Numerical Method

Dept. Material Science & Engineering
Undergraduate Course (4th year)
Chang Jae Yu

Homework #3

- Gauss-Jordan Method -> Inverse Matrix

```
double **inverse_matrix(double **mat_data, int n, double err)
```

A B Order

$$\begin{bmatrix} a_{11} & \cdots & a_{1n} & b_1 & ord_1 \\ \vdots & \ddots & \vdots & \vdots & \vdots \\ a_{n1} & \cdots & a_{nn} & b_n & ord_n \end{bmatrix}$$

```
double** mat_ab = (double*)malloc(sizeof(double*)*(n));
for (i = 0; i <= n - 1; i++)
    *(mat_ab + i) = (double*)malloc(sizeof(double)*(n + 2));

double** mat_id = (double*)malloc(sizeof(double*)*(n));
for (i = 0; i <= n - 1; i++)
    *(mat_id + i) = (double*)malloc(sizeof(double)*(n));

///// Matrix를 받아 mat_ab (n * n+2) 행렬을 생성 /////

for (i = 0; i <= n-1; i++) {            //matrix를 합체
    for (j = 0; j <= n-1; j++)
    mat_ab[i][j] = mat_data[i][j];      //a matrix
    mat_ab[i][n] = mat_data[i][n];      //b matrix
    mat_ab[i][n+1] = i+1;               //order
}
```

- Gauss Elimination

```
double **gauss_elimination(double **mat_data, int n, double err)
```

Echelon Form

$$\begin{bmatrix} a_{11} & \cdots & a_{1n} & b_1 & ord_1 \\ \vdots & \ddots & \vdots & \vdots & \vdots \\ 0 & \cdots & a_{nn} & b_n & ord_n \end{bmatrix}$$

```
///// solution matrix 생성 및 계산 //////

double** mat_sol = (double*)malloc(sizeof(double*)*(2));
for (i = 0; i <= n - 1; i++)
    *(mat_sol + i) = (double*)malloc(sizeof(double)*(n));

double sum;

for (i = n - 1; i >= 0; i--) {
    sum = 0;
    for (j = n - 1; j > i; j--)
        sum = sum + (mat_sol[j][1]) * (mat_ab[i][j]);
    mat_sol[i][1] = (mat_ab[i][n] - sum) / (mat_ab[i][i]);
}
```

Homework #3

```
double **inverse_matrix(double **mat_data, int n, double err)
```

Normalization

Row Swap

Gauss Elim.

Diagonal Form

Identity Matrix

```
max_row = fabs(mat_ab[i][0]);
for (j = 1; j <= n-1; j++) {
    if (fabs(max_row) < fabs(mat_ab[i][j]))
        max_row = mat_ab[i][j];
```

```
if (fabs(pivot) < fabs(mat_ab[i][j])) {
    pivot = fabs(mat_ab[i][j]);
    for (k = 0; k <= n + 1; k++) { // k = Column
        dummy = mat_ab[j][k];
        mat_ab[j][k] = mat_ab[i][k];
        mat_ab[i][k] = dummy;
```

```
mult_lower = mat_ab[i][j] / mat_ab[j][j];
for (k = 0; k <= n; k++)
    mat_ab[i][k] = mat_ab[i][k] - mult_lower * mat_ab[j][k];
for (k = 0; k <= n - 1; k++)
    mat_id[i][k] = mat_id[i][k] - mult_lower * mat_id[j][k];
```

```
mult_upper = mat_ab[i][j] / mat_ab[j][j];
for (k = 0; k <= n; k++) {
    mat_ab[i][k] = mat_ab[i][k] - (mult_upper)*(mat_ab[j][k]);
}
```

```
mult_diagonal = mat_ab[i][i];
mat_ab[i][i] = (mat_ab[i][i] / (mult_diagonal));
mat_ab[i][n] = (mat_ab[i][n] / (mult_diagonal));
for (j = 0; j <= n-1; j++)
    mat_id[i][j] = (mat_id[i][j] / (mult_diagonal));
```

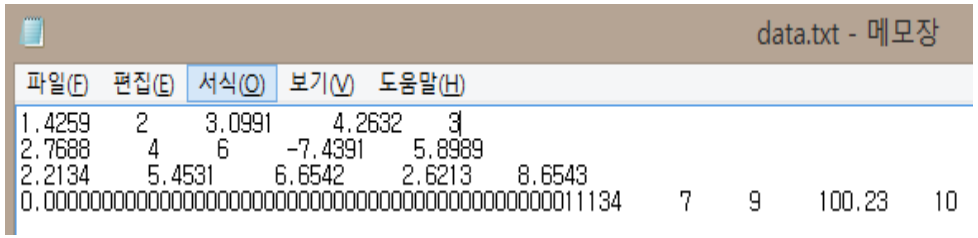
Independence of Row, Column

```
///// mat_ab : Upper Triangle 생성 /////

for (j = 0; j <= n - 1; j++) {          // j = Column
/// Row Swap ///
    pivot = mat_ab[j][j];
    for (i = j; i <= n - 1; i++) {      // i = Row
        if (fabs(pivot) < fabs(mat_ab[i][j])) {
            pivot = fabs(mat_ab[i][j]);
            for (k = 0; k <= n + 1; k++) { // k = Column
                dummy = mat_ab[j][k];
                mat_ab[j][k] = mat_ab[i][k];
                mat_ab[i][k] = dummy;
            }
            for (k = 0; k <= n - 1; k++) {
                dummy_id = mat_id[j][k];
                mat_id[j][k] = mat_id[i][k];
                mat_id[i][k] = dummy_id;
            }
        }
    }
/// Zero Diagonal 검사 ///
    if (pivot == 0) {
        printf("₩n₩nNot independent. Inv. Matrix cannot exist.₩n");
        return 10;
    }
/// Dependence of Row 검사 ///

    for (dep_row = 0; dep_row < n; dep_row++) {
        count = 0;
        for (dep_col = 0; dep_col < n; dep_col++) {
            if (mat_ab[dep_row][dep_col] != 0)
                count++;
        }
        if (count == 0) {
            printf("₩n₩nNot independent. Inv. Matrix cannot exist.₩n");
            return 10;
```

Homework #3




```
<Homework #3> Inverse Matrix by Gauss-Jordan Method

Put the size of square matrix : 4
Size of matrix is 4 (Error Size = 10^(-20))
Open the file 'data.txt'

A

|   1.425900   2.000000   3.099100   4.263200|
|   2.768800   4.000000   6.000000  -7.439100|
|   2.213400   5.453100   6.654200   2.621300|
|   0.000000   7.000000   9.000000 100.230000|

Warning! It will make error because A[3, 0] is too small
```

```
A^(-1)                                                Original row

|   4.352677  -2.405867   0.657307  -0.380892|          2
|   2.139678  -2.229731   1.410822  -0.293398|          3
|  -3.257665   2.660284  -1.229190   0.368156|          4
|   0.143083  -0.083153   0.011842  -0.002590|          1
```

```
X = A^(-1)B   (Calculating by Inverse Matrix)

| X1 |   |   4.352677  -2.405867   0.657307  -0.380892| |   3.000000 |
| X2 |   |   2.139678  -2.229731   1.410822  -0.293398| |   5.898900 |
| X3 | = |  -3.257665   2.660284  -1.229190   0.368156| |   8.654300 |
| X4 |   |   0.143083  -0.083153   0.011842  -0.002590| |  10.000000 |

   |   0.745670 |
   |   2.541771 |
 = |  -1.036468 |
   |   0.015323 |
```

```
Echelon Form
                A                                   B        Original row

|  -0.372196  -0.537699  -0.806549   1.000000| |  -0.792959|      2
|   0.000000   0.338954   0.279185   1.287634| |   0.591908|      3
|   0.000000   0.000000   0.032269   0.734691| |  -0.022188|      4
|   0.000000   0.000000   0.000000   1.639363| |   0.025120|      1

X (Calculating by Gauss Elimination)

| X2|   |   0.745670|
| X3|   |   2.541771|
| X4|   |  -1.036468|
| X1|   |   0.015323|
계속하려면 아무 키나 누르십시오 . . .
```

Homework #3

```
A^(-1)                                              Original row

|   4.352677  -2.405867   0.657307  -0.380892|          2
|   2.139678  -2.229731   1.410822  -0.293398|          3
|  -3.257665   2.660284  -1.229190   0.368156|          4
|   0.143083  -0.083153   0.011842  -0.002590|          1
```

```
X = A^(-1)B   (Calculating by Inverse Matrix)

| X1 |   |   4.352677  -2.405867   0.657307  -0.380892| |   3.000000 |
| X2 |   |   2.139678  -2.229731   1.410822  -0.293398| |   5.898900 |
| X3 | = |  -3.257665   2.660284  -1.229190   0.368156| |   8.654300 |
| X4 |   |   0.143083  -0.083153   0.011842  -0.002590| |  10.000000 |

   |   0.745670 |
   |   2.541771 |
 = |  -1.036468 |
   |   0.015323 |
```

```
Echelon Form
                A                                   B        Original row

|  -0.372196  -0.537699  -0.806549   1.000000| |  -0.792959|      2
|   0.000000   0.338954   0.279185   1.287634| |   0.591908|      3
|   0.000000   0.000000   0.032269   0.734691| |  -0.022188|      4
|   0.000000   0.000000   0.000000   1.639363| |   0.025120|      1

X (Calculating by Gauss Elimination)

| X2|   |   0.745670|
| X3|   |   2.541771|
| X4|   |  -1.036468|
| X1|   |   0.015323|
계속하려면 아무 키나 누르십시오 . . .
```

=

```
A =

    1.4259    2.0000    3.0991    4.2632
    2.7688    4.0000    6.0000   -7.4391
    2.2134    5.4531    6.6542    2.6213
    0.0000    7.0000    9.0000  100.2300

>> B = [3 ; 5.8989 ; 8.6543 ; 10]

B =

    3.0000
    5.8989
    8.6543
   10.0000

>>
```

을(를) 이동하려면 클릭하여 끌어서 놓습니다

```
>> inv (A)

ans =

    4.3527   -2.4059    0.6573   -0.3809
    2.1397   -2.2297    1.4108   -0.2934
   -3.2577    2.6603   -1.2292    0.3682
    0.1431   -0.0832    0.0118   -0.0026

>> inv(A) * B

ans =

    0.7457
    2.5418
   -1.0365
    0.0153
```

Matlab R2016a

Homework #3

```
A

| 1.000000 3.000000|
| 2.000000 5.000000|

AX = B

| 1.000000 3.000000| | X1 |   | 14.000000 |
| 2.000000 5.000000| | X2 | = | 22.000000 |

A^(-1)

| -5.000000 3.000000|          2
| 2.000000 -1.000000|          1

X = A^(-1)B   (Calculating by Inverse Matrix)

| X1 |   | -5.000000 3.000000| | 14.000000 |
| X2 | = | 2.000000 -1.000000| | 22.000000 |

   | -4.000000 |
 = | 6.000000 |
```

$$\begin{bmatrix} a & b \\ c & d \end{bmatrix}^{-1} = \frac{1}{det}\begin{bmatrix} d & -b \\ -c & a \end{bmatrix}$$

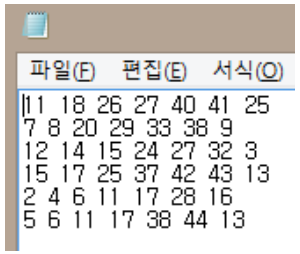
$$\begin{bmatrix} 1 & 3 \\ 2 & 5 \end{bmatrix}^{-1} = \frac{1}{-1}\begin{bmatrix} 5 & -3 \\ -2 & 1 \end{bmatrix} = \begin{bmatrix} -5 & 3 \\ 2 & -1 \end{bmatrix}$$

$$\begin{bmatrix} a & b \\ c & d \end{bmatrix}\begin{bmatrix} x_1 \\ x_2 \end{bmatrix} = \begin{bmatrix} e & f \\ g & h \end{bmatrix} \Rightarrow \begin{bmatrix} c' & d' \\ a' & b' \end{bmatrix}\begin{bmatrix} x_1 \\ x_2 \end{bmatrix} = \begin{bmatrix} g' & h' \\ e' & f \end{bmatrix}$$

$$\begin{bmatrix} x_1 \\ x_2 \end{bmatrix} = \begin{bmatrix} a & b \\ c & d \end{bmatrix}^{-1}\begin{bmatrix} e & f \\ g & h \end{bmatrix} \Rightarrow \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} = \begin{bmatrix} c' & d' \\ a' & b' \end{bmatrix}^{-1}\begin{bmatrix} g' & h' \\ e' & f' \end{bmatrix}$$

- Scaled Partial Pivoting 계산 과정 중에 row 의 순서가 달라져도 inverse matrix는 달라지지 않는다.
  -> A 행렬의 row 순서가 바뀌는 것과 동시에 B 행렬의 row 역시 바뀌기 때문

Homework #3

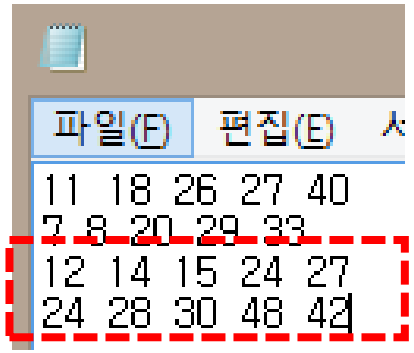


6 X 6 Matrix

```
>> inv(A)

ans =

   -0.0060    0.8685    1.3624   -1.3751   -1.0338    0.2664
    0.0262   -0.7498   -0.9088    1.0242    0.7719   -0.2079
    0.1094    0.5682    0.6698   -0.8365   -0.5415    0.0822
   -0.0841   -0.3588   -0.5630    0.6645    0.4371   -0.1299
   -0.0157   -0.2501   -0.3547    0.3855    0.1369    0.0248
    0.0158    0.2161    0.3255   -0.3640   -0.1395    0.0291
```



Not Independent

```
A^(-1)                                                  Original row

| -0.005979 0.868478 1.362368 -1.375103 -1.033792 0.266427|              3
| 0.026192 -0.749836 -0.908806 1.024158 0.771863 -0.207938|              1
| 0.109407 0.568216 0.669836 -0.836476 -0.541504 0.082226|               2
| -0.084067 -0.358755 -0.563006 0.664499 0.437074 -0.129906|             5
| -0.015715 -0.250098 -0.354716 0.385505 0.136882 0.024763|              6
| 0.015809 0.216110 0.325526 -0.363952 -0.139488 0.029055|               4

Echelon Form
                A                                       B        Original row

| 0.375000 0.437500 0.468750 0.750000 0.843750 1.000000| | 0.093750|      3
| 0.000000 0.126016 0.298780 0.121951 0.371951 0.284553| | 0.542683|      1
| 0.000000 0.000000 0.306452 0.398981 0.466893 0.518676| | 0.209677|      2
| 0.000000 0.000000 0.000000 0.213395 0.295311 0.702414| | 0.308271|      5
| -0.000000 0.000000 0.000000 0.000000 0.409217 0.433451| | 0.144635|     6
| 0.000000 0.000000 0.000000 0.000000 0.000000 -0.063898| | 0.208862|     4

X (Calculating by Gauss Elimination)

| X3|  | -19.199518|
| X1|  | 14.140523|
| X2|  | -8.610695|
| X5|  | 6.923387|
| X6|  | 3.815686|
| X4|  | -3.268672|
```

```
C:₩Windows₩system32₩cmd.e
<Homework #3> Inverse Matrix by Gauss-Jordan Method

Put the size of square matrix : 4
Size of matrix is 4 (Error Size = 10^(-20))
Open the file 'data.txt'

A

| 11.000000 18.000000 26.000000 27.000000|
| 7.000000 8.000000 20.000000 29.000000|
| 12.000000 14.000000 15.000000 24.000000|
| 24.000000 28.000000 30.000000 48.000000|

Not independent. Inv. Matrix cannot exist.
```

Compiler & Language

- Microsoft Visual Studio 2015 with C programming language.

Question & Answer

Normalization

```
max_row = fabs(mat_ab[i][0]);
for (j = 1; j <= n-1; j++) {
    if (fabs(max_row) < fabs(mat_ab[i][j]))
        max_row = mat_ab[i][j];
```

Row Swap

```
if (fabs(pivot) < fabs(mat_ab[i][j])) {
    pivot = fabs(mat_ab[i][j]);
    for (k = 0; k <= n + 1; k++) { // k = Column
        dummy = mat_ab[j][k];
        mat_ab[j][k] = mat_ab[i][k];
        mat_ab[i][k] = dummy;
```

Gauss Elim.

```
mult_lower = mat_ab[i][j] / mat_ab[j][j];
for (k = 0; k <= n; k++)
    mat_ab[i][k] = mat_ab[i][k] - mult_lower * mat_ab[j][k];
for (k = 0; k <= n - 1; k++)
    mat_id[i][k] = mat_id[i][k] - mult_lower * mat_id[j][k];
```

Diagonal Form

```
mult_upper = mat_ab[i][j] / mat_ab[j][j];
for (k = 0; k <= n; k++) {
    mat_ab[i][k] = mat_ab[i][k] - (mult_upper)*(mat_ab[j][k]);
}
```

Identity Matrix

```
mult_diagonal = mat_ab[i][i];
mat_ab[i][i] = (mat_ab[i][i] / (mult_diagonal));
mat_ab[i][n] = (mat_ab[i][n] / (mult_diagonal));
for (j = 0; j <= n-1; j++)
    mat_id[i][j] = (mat_id[i][j] / (mult_diagonal));
```

```
A^(-1)                                                Original row

¦   4.352677  -2.405867   0.657307  -0.380892¦          2
¦   2.139678  -2.229731   1.410822  -0.293398¦          3
¦  -3.257665   2.660284  -1.229190   0.368156¦          4
¦   0.143083  -0.083153   0.011842  -0.002590¦          1
```

```
Echelon Form
                A                                    B        Original row

¦  -0.372196  -0.537699  -0.806549   1.000000¦ ¦  -0.792959¦      2
¦   0.000000   0.338954   0.279185   1.287634¦ ¦   0.591908¦      3
¦   0.000000   0.000000   0.032269   0.734691¦ ¦  -0.022188¦      4
¦   0.000000   0.000000   0.000000   1.639363¦ ¦   0.025120¦      1

X (Calculating by Gauss Elimination)

¦ X2¦  ¦   0.745670¦
¦ X3¦  ¦   2.541771¦
¦ X4¦  ¦  -1.036468¦
¦ X1¦  ¦   0.015323¦
계속하려면 아무 키나 누르십시오 . . .
```

$$\begin{bmatrix} a & b \\ c & d \end{bmatrix}\begin{bmatrix} x_1 \\ x_2 \end{bmatrix}=\begin{bmatrix} e & f \\ g & h \end{bmatrix} \Rightarrow \begin{bmatrix} c' & d' \\ a' & b' \end{bmatrix}\begin{bmatrix} x_1 \\ x_2 \end{bmatrix}=\begin{bmatrix} g' & h' \\ e' & f \end{bmatrix}$$