myMatrix.h

```
#include "../../include/myMatrix.h"
```

자신의 폴더에 위치에 있는 myMatrix.h라는 헤더파일 가져오는 방법

```
#include <iostream>
#include <string>
#include <fstream>
```

```
typedef struct {
    double** at;//2d array, 더블 포인터
    int rows, cols;//dimension 정보 저장
}Matrix;
```

행렬을 표현할 구조체 선언해주기, 앞으로 이런 형식으로 행렬을 선언해 줄 것이다.

행렬을 가져올 때 기본적으로 지켜야 될 규칙들

```
#define ASGN        999     // enter your assignment number
#define EVAL        0       // [¡Ø DO NOT EDIT !!!]

#include "../../include/myMatrix.h"
int main(int argc, char* argv[])
{
    /*   [¡Ø DO NOT EDIT !!!]   Resources file path setting for evaluation  */
    std::string path = "C:/NP_data/Assignment" + std::to_string(ASGN) + "/";


#if EVAL
    path += "eval/";
#endif
    return 0;
}
```

Nonlinear equation

- mut_one()
- nonlinearSys()

mul_one()

A행렬의 각 요소들의 값들에다가 특정한 배수를 취하고 싶을 때 사용한다.

```
 Matrix mul_one(Matrix A, double x);
```

Parameters

> Matrix A: 지금 행렬에다가 배수를 취하고 싶은 행렬
>
> x: 배수를 취할 값

Example code

```
#include "../../include/myMatrix.h"
int main(int argc, char* argv[]){
    Matrix Z = zeros(2, 1);
    double x=-1.0;
    Z.at[0][0]=2.5;
    Z.at[1][0]=2.0;
    Matrix mul=zeros(2,1);
    mul=mul_one(Z,x);
    freeMat(T);
    return 0;
}
```

output

```
-Z =
      -2.500000
      -2.000000
```

Warning

- 행렬과 곱할 값을 입력받아야 한다.
- x는 double 형식이어야 한다.

nonlinearSys()

nonlinear 방정식을 풀고 싶을 때 사용을 한다.

이때 비선형 방정식이 모인 행렬과 해당 비선형이 모인 행렬을 편미분한 행렬을 만들어야 한다.

```
Matrix nonlinearSys(Matrix Funcs(Matrix _Z), Matrix Jacob(Matrix _Z), Matrix
_Z0, double tol);
```

Parameters

> Funcs(Matrix _Z): 비선형 방정식들이 모아있는 행렬, Matrix _z는 해당 비선형인 값들이 모인 행렬
>
> Jacob(Matrix _Z): 비선형 방정식들을 편미분을 한 행렬, Matrix _z는 해당 비선형인 값들이 모인 행렬
>
> Matrix _Z0: 해당 비선형인 값들이 모인 행렬
>
> tol: tolerance를 의미

Example code

```
#include "../../include/myMatrix.h"
Matrix myFuncEx1(Matrix X)
{
    int n = X.rows;
    Matrix F = zeros(n, 1);
```

```
    double x1 = X.at[0][0];
    double x2 = X.at[1][0];

    F.at[1][0] = 9 * (x1) * (x1)+25 * x2 * x2 - 225;
    F.at[0][0] = x2 - 1.0 / 2.0 * (exp(x1 / 2.0) + exp(-x1 / 2.0));
    return F;
}


Matrix myJacobEx1(Matrix X)
{
    int n = X.rows;
    Matrix J = zeros(n, n);
    double x1 = X.at[0][0];
    double x2 = X.at[1][0];

    J.at[0][0] = -1.0 / 4.0 * (exp(x1 / 2.0) - exp(-x1 / 2.0));
    J.at[0][1] = 1.0;
    J.at[1][0] = 18 * x1;
    J.at[1][1] = 50 * x2;
    return J;
}
int main(int argc, char* argv[]){
    double z0[2] = { 2.5, 2 };
    Matrix Z = zeros(2, 1);
    Z = arr2Mat(z0, 2, 1);
    Z = nonlinearSys(myFuncEx1, myJacobEx1, Z, 0.001);
    printMat(Z, "Z")
    return 0;
}
```

output

```
iter =0          x=3.139         y=2.400         loss=7.674
iter =1          x=3.034         y=2.385         loss=0.104
iter =2          x=3.031         y=2.386         loss=0.000
Early Termination
Z =
       3.031157
       2.385865
```

Warning

- 초기 Z행렬과 _Z행렬의 값은 같아야 한다.
- tol는 double 형식이어야 한다.

Error Handling

- 가우스 소거법 시 해당 행렬들의 크기가 맞지 않으면 가우스 소거법에 적용한 에러가 발생한다.