hw04 Yonghyun Kwon

April 18, 2020

1 Problem 1

1.1 How to compile

gcc -c mat_vec.c -Wall -pedantic

gcc -c eval.c -Wall -pedantic

gcc -o testeval testeval.c eval.o mat_vec.o -Wall -pedantic -lm -llapack

1.2 How to use and test the funcion

evd function in eval.c calculates the eigenvalues of a positive definite matrix A using LAPACK package.

You may test this function comparing with eigen.R R script.

Both *testeval* and *eigen.R* computes the eigenvector and eigenvalue of a symmetric matrix

```
> mat
      [,1] [,2] [,3] [,4] [,5] [,6] [,7] [,8] [,9]
 [1,]    2    1    1    1    1    1    1    1    1
 [2,]    1    3    0    0    0    0    0    0    0
 [3,]    1    0    4    0    0    0    0    0    0
 [4,]    1    0    0    5    0    0    0    0    0
 [5,]    1    0    0    0    6    0    0    0    0
 [6,]    1    0    0    0    0    7    0    0    0
 [7,]    1    0    0    0    0    0    8    0    0
 [8,]    1    0    0    0    0    0    0    9    0
 [9,]    1    0    0    0    0    0    0    0   10
```

We can see that both R and c codes return the same eigen values and eigen vector.

1.3 R results

```
> eigen1$values[9:1]
[1]  0.4277777  3.2344480  4.2325592  5.2167362  6.1991267  7.1832593  8.1706597  9.1632842 10.1721489
> eigen1$vectors[,9:1]
            [,1]        [,2]        [,3]        [,4]        [,5]        [,6]        [,7]        [,8]       [,9]
 [1,]  0.85290777  0.21557820 -0.21109465  0.19859806 -0.18479325  0.17204573  0.16174524 -0.15596577 0.16684456
 [2,] -0.33158401  0.91951382 -0.17126533  0.08959030 -0.05776365  0.04112720  0.03128135 -0.02530563 0.02326284
 [3,] -0.23876111 -0.28159838 -0.90770299  0.16322196 -0.08403029  0.05404704  0.03878169 -0.03020670 0.02703184
 [4,] -0.18654119 -0.12210244  0.27506309  0.91631256 -0.15410652  0.07880224  0.05101312 -0.03746220 0.03225827
 [5,] -0.15306420 -0.07795124  0.11943520 -0.25355193 -0.92801827  0.14539985  0.07451432 -0.04930501 0.03999008
 [6,] -0.12977464 -0.05725009  0.07627793 -0.11136774  0.23073969  0.93881037  0.13816588 -0.07209675 0.05259670
 [7,] -0.11263639 -0.04523677  0.05603131 -0.07135438  0.10261313 -0.21064914  0.94776451 -0.13407365 0.07681083
 [8,] -0.09949669 -0.03739073  0.04427840 -0.05249384  0.06597701 -0.09470021 -0.19502880 -0.95517953 0.14234075
 [9,] -0.08910238 -0.03186410  0.03660109 -0.04151936  0.04861863 -0.06107972 -0.08841725  0.18640233 0.96918729
>
```

1.4 C results

```
Eigen Values:
0.427778 3.23445 4.23256 5.21674 6.19913 7.18326 8.17066 9.16328 10.1721

Eigen Vectors:
0.852908 0.215578 0.211095 0.198598 0.184793 -0.172046 -0.161745 -0.155966 0.166845
-0.331584 0.919514 0.171265 0.0895903 0.0577637 -0.0411272 -0.0312814 -0.0253056 0.0232628
-0.238761 -0.281598 0.907703 0.163222 0.0840303 -0.054047 -0.0387817 -0.0302067 0.0270318
-0.186541 -0.122102 -0.275063 0.916313 0.154107 -0.0788022 -0.0510131 -0.0374622 0.0322583
-0.153064 -0.0779512 -0.119435 -0.253552 0.928018 -0.1454 -0.0745143 -0.049305 0.0399901
-0.129775 -0.0572501 -0.0762779 -0.111368 -0.23074 -0.93881 -0.138166 -0.0720968 0.0525967
-0.112636 -0.0452368 -0.0560313 -0.0713544 -0.102613 0.210649 -0.947765 -0.134074 0.0768108
-0.0994967 -0.0373907 -0.0442784 -0.0524938 -0.065977 0.0947002 0.195029 -0.95518 0.142341
-0.0891024 -0.0318641 -0.0366011 -0.0415194 -0.0486186 0.0610797 0.0884173 0.186402 0.969187
```

To test whether they are reliable, one can use

$$P\Lambda P^T = A, \quad P^T P = I$$

For instance, from R results, we can observe that

```
> eigen1 <- eigen(mat)
> lambda <- eigen1$values[9:1]
> P <- eigen1$vectors[,9:1]
> Lambda <- diag(lambda)
> P %*% Lambda %*% t(P)
      [,1]         [,2]         [,3]         [,4]         [,5]         [,6]         [,7]         [,8]         [,9]
 [1,]    2  1.000000e+00  1.000000e+00  1.000000e+00  1.000000e+00  1.000000e+00  1.000000e+00  1.000000e+00  1.000000e+00
 [2,]    1  3.000000e+00 -2.125036e-15 -2.197027e-15 -3.722717e-15 -2.289835e-16  2.189221e-15  1.318390e-15  3.302913e-15
 [3,]    1 -2.125036e-15  4.000000e+00  4.524159e-15  3.141584e-15 -1.327063e-15 -7.480128e-15  4.274359e-15  1.942890e-15
 [4,]    1 -2.193558e-15  4.513750e-15  5.000000e+00  1.247266e-15 -2.716577e-15  4.624773e-15 -2.088607e-15  2.053913e-15
 [5,]    1 -3.668940e-15  3.134645e-15  1.233388e-15  6.000000e+00  1.231307e-14 -5.988265e-15 -1.908196e-15  3.552714e-15
 [6,]    1 -2.289835e-16 -1.327063e-15 -2.612494e-15  1.231307e-14  7.000000e+00  3.566591e-15  3.441691e-15 -1.110223e-15
 [7,]    1  2.210038e-15 -7.487067e-15  4.676814e-15 -5.988265e-15  3.545775e-15  8.000000e+00  1.873501e-15 -7.771561e-16
 [8,]    1  1.373901e-15  4.274359e-15 -2.088607e-15 -1.852685e-15  3.441691e-15  1.873501e-15  9.000000e+00 -4.440892e-16
 [9,]    1  3.302913e-15  1.887379e-15  1.998401e-15  3.552714e-15 -1.221245e-15 -6.661338e-16 -6.661338e-16  1.000000e+01
> t(P) %*% P
              [,1]          [,2]          [,3]          [,4]          [,5]          [,6]          [,7]          [,8]          [,9]
 [1,]  1.000000e+00 -1.379105e-16  1.500536e-16 -3.339343e-17  3.469447e-18 -2.341877e-17 -5.551115e-17  2.428613e-17 -8.326673e-17
 [2,] -1.379105e-16  1.000000e+00  4.080937e-16  4.080937e-16  7.958044e-17 -2.151057e-16  7.242471e-17 -1.535230e-16 -9.367507e-17
 [3,]  1.500536e-16  4.080937e-16  1.000000e+00 -6.303551e-16  5.195497e-16 -8.109832e-17  2.875304e-16 -5.030698e-17  9.020562e-17
 [4,] -3.339343e-17  4.080937e-16 -6.303551e-16  1.000000e+00  6.288373e-17  6.765422e-17 -1.739060e-16  4.510281e-17  0.000000e+00
 [5,]  3.469447e-18  7.958044e-17  5.195497e-16  6.288373e-17  1.000000e+00 -1.330967e-15  2.298509e-16  3.122502e-17 -7.632783e-17
 [6,] -2.341877e-17 -2.151057e-16 -8.109832e-17  6.765422e-17 -1.330967e-15  1.000000e+00 -5.377643e-17 -4.198031e-16 -2.775558e-17
 [7,] -5.551115e-17  7.242471e-17  2.875304e-16 -1.739060e-16  2.298509e-16 -5.377643e-17  1.000000e+00  1.595946e-16  2.081668e-16
 [8,]  2.428613e-17 -1.535230e-16 -5.030698e-17  4.510281e-17  3.122502e-17 -4.198031e-16  1.595946e-16  1.000000e+00 -1.387779e-16
 [9,] -8.326673e-17 -9.367507e-17  9.020562e-17  0.000000e+00 -7.632783e-17 -2.775558e-17  2.081668e-16 -1.387779e-16  1.000000e+00
> all.equal(P %*% Lambda %*% t(P), mat)
[1] TRUE
> all.equal(t(P) %*% P, diag(9))
[1] TRUE
```

Hence, we can say the eigenvalues and eigenvectors computed in R code are accurate.

2 Problem 2

2.1 How to compile

gcc -Wall -pedantic -o RmathGamma RmathGamma.c -lRmath

2.2 Results

log(f(0.5 | shape = 1, rate = 0.5)) = -0.306853

log(f(1 | shape = 1, rate = 0.5)) = -1.30685

log(f(2 | shape = 1, rate = 0.5)) = -3.30685

log(f(0.5 | shape = 2, rate = 0.5)) = -0.306853

log(f(1 | shape = 2, rate = 0.5)) = -0.613706

log(f(2 | shape = 2, rate = 0.5)) = -1.92056

log(f(0.5 | shape = 3, rate = 0.5)) = -1

log(f(1 | shape = 3, rate = 0.5)) = -0.613706

log(f(2 | shape = 3, rate = 0.5)) = -1.22741

log(f(0.5 | shape = 1, rate = 1)) = -0.5

log(f(1 | shape = 1, rate = 1)) = -1

log(f(2 | shape = 1, rate = 1)) = -2

log(f(0.5 | shape = 2, rate = 1)) = -1.19315

log(f(1 | shape = 2, rate = 1)) = -1

log(f(2 | shape = 2, rate = 1)) = -1.30685

log(f(0.5 | shape = 3, rate = 1)) = -2.57944

log(f(1 | shape = 3, rate = 1)) = -1.69315

log(f(2 | shape = 3, rate = 1)) = -1.30685

log(f(0.5 | shape = 1, rate = 2)) = -0.943147

log(f(1 | shape = 1, rate = 2)) = -1.19315

log(f(2 | shape = 1, rate = 2)) = -1.69315

log(f(0.5 | shape = 2, rate = 2)) = -2.32944

log(f(1 | shape = 2, rate = 2)) = -1.88629

log(f(2 | shape = 2, rate = 2)) = -1.69315

log(f(0.5 | shape = 3, rate = 2)) = -4.40888

log(f(1 | shape = 3, rate = 2)) = -3.27259

log(f(2 | shape = 3, rate = 2)) = -2.38629

3 Problem 3

3.1 How to compile

R CMD SHLIB skewkt_Call.c

Then run:

Rscript run_skewkt_Call.R

3.2 Results

The example results are as follows

Skew estimated in C: 2910.828

Skew estimated in R: 2910.828

We can see that the computation time is much longer when we use r based functions.