NA-HW #07

2015004693_양상헌

Goal:

- **Jacobi transformation [jacobi()]**
- **Sorting eigenvalues in descending order [eigsrt()]**
- **Gaussian Distribution Random Number Generator [gasdev()]**

위 함수들을 이용하여 가우스분포(표준정규분포 N(0,1))를 만족하는 난수를 생성하여 11x11 크기의 Symmetric Matrix를 생성하여 이 Matrix의 Eigenvalue 와 EigenVector를 구하고 이를 내림차순으로 정렬한다.

Background Knowledge:

Jacobi Transformation Method는 Symmetric Matrix의 EigenValue들을 하나씩 차례대로 구하는 방식으로서, Matrix **A**에 Rotation Matrix인 **P**와 **P**의 역행렬을 이용하여 **A' = P^-1AP**를 반복해 줌으로서 Eigen Value를 변경시키지 않으면서 Diagonal Element를 제외한 나머지 Element들을 0으로 만들어주고, 이를 이용하여 쉽게 Eigen Value와 Eigen Vector를 구하는 Method 이다.

Process:

(전체적인 과정은 NR에 이미 존재하는 **'xjacobi.c'** 파일을 편집하여 구현하였다.)

가장먼저 N(0,1)의 분포로 발생하는 난수들로 이루어진 Symmetric Matrix를 생성하기 위해 'gasdev()'함수를 사용하여 난수를 생성하였고, 중복되는 난수 발생을 방지하기 위해 'clock()'을 사용하여 'gasdev()'안에 들어가는 parameter seed 값을 계속해서 바꿔주었다. 또한 Matrix가 생성된 후, 전체 Matrix를 출력해주는 부분도 추가하였다.

이후에 'jacobi()'함수를 사용하여 Jacobi Transformation을 실행해주는 과정은 원래의 'xjacobi.c'파일에서 구현되었던 방식과 같은 방법으로 해주었고, Eigen Value들을 Descending order로 sorting 하기 위해 'eigsrt()'함수를 호출하는 부분이 추가되었다.

< 편집된 xjacobi.c >, (편집된 부분은 빨간색으로 표시)

```
#include <stdio.h>
#define NRANSI
#include "nr.h"
#include "nrutil.h"
#include <time.h>
#define NP 11

int main(void)
{
    int i,j,k,kk,l,ll,nrot;
```

```
    float *d,*r,**v,**e;
    float a[NP][NP] = {0.0, };
    long idum = -523;
    clock_t T;

    // random한 number 로 11x11 symmetric matrix 생성
    // (Gaussian distribution) N(0,1)
    for(i= 0; i < NP; i++){
        for(j=i;j<NP;j++){
            T = clock();
            idum = (long)T + idum;
            a[i][j] = a[j][i]= gasdev(&idum);
        }
    }

    //생성된 Matrix 출력
    printf("\nRANDOM NUMBER MATRIX : \n");
    for(i = 0;i<NP;i++){
        for(j = 0;j<NP;j++){
            printf("%10.6f", a[i][j]);
        }
        printf("\n");
    }

    d=vector(1,NP);
    r=vector(1,NP);
    v=matrix(1,NP,1,NP);
    e=convert_matrix(&a[0][0],1,NP,1,NP);

    //jacobi transformation 실행
    jacobi(e,NP,d,v,&nrot);
    printf("\n number of JACOBI rotations: %3d\n\n",nrot);

    //eigenvalue를 descending order로 sort
    eigsrt(d,v,NP);

    printf("eigenvalues in descending order: \n");
    for (j=1;j<=NP;j++) {
        printf("%12.6f\n",d[j]);
    }

    printf("\neigenvectors:\n");
    for (j=1;j<=NP;j++) {
        printf("%9s %3d \n","number",j);
        for (k=1;k<=NP;k++) {
            printf("%12.6f\n",v[k][j]);
        }
        printf("\n");
    }

    printf("eigenvector test\n");
    for (j=1;j<=NP;j++) {
        for (l=1;l<=NP;l++) {
            r[l]=0.0;
            for (k=1;k<=NP;k++) {
                if (k > l) {
                    kk=l;
                    ll=k;
                } else {
                    kk=k;
                    ll=l;
                }
                r[l] += (e[ll][kk]*v[k][j]);
            }
        }
        printf("vector number %3d\n",j);
        printf("%11s %14s %10s %14s\n",
            "vector","mtrx*vec.","ratio", "eigenValue");
        for (l=1;l<=NP;l++)
            printf("%12.6f %12.6f %12.6f %12.6f\n",
                v[l][j],r[l],r[l]/v[l][j],d[j]);
```

```
    }

    free_convert_matrix(e,1,NP,1,NP);
    free_matrix(v,1,NP,1,NP);
    free_vector(r,1,NP);
    free_vector(d,1,NP);
    return 0;
}
#undef NRANSI
```

Results & Discussion:

*/**출력되는 화면을 사진으로 첨부하려 하였으나 출력되는 양이 너무 많아 한번에 캡쳐하는 것이 불가능하여 ctrl C + V 로 출력 결과를 옮겨서 첨부합니다. 전체 출력은 txt파일로 따로 첨부합니다. (IDE: "Xcode", Macbook Pro 2015 Retina)**/*

-실행결과:

```
 RANDOM NUMBER MATRIX :
-0.836854 -0.172280  0.187118  1.615440 -0.176774  0.653145 -0.546364  0.194146 -0.827422 -0.881112  0.451113
-0.172280  0.308985  2.545724  0.551628 -0.959105  1.536523  0.641928  2.087343 -0.505979  2.563689  0.286812
 0.187118  2.545724  0.299986 -1.101832  1.378471  0.274802 -0.815709 -1.069158 -0.467867  0.862028  1.189074
 1.615440  0.551628 -1.101832  1.751060 -2.524084  1.754804  1.895667 -0.813362 -1.021438 -0.006981  0.889103
-0.176774 -0.959105  1.378471 -2.524084  1.236279 -1.029396  0.539402 -1.699266  0.660269 -1.150319  0.965194
 0.653145  1.536523  0.274802  1.754804 -1.029396 -0.433447  0.161802 -0.880663  1.404175 -0.130204 -0.478606
-0.546364  0.641928 -0.815709  1.895667  0.539402  0.161802 -1.148222 -0.494026 -0.227482  1.201077 -1.485524
 0.194146  2.087343 -1.069158 -0.813362 -1.699266 -0.880663 -0.494026 -2.244630  0.268676 -1.962947 -0.652977
-0.827422 -0.505979 -0.467867 -1.021438  0.660269  1.404175 -0.227482  0.268676  1.155450  0.696244 -1.097989
-0.881112  2.563689  0.862028 -0.006981 -1.150319 -0.130204  1.201077 -1.962947  0.696244  0.204259 -0.852870
 0.451113  0.286812  1.189074  0.889103  0.965194 -0.478606 -1.485524 -0.652977 -1.097989 -0.852870 -0.405426

 number of JACOBI rotations: 264

eigenvalues in descending order:
    6.259894
    4.558633
    3.583594
    1.945461
    1.340914
   -0.548080
   -1.406669
   -2.010814
   -2.999542
   -4.404300
   -6.431648

eigenvectors:
   number  1
    0.140573
    0.315096
   -0.096076
    0.619569
   -0.537513
    0.301503
    0.202346
    0.049895
   -0.118142
    0.214248
   -0.068845

   number  2
    0.173205
   -0.561146
   -0.542352
    0.252891
   -0.130683
   -0.098647
   -0.039245
    0.075641
   -0.118427
```

```
-0.498906
-0.003493

number  3
-0.288543
-0.106843
-0.375239
-0.200264
-0.155005
 0.085810
 0.172485
 0.079649
 0.577176
 0.215227
-0.526762

number  4
 0.045393
-0.288206
 0.030152
 0.361540
 0.470242
 0.214256
 0.349260
-0.573083
 0.183051
 0.166579
 0.067534

number  5
 0.261189
 0.071267
 0.164401
 0.036617
 0.019399
 0.544152
-0.362351
 0.109949
 0.556976
-0.350465
 0.159438

number  6
-0.151986
-0.330287
 0.002785
-0.013213
-0.437774
-0.098502
-0.496432
-0.357414
 0.122982
 0.396394
 0.346175

number  7
-0.571981
 0.216355
-0.203243
 0.254086
 0.109810
-0.168594
 0.152815
 0.226407
 0.255019
-0.129822
 0.568487

number  8
 0.621323
 0.098285
-0.120932
 0.102845
```

```
    0.114776
   -0.529116
    0.031182
    0.195690
    0.370502
    0.301625
    0.135804

   number  9
    0.150252
    0.227491
   -0.649893
   -0.312574
    0.225961
    0.384704
   -0.108277
   -0.039395
   -0.238913
    0.266553
    0.255239

   number  10
    0.184419
   -0.189582
    0.125643
   -0.440500
   -0.381974
    0.126113
    0.626073
   -0.000119
    0.066492
   -0.075313
    0.395415

   number  11
    0.061745
    0.482722
   -0.183835
   -0.113878
   -0.186093
   -0.260773
    0.004893
   -0.652748
    0.127376
   -0.408180
   -0.069510

eigenvector test
vector number   1
     vector     mtrx*vec.      ratio    eigenValue
    0.140573    0.879971    6.259891    6.259894
    0.315096    1.972466    6.259892    6.259894
   -0.096076   -0.601425    6.259892    6.259894
    0.619569    3.878439    6.259894    6.259894
   -0.537513   -3.364772    6.259893    6.259894
    0.301503    1.887376    6.259891    6.259894
    0.202346    1.266668    6.259895    6.259894
    0.049895    0.312339    6.259898    6.259894
   -0.118142   -0.739557    6.259894    6.259894
    0.214248    1.341171    6.259895    6.259894
   -0.068845   -0.430965    6.259895    6.259894
vector number   2
     vector     mtrx*vec.      ratio    eigenValue
    0.173205    0.789578    4.558633    4.558633
   -0.561146   -2.558061    4.558634    4.558633
   -0.542352   -2.472382    4.558633    4.558633
    0.252891    1.152839    4.558634    4.558633
   -0.130683   -0.595735    4.558636    4.558633
   -0.098647   -0.449694    4.558632    4.558633
   -0.039245   -0.178904    4.558631    4.558633
    0.075641    0.344820    4.558636    4.558633
   -0.118427   -0.539864    4.558632    4.558633
   -0.498906   -2.274331    4.558633    4.558633
```

```
   -0.003493   -0.015923    4.558640    4.558633
vector number   3
    vector     mtrx*vec.     ratio    eigenValue
   -0.288543   -1.034019    3.583594    3.583594
   -0.106843   -0.382883    3.583594    3.583594
   -0.375239   -1.344705    3.583594    3.583594
   -0.200264   -0.717667    3.583595    3.583594
   -0.155005   -0.555475    3.583595    3.583594
    0.085810    0.307508    3.583596    3.583594
    0.172485    0.618115    3.583594    3.583594
    0.079649    0.285428    3.583595    3.583594
    0.577176    2.068363    3.583594    3.583594
    0.215227    0.771287    3.583594    3.583594
   -0.526762   -1.887700    3.583592    3.583594
vector number   4
    vector     mtrx*vec.     ratio    eigenValue
    0.045393    0.088311    1.945461    1.945461
   -0.288206   -0.560693    1.945462    1.945461
    0.030152    0.058661    1.945476    1.945461
    0.361540    0.703362    1.945461    1.945461
    0.470242    0.914837    1.945461    1.945461
    0.214256    0.416826    1.945462    1.945461
    0.349260    0.679472    1.945462    1.945461
   -0.573083   -1.114911    1.945460    1.945461
    0.183051    0.356118    1.945460    1.945461
    0.166579    0.324074    1.945462    1.945461
    0.067534    0.131384    1.945459    1.945461
vector number   5
    vector     mtrx*vec.     ratio    eigenValue
    0.261189    0.350232    1.340914    1.340914
    0.071267    0.095563    1.340916    1.340914
    0.164401    0.220448    1.340914    1.340914
    0.036617    0.049100    1.340908    1.340914
    0.019399    0.026013    1.340911    1.340914
    0.544152    0.729660    1.340913    1.340914
   -0.362351   -0.485882    1.340914    1.340914
    0.109949    0.147432    1.340912    1.340914
    0.556976    0.746857    1.340914    1.340914
   -0.350465   -0.469944    1.340914    1.340914
    0.159438    0.213792    1.340913    1.340914
vector number   6
    vector     mtrx*vec.     ratio    eigenValue
   -0.151986    0.083301   -0.548080   -0.548080
   -0.330287    0.181024   -0.548080   -0.548080
    0.002785   -0.001526   -0.548027   -0.548080
   -0.013213    0.007241   -0.548043   -0.548080
   -0.437774    0.239935   -0.548080   -0.548080
   -0.098502    0.053987   -0.548081   -0.548080
   -0.496432    0.272084   -0.548080   -0.548080
   -0.357414    0.195891   -0.548078   -0.548080
    0.122982   -0.067404   -0.548079   -0.548080
    0.396394   -0.217256   -0.548080   -0.548080
    0.346175   -0.189731   -0.548079   -0.548080
vector number   7
    vector     mtrx*vec.     ratio    eigenValue
   -0.571981    0.804588   -1.406669   -1.406669
    0.216355   -0.304340   -1.406668   -1.406669
   -0.203243    0.285896   -1.406668   -1.406669
    0.254086   -0.357415   -1.406669   -1.406669
    0.109810   -0.154467   -1.406670   -1.406669
   -0.168594    0.237156   -1.406669   -1.406669
    0.152815   -0.214960   -1.406669   -1.406669
    0.226407   -0.318479   -1.406668   -1.406669
    0.255019   -0.358727   -1.406669   -1.406669
   -0.129822    0.182617   -1.406670   -1.406669
    0.568487   -0.799674   -1.406669   -1.406669
vector number   8
    vector     mtrx*vec.     ratio    eigenValue
    0.621323   -1.249365   -2.010815   -2.010814
    0.098285   -0.197634   -2.010817   -2.010814
   -0.120932    0.243171   -2.010815   -2.010814
    0.102845   -0.206802   -2.010819   -2.010814
    0.114776   -0.230792   -2.010814   -2.010814
```

```
   -0.529116    1.063954   -2.010814   -2.010814
    0.031182   -0.062700   -2.010805   -2.010814
    0.195690   -0.393496   -2.010813   -2.010814
    0.370502   -0.745011   -2.010815   -2.010814
    0.301625   -0.606513   -2.010815   -2.010814
    0.135804   -0.273077   -2.010814   -2.010814
vector number  9
    vector     mtrx*vec.     ratio    eigenValue
    0.150252   -0.450688   -2.999543   -2.999542
    0.227491   -0.682368   -2.999544   -2.999542
   -0.649893    1.949382   -2.999542   -2.999542
   -0.312574    0.937578   -2.999541   -2.999542
    0.225961   -0.677781   -2.999544   -2.999542
    0.384704   -1.153937   -2.999543   -2.999542
   -0.108277    0.324781   -2.999542   -2.999542
   -0.039395    0.118166   -2.999549   -2.999542
   -0.238913    0.716630   -2.999541   -2.999542
    0.266553   -0.799537   -2.999543   -2.999542
    0.255239   -0.765599   -2.999541   -2.999542
vector number  10
    vector     mtrx*vec.     ratio    eigenValue
    0.184419   -0.812236   -4.404298   -4.404300
   -0.189582    0.834974   -4.404301   -4.404300
    0.125643   -0.553371   -4.404300   -4.404300
   -0.440500    1.940094   -4.404300   -4.404300
   -0.381974    1.682330   -4.404301   -4.404300
    0.126113   -0.555440   -4.404299   -4.404300
    0.626073   -2.757412   -4.404299   -4.404300
   -0.000119    0.000522   -4.401992   -4.404300
    0.066492   -0.292851   -4.404304   -4.404300
   -0.075313    0.331702   -4.404306   -4.404300
    0.395415   -1.741524   -4.404299   -4.404300
vector number  11
    vector     mtrx*vec.     ratio    eigenValue
    0.061745   -0.397123   -6.431648   -6.431648
    0.482722   -3.104695   -6.431648   -6.431648
   -0.183835    1.182361   -6.431650   -6.431648
   -0.113878    0.732424   -6.431644   -6.431648
   -0.186093    1.196887   -6.431646   -6.431648
   -0.260773    1.677199   -6.431647   -6.431648
    0.004893   -0.031471   -6.431737   -6.431648
   -0.652748    4.198247   -6.431650   -6.431648
    0.127376   -0.819235   -6.431648   -6.431648
   -0.408180    2.625268   -6.431646   -6.431648
   -0.069510    0.447062   -6.431648   -6.431648
Program ended with exit code: 0
```

-Overall Discussion

전반적으로 원하는 결과가 예상에 맞게 잘 도출된 것 같다고 생각한다.

11x11 크기의 Matrix **A**는 Singular 하지 않다는 전제하에 N개의 Eigen Value와 Eigen Vector가 존재하는데 위 결과에서 11개가 모두 잘 확인 되었고, Eigen Vector의 값과 Matrix **A** 와 각각의 Eigen Vector를 곱한 결과값 사이의 scale 된 비율(ratio)이 각각의 Eigen Value 들과 서로 오차 범위 내에서 같은 값을 갖는다는 결과를 확인 할 수 있어 알맞은 Eigen Value 가 잘 구해졌다고 생각한다.