Numerical Methods
Homework 3
- System of linear equations -

20120778
Subin Lee

Problem

- Gauss-Jordan 방법으로 역행렬을 구하는 프로그램을 완성하시오
- 몇 개의 선형 연립방정식 예제를 만들고 완성한 프로그램을 이용, 방정식의 해를 구하시오.

Codes

- Input: The matrix and the size of the matrix
- Scaled partial pivoting
- Elimination
- Substitution

```
// Pivot
scaled_pivot(matrix, SIZE);
printf("\n***AFTER PIVOT***\n");
print_array(matrix, SIZE);

//Eleminating 시작
for (int i=1; i<SIZE;i++){
    for (int j=i; j<SIZE; j++){
        factor = matrix[j][i-1]/matrix[i-1][i-1];
        for (int k=0; k<=SIZE; k++){
            matrix[j][k]=matrix[j][k]-factor*matrix[i-1][k];
        }
    }
}

printf("\n***AFTER ELEMINATION***\n");
print_array(matrix, SIZE);

//Substitute 시작
factor = matrix[SIZE-1][SIZE-1];
matrix[SIZE-1][SIZE]=matrix[SIZE-1][SIZE]/factor;
matrix[SIZE-1][SIZE-1]=matrix[SIZE-1][SIZE-1]/factor;

for(int i=SIZE-2; i!=-1;i--){
    for(int j= i+1; j<SIZE; j++){
        matrix[i][SIZE]=matrix[i][SIZE]-matrix[i][j]*matrix[j][SIZE];
        matrix[i][j]=0.0;
    }
    factor = matrix[i][i];
    matrix[i][SIZE]=matrix[i][SIZE]/factor;
    matrix[i][i]=matrix[i][i]/factor;
}

printf("\n***AFTER SUBSTITUTION***\n");
print_array(matrix, SIZE);
printf("*************************\n");

return 0;
```

Results

```
Insert the size of the materix
3
Insert the matrix
14.294 38.71 64.21 51.871 7.369 5.147 98.18 5.39 27.281
 45.2 68.324 4.258


***ORIGINAL MATRIX***
 14.294000 38.710000 64.210000 51.871000
 7.369000 5.147000 98.180000 5.390000
 27.281000 45.200000 68.324000 4.258000

***AFTER PIVOT***
 27.281000 45.200000 68.324000 4.258000
 14.294000 38.710000 64.210000 51.871000
 7.369000 5.147000 98.180000 5.390000

***AFTER ELEMINATION***
 27.281000 45.200000 68.324000 4.258000
 0.000000 15.027261 28.411340 49.640002
 0.000000 0.000000 93.076826 27.568582

***AFTER SUBSTITUTION***
 1.000000 0.000000 0.000000 -5.130960
 0.000000 1.000000 0.000000 2.743334
 0.000000 0.000000 1.000000 0.296192
**************************
SubinLeeui-Mac:Homework_3 Subin$
```

```
Insert the size of the materix
4
Insert the matrix
25.4 63.7 41.94 4.697 51.286 14.294 38.71 64.21 51.871
7.369 5.147 98.18 5.39 27.281 45.2 68.324 4.258 54.32 9
.63 15.274 31.257 72.149


***ORIGINAL MATRIX***
 25.400000 63.700000 41.940000 4.697000 51.286000
 14.294000 38.710000 64.210000 51.871000 7.369000
 5.147000 98.180000 5.390000 27.281000 45.200000
 68.324000 4.258000 54.320000 9.630000 15.274000

***AFTER PIVOT***
 68.324000 4.258000 54.320000 9.630000 15.274000
 25.400000 63.700000 41.940000 4.697000 51.286000
 14.294000 38.710000 64.210000 51.871000 7.369000
 5.147000 98.180000 5.390000 27.281000 45.200000

***AFTER ELEMINATION***
 68.324000 4.258000 54.320000 9.630000 15.274000
 0.000000 62.117054 21.746100 1.116970 45.607767
 0.000000 0.000000 39.605924 49.176264 -23.594175
 0.000000 0.000000 0.000000 65.721374 -47.436679

***AFTER SUBSTITUTION***
 1.000000 0.000000 0.000000 0.000000 0.046388
 0.000000 1.000000 0.000000 0.000000 0.642012
 0.000000 0.000000 1.000000 0.000000 0.300473
 0.000000 0.000000 0.000000 1.000000 -0.721785
**************************
SubinLeeui-Mac:Homework_3 Subin$
```

Key codes & Discussion

```c
//***scaled_pivot 함수
void scaled_pivot (double **array, int n)
{
    double ** dummy;
    dummy = (double **) malloc( sizeof(double*)*n );
    for( int i=0; i<n; i++) {
        dummy[i] = (double *) malloc( sizeof(double)*n );
    }

    //각 줄의 s_max 계산
    for (int col=0; col<n; col++){
        double max_ary[n-col];
        for(int i=col; i<n; i++){
            max_ary[i-col]=fabs(array[i][i]);
            for(int j=col; j<n; j++){
                if(fabs(array[i][j])>max_ary[i-col]){
                    max_ary[i-col]=fabs(array[i][j]);
                }
            }
        max_ary[i-col]=fabs(array[i][col]/max_ary[i-col]);
        }
        double max=max_ary[0];
        int max_row=0;
        for (int i=0; i<n-col; i++){
            if (max<=max_ary[i]) {
                max=max_ary[i];
                max_row=i+col;
            }
        }
        // row 치환
        dummy[1]=array[col];
        array[col]=array[max_row];
        array[max_row]=dummy[1];
    }
}
```

```c
//***Pivot 함수
void pivot (double **array, int n)
{
    double ** dummy;
    dummy = (double **) malloc( sizeof(double*)*n );
    for( int i=0; i<n; i++) {
        dummy[i] = (double *) malloc( sizeof(double)*n );
    }

    for (int i = 0; i<n; i ++){
        if (array[i][i]==0){
            dummy[i]=array[i+1];
            array[i+1]=array[i];
            array[i]=dummy[i];
        }
    }
}
```

- Calculated the max values (S_{max}) for each row
- Compared $a_{i,col}/S_{max}$, and recorded the values and i
- Replaced the first row to i row

- Comparison between scaled pivot and pivot

```
-5.130959550891152
2.743334220754034
0.296191685641637
```

```
-5.130959550891154
2.743334220754034
0.296191685641637
```

No significant difference