6885   -3.6885    0.3115    4.3115    8.3115
   40.7345   13.7345  -32.2655  -44.2655   56.7345   52.7345   45.7345   68.7345
   59.8123   27.8123  -37.1877  -52.1877   78.8123   78.8123   56.8123   85.8123
   65.7858   37.7858  -23.2142  -56.2142   83.7858   77.7858   62.7858   87.7858
   67.5182   50.5182    7.5182  -61.4818   68.5182   54.5182   70.5182   84.5182
   47.3206   58.3206   63.3206  -69.6794   28.3206    4.3206   72.3206   59.3206
     ⋮
```

```
C = X*X'
```

```
C = 1024×1024
10^7 ×
    1.0834    1.0591    0.9640    0.8609    0.7183    0.6881    0.7272    0.7295 ···
    1.0591    1.0845    0.9946    0.8738    0.7132    0.6738    0.7165    0.7171
    0.9640    0.9946    1.0051    0.8850    0.6793    0.6136    0.6478    0.6394
    0.8609    0.8738    0.8850    0.9035    0.7185    0.6195    0.6352    0.6253
    0.7183    0.7132    0.6793    0.7185    0.7696    0.6859    0.6475    0.6129
    0.6881    0.6738    0.6136    0.6195    0.6859    0.7842    0.7666    0.7132
```

```
    0.7272    0.7165    0.6478    0.6352    0.6475    0.7666    0.8676    0.8421
    0.7295    0.7171    0.6394    0.6253    0.6129    0.7132    0.8421    0.8858
    0.7635    0.7496    0.6727    0.6343    0.5940    0.6802    0.8030    0.8596
    0.8256    0.8158    0.7453    0.6828    0.6074    0.6730    0.7616    0.8027
     ⋮
```

Part a)

Looking at `X = UΣV' and C = X' * X we can see that`

The eigenvalues of C are the squares of the singular values of X. This means that if σ is a singular value of X, then σ^2 is an eigenvalue of C.

Additionally, the left-singular vectors of X (the columns of U) are the same as the eigenvectors of C.

Part b)

```
[V, D] = eig(C);
eigenvalues = flip(diag(D))
```

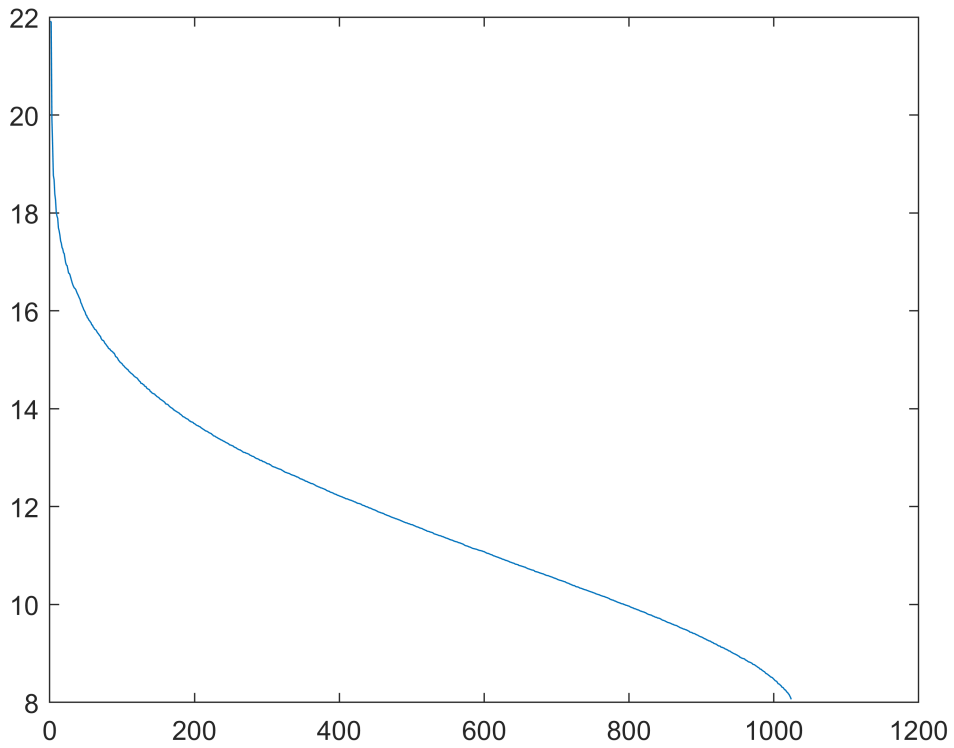
```
eigenvalues = 1024×1
10^9 ×
    3.3037
    3.2362
    0.4473
    0.2597
    0.1409
    0.1297
    0.0966
    0.0858
    0.0667
    0.0615
     ⋮
```

```
eigenvectors = flip(V')'
```

```
eigenvectors = 1024×1024
    0.0130    0.0432    0.0129   -0.0753   -0.0396   -0.0243   -0.0796   -0.0188 ···
    0.0118    0.0434    0.0098   -0.0746   -0.0409   -0.0270   -0.0892   -0.0162
    0.0091    0.0410    0.0067   -0.0747   -0.0442   -0.0344   -0.0941   -0.0276
    0.0078    0.0384    0.0104   -0.0676   -0.0404   -0.0288   -0.0762   -0.0641
    0.0085    0.0344    0.0227   -0.0480   -0.0341    0.0052   -0.0428   -0.0770
    0.0134    0.0344    0.0347   -0.0369   -0.0391    0.0266   -0.0177   -0.0575
    0.0188    0.0370    0.0440   -0.0313   -0.0463    0.0237   -0.0048   -0.0204
    0.0226    0.0370    0.0424   -0.0264   -0.0375    0.0219    0.0033   -0.0113
    0.0260    0.0381    0.0288   -0.0331   -0.0184    0.0250   -0.0029   -0.0091
    0.0276    0.0410    0.0052   -0.0489    0.0117    0.0221   -0.0305   -0.0145
     ⋮
```

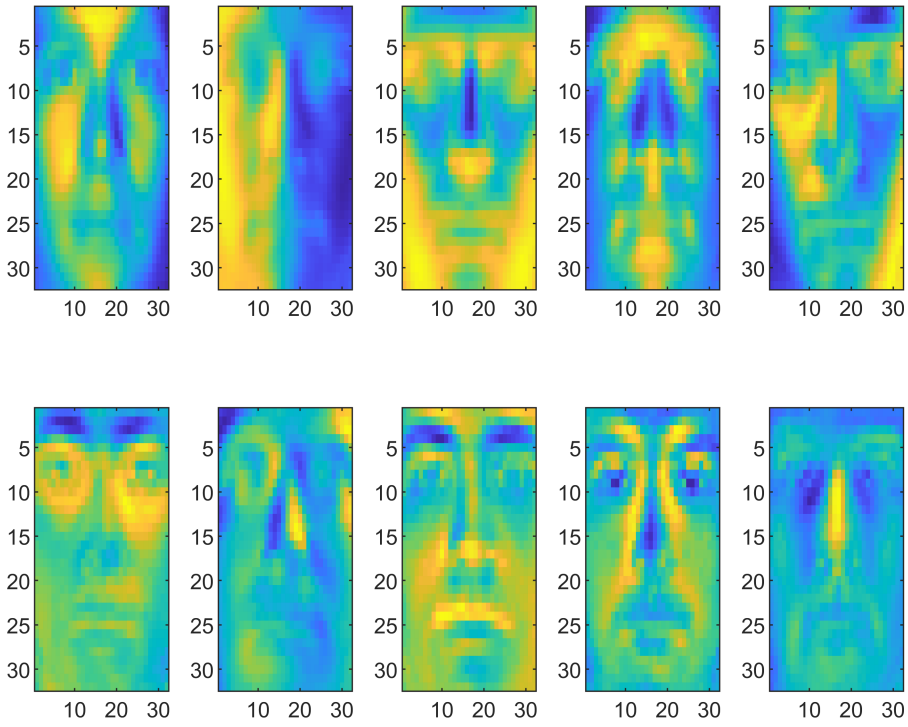
The eigen values are all real.

```
plot(log(eigenvalues))
```

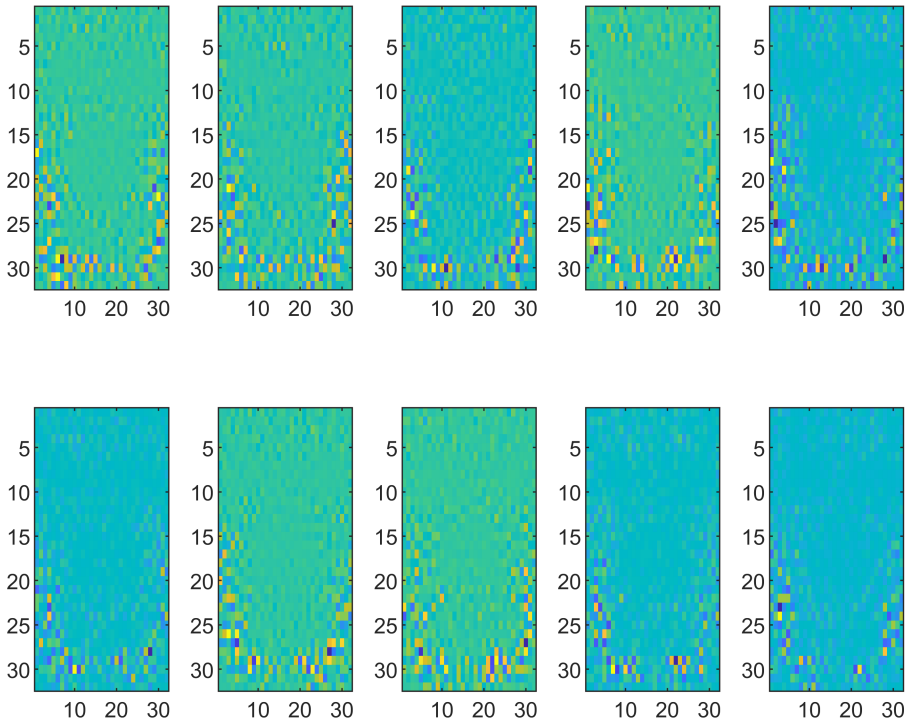


Part c)

```
L = reshape(eigenvectors, 32,32,[]);
figure;
tiledlayout(2,5)
for i = 1:10
    nexttile;
    imagesc(L(:,:,i))
end
```




```
figure;
tiledlayout(2,5)
for i = 1024-9:1024
    nexttile;
    imagesc(L(:,:,i))
end
```


5
10
15
20
25
30
10 20 30