Homework #3

Numerical Analysis for Materials

POSTECH
POHANG UNIVERSITY OF SCIENCE AND TECHNOLOGY
포항공과대학교

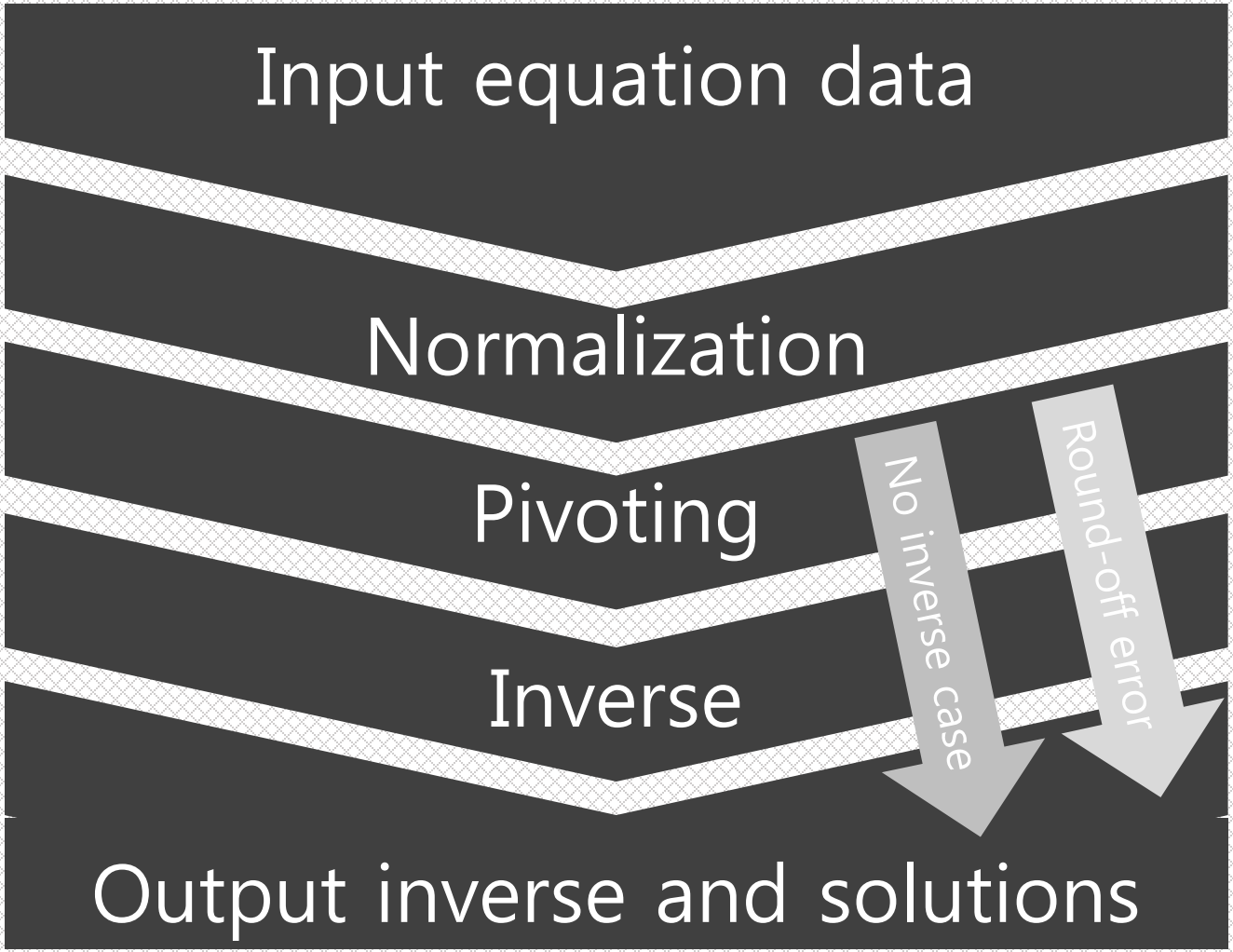
학과: 신소재공학과
학번: 20110085
이름: 도경연

MSE POSTECH Materials Science and Engineering

Problem

- n×n matrix의 inverse를 구하시오.

Main structure



*Error code:
- Normal → 0
- Round-off error → 1
- No inverse or unique solution case → 2

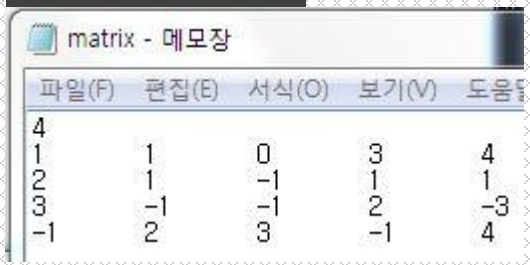
*Input file name:
- matrix.txt

*Output file name:
- matrix_inverse.txt
- solutions.txt

*Total # of functions = 3
*Total # of code lines = 208

Running Program

Input



Output

Program

```
Your matrix is
1.00    1.00    0.00    3.00    |1.00    0.00    0.00    0.00    |4.00
2.00    1.00    -1.00   1.00    |0.00    1.00    0.00    0.00    |1.00
3.00    -1.00   -1.00   2.00    |0.00    0.00    1.00    0.00    |-3.00
-1.00   2.00    3.00    -1.00   |0.00    0.00    0.00    1.00    |4.00

Normalized matrix is
0.33    0.33    0.00    1.00    |0.33    0.00    0.00    0.00    |1.33
1.00    0.50    -0.50   0.50    |0.00    0.50    0.00    0.00    |0.50
1.00    -0.33   -0.33   0.67    |0.00    0.00    0.33    0.00    |-1.00
-0.33   0.67    1.00    -0.33   |0.00    0.00    0.00    0.33    |1.33

Partial pivoted matrix is
1.00    -0.33   -0.33   0.67    |0.00    0.00    0.33    0.00    |-1.00
0.00    0.83    -0.17   -0.17   |0.00    0.50    -0.33   0.00    |1.50
0.00    0.00    1.00    -0.00   |0.00    -0.33   0.33    0.33    |-0.00
0.00    0.00    0.00    0.87    |0.33    -0.20   -0.00   -0.07   |0.87

Inverse matrix is
1.00    0.00    0.00    0.00    |-0.23   0.21    0.33    0.18    |-1.00
0.00    1.00    0.00    0.00    |0.08    0.49    -0.33   0.05    |2.00
0.00    0.00    1.00    -0.00   |0.00    -0.33   0.33    0.33    |-0.00
0.00    0.00    0.00    1.00    |0.38    -0.23   -0.00   -0.08   |1.00
```

Case study

Zero pivot

```
Your matrix is
0.00    4.00    -1.00   |1.00   0.00    0.00    |5.00
1.00    1.00    1.00    |0.00   1.00    0.00    |6.00
2.00    -2.00   1.00    |0.00   0.00    1.00    |1.00

Normalized matrix is
0.00    1.00    -0.25   |0.25   0.00    0.00    |1.25
1.00    1.00    1.00    |0.00   1.00    0.00    |6.00
1.00    -1.00   0.50    |0.00   0.00    0.50    |0.50

Partial pivoted matrix is
1.00    1.00    1.00    |0.00   1.00    0.00    |6.00
0.00    -2.00   -0.50   |0.00   -1.00   0.50    |-5.50
0.00    0.00    -0.50   |0.25   -0.50   0.25    |-1.50

Inverse matrix is
1.00    0.00    0.00    |0.38   -0.25   0.63    |1.00
0.00    1.00    0.00    |0.13   0.25    -0.13   |2.00
0.00    0.00    1.00    |-0.50  1.00    -0.50   |3.00
```

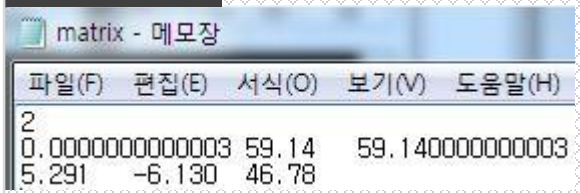
Round-off error

```
Your matrix is
0.00    59.14   |1.00   0.00    |59.14
5.29    -6.13   |0.00   1.00    |46.78

Normalized matrix is
0.00    1.00    |0.02   0.00    |1.00
0.86    -1.00   |0.00   0.16    |7.63

Partial pivoted matrix is
Round-off error
Inverse matrix is
Round-off error
```

Input



No inverse

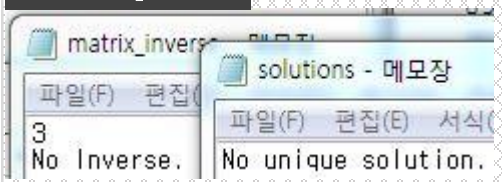
```
Your matrix is
1.00    1.00    1.00    |1.00   0.00    0.00    |1.00
1.00    2.00    3.00    |0.00   1.00    0.00    |4.00
0.00    1.00    2.00    |0.00   0.00    1.00    |3.00

Normalized matrix is
1.00    1.00    1.00    |1.00   0.00    0.00    |1.00
0.33    0.67    1.00    |0.00   0.33    0.00    |1.33
0.00    0.50    1.00    |0.00   0.00    0.50    |1.50

Partial pivoted matrix is
1.00    1.00    1.00    |1.00   0.00    0.00    |1.00
0.00    0.50    1.00    |0.00   0.00    0.50    |1.50
0.00    0.00    0.00    |-0.33  0.33    -0.33   |0.00

Inverse matrix is
No inverse
```

Output



Conclusion

- 입력 받은 행렬의 역행렬을 구할 수 있다.
- Pivoting과 Inverse에서 예외처리가 수행된다.
- 예외처리는 ‘round-off error’와 ‘inverse가 없는 경우’로 구분 가능하다.
- 이 프로그램은 scaled partial pivoting을 성공적으로 수행한다.

Appendix_code main

```c
int main(void)
{
    //n is a size of matrix.
    //er is error code;0_normal, 1_round-off error, 2_no inverse.
    int n,er=0;
    //tol is a tolerance of calculation.
    double tol=pow(10,-12);
    //A is a double pointer for dynamic allocation matrix.
    double **A = NULL;
    //B is a pointer for dynamic allocation vector.
    double *B = NULL;
    //i and j are variable for loop.
    int i,j;
    //f is a pointer for a file.
    FILE *f;
    //Read matrix
    f = fopen("matrix.txt","r");
    //Scan the size of matrix from file.
    fscanf(f,"%d\n",&n);
    //Create the matrix and vector by dynamic allocation
    A = (double**)calloc(n,sizeof(double));
    for(i=0;i<n;i++)
        A[i]=(double*)calloc(2*n,sizeof(double));
    B = (double*)calloc(n,sizeof(double));
    //Scan elements of matrix from file.
    for(i=0;i<n;i++){
        for(j=0;j<n;j++)
            fscanf(f,"%lf\t",&A[i][j]);
        fscanf(f,"%lf\n",&B[i]);
    }
    fclose(f);
    //Create Identity
    for(i=0;i<n;i++){
        for(j=0;j<n;j++){
            if(i==j)
                A[i][j+n]=1;
        }
    }
```

```c
    //View the insert matrix
    printf("Your matrix is\n");
    Print_matrix(A,B,n,&er);

    //Normalization
    Normalization(A,B,n);
    printf("Normalized matrix is\n");
    Print_matrix(A,B,n,&er);

    //Partial_pivoting
    Pivoting(A,B,n,&er,tol);
    printf("Partial pivoted matrix is\n");
    Print_matrix(A,B,n,&er);

    //Inverse
    Inverse(A,B,n,&er,tol);
    printf("Inverse matrix is\n");
    Print_matrix(A,B,n,&er);
```

```c
    //Write inverse matrix
    f = fopen("matrix_inverse.txt","w");
    fprintf(f,"%d\n",n);
    if(er==0){
        for(i=0;i<n;i++){
            for(j=n;j<2*n;j++)
                fprintf(f,"%.4lf\t",A[i][j]);
            fprintf(f,"\n");
        }
    }
    else if(er==1){
        fprintf(f,"Round-off error.");
    }
    else if(er==2){
        fprintf(f,"No Inverse.");
    }
    fclose(f);

    //Write solutions
    f = fopen("solutions.txt","w");
    if(er == 0)
        for(i=0;i<n;i++)
            fprintf(f,"x%d = %lf\n",i,B[i]);
    else if(er == 1){
        fprintf(f,"Round-off error.");
    }
    else if(er == 2){
            fprintf(f,"No unique solution.\n");
    }
    fclose(f);

    return 0;
}
```

Appendix_code Print_matrix

```
void Print_matrix(double** A,double* B,int n,int* er){
    int i,j;
    if(*er==0){
    for(i=0;i<n;i++){
        for(j=0;j<2*n;j++){
            if(j==n)
                printf("|");
            printf("%.2lf\t",A[i][j]);
        }
        printf("|%.2lf\n",B[i]);
    }
    printf("\n");
    }
    else if(*er==1){
        printf("Round-off error\n");
        return;
    }
    else if(*er==2){
        printf("No inverse\n");
        return;
    }
}
```

Appendix_code Normalization

```
void Normalization(double** A,double* B,int n){
    double s[n];
    int i,j;
    for(i=0;i<n;i++){
        s[i] = fabs(A[i][0]);
        for(j=1;j<n;j++)
            if(s[i]<fabs(A[i][j]))
                s[i] = fabs(A[i][j]);
    }
    for(i=0;i<n;i++){
        for(j=0;j<2*n;j++)
            A[i][j] /= s[i];
        B[i] /= s[i];
    }
}
```

Appendix_code Pivoting

```
void Pivoting(double** A,double* B,int n,int* er,double tol){
    double sup,dummy,r;
    int m,i,j,ord;
    for(m=0;m<n-1;m++){
        //Find maximum of mth column under mth row
        sup = fabs(A[m][m]);
        for(i=m+1,ord=m;i<n;i++){
            if(sup < fabs(A[i][m])){
                sup = fabs(A[i][m]);
                ord = i;
            }
        }
        //Exchange rows
        if(ord != m){
            for(j=m;j<2*n;j++){
                dummy = A[m][j];
                A[m][j] = A[ord][j];
                A[ord][j] = dummy;
            }
            dummy = B[m];
            B[m] = B[ord];
            B[ord] = dummy;
        }
        //Eliminate lower elements
        for(i=m+1;i<n;i++){
            //Error check
            if(A[m][m] == 0){
                *er = 2;
                return;
            }
            r = A[i][m]/A[m][m];
            //Error check
            if(fabs(r) < tol && A[i][m] != 0){
                *er = 1;
                return;
            }
            for(j=m;j<2*n;j++)
                A[i][j] -= r*A[m][j];
            B[i] -= r*B[m];
        }
    }
}
```

Appendix_code Inverse

```
void Inverse(double** A,double* B,int n,int* er,double tol){
    int m,i,j;
    double r;
    for(m=n-1;m>0;m--){
        for(i=m-1;i>=0;i--){
            if(A[m][m]<tol){
                *er = 2;
                return;
            }
            r = A[i][m]/A[m][m];
            B[i] -= r*B[m];
            for(j=m;j<2*n;j++)
                A[i][j] -= r*A[m][j];
        }
    }
    for(i=0;i<n;i++){
        r = A[i][i];
        A[i][i] /= r;
        for(j=n;j<2*n;j++)
            A[i][j] /= r;
        B[i] /= r;
    }
}
```