myNP.h

```
#include "../../include/myNP.h"
```

자신의 폴더에 위치에 있는 myNP.h라는 헤더파일 가져오는 방법

```
#define     PI      3.14159265358979323846264338327950288419716939937510582
#include <stdio.h>
#include <stdlib.h>
#include <math.h>
```

PI 값 정의 실시

기본 library를 include 시키기

- # Non-Linear Solver
  - bisection()
  - newtonRaphson()
  - secantfzero()

bisection()

bisection을 이용해서 non-linear 문제를 푼다. 양 끝값 a, b의 함수 값과 양 끝 값의 함수값의 평균을 이용한다. xt는 true 값을 의미한다.

1. f(a)f(b)<0이어야 한다.
2. xn(1)=(a+b)/2라고 놓는다.
3. f(a)f(xn(1))<0이면 a<xt<xn(1)이다. 그리고 xn(2)=(a+xn)/2로 놓는다.
4. f(a)f(xn(1))>0이면 xn(1)<xt<b이다. 그리고 xn(2)=(b+xn)/2로 놓는다.
5. f(xn(1))-f(xt)를 이용해 tolerance를 구한다. 그리고 f(xn(1))-f(xt)의 값이 초기에 설정한 tolerance의 값보다 작을 때까지 반복을 실시한다.

f(x)=8-4.5(x-sin(x))=0의 해를 푼다는 가정한다.

```
double bisection(double func(double x), float _a0, float _b0, float _tol);
```

Parameters

> func(double x): 방정식을 계산할 함수
>
> _a0: 초기 선택한 두 값 중에서 작은 값
>
> _b0: 초기 선택한 두 값 중에서 큰 값
>
> _tol: tollerance의 값을 설정

Example code

```
```

```
#include "../../include/myNP.h"
double func(double x)
{

    double F = 0;
     F = 8 - 4.5 * (x - sin(x));
    return F;
}

int main(){
    float tol = 0.00001;
    float a0 = 2;
    float b0 = 3;
    double sol_bm;

    sol_bm = bisection(func, a0, b0, tol);
    printf("%f", sol_bm);
    return 0;
}
```

output

```
2.430466
```

Warning

- f(a)f(b)<0이어야 한다.
- 계산하기 원하는 함수도 같이 넣어주어야 한다.
- _a, _b, _tol는 double 형식이어야 한다.

Error Handling

- f(a)f(b)>0 이면 해가 존재하지 않는다.
- f(a)f(b)=0이면 a 또는 b가 true 값이다.
- 반복횟수가 기존에 설정한 최대 반복횟수를 넘어서게 되면 발산을 한다.

newtonRaphson()

Newton's method을 이용해서 non-linear 문제를 푼다. 이때 함수 값과 미분 값을 이용해서 해를 구한다.

$$x_{k+1} = x_k - \frac{f(x_k)}{f'(x_k)}$$

1. 초기 값 x(1)과 tolerance의 값을 설정해준다.
2. x(1)에서 접선을 그린 후 그 접선이 0과 만나는 값을 x(2)로 놓는다.
3. x(2)를 원래 함수에 다시 대입을 한 후 접선을 그린다. 그리고 그 접선이 0과 만나는 지점을 x(3)로 한다.
4. 이 반복을 f(x(i))-f(xt)가 기존에 설정한 tolerance값보다 작을 때까지 반복을 한다. 여기서 i는 1,2, ... 이다.

f(x)=8-4.5(x-sin(x))=0의 해를 푼다는 가정한다.

```
double newtonRaphson(double func(double x), double dfunc(double x), double
x0, double tol);
```

Parameters

> func(double x): 방정식을 계산할 함수
>
> dfunc(double x): func를 1번 미분한 함수
>
> _x0: 초기 설정 값
>
> _tol: tollerance의 값을 설정

Example code

```
#include "../../include/myNP.h"
double func(double x)
{

    double F = 0;
     F = 8 - 4.5 * (x - sin(x));
    return F;
}

int main(){
    double sol_nr;
    double x0 = 3;

    sol_nr = newtonRaphson(func, dfunc, x0, tol);
    printf("%f", sol_nr);
    return 0;
}
```

output

```
2.430466
```

Warning

- 계산 시 원하는 특정 함수와 특정 함수를 미분한 함수가 필요하다.
- 초기에 tollerance 값과 xo 값을 설정해주어야 한다.

Error Handling

- 미분값이 0이 되면 안 된다.

secantfzero()

Secant method을 이용해서 non-linear 문제를 푼다. Newton's method와 방법은 유사하다. 하지만 함수 값과 두 값의 기울기를 이용해서 해를 구한다는 차이를 갖고 있다.

$$x_{k+1} = x_k - \frac{x_k - x_{k-1}}{f(x_k) - f(x_{k-1})}$$

1. 초기 값 x(0), x(1)과 tolerance의 값을 설정해준다.
2. x(0), x(1)을 이용해 기울기를 구하고 x(1)을 지나도록 직선을 그린다.
3. 직선이 0과 만나는 값을 x(2)로 놓는다.

4. x(2)를 원래 함수에 다시 대입을 한 후 x(2)와 x(1)의 기울기를 구하고 x(2)를 지나도록 직선을 그린다. 그리고 그 직선이 0과 만나는 지점을 x(3)로 한다.
5. 이 반복을 f(x(i))-f(xt)가 기존에 설정한 tolerance값보다 작을 때까지 반복을 한다. 여기서 i는 1,2, ... 이다.

f(x)=8-4.5(x-sin(x))=0의 해를 푼다는 가정한다.

```
double secantfzero(double func(double x), double _x0, double _x1, double
_tol);
```

Parameters

> func(double x): 방정식을 계산할 함수
>
> _x0: 초기 설정한 두 값 중 큰 값
>
> _x1: 초기 설정 값 두 값 중 작은 값
>
> _tol: tollerance의 값을 설정

Example code

```
#include "../../include/myNP.h"
double func(double x)
{

    double F = 0;
     F = 8 - 4.5 * (x - sin(x));
    return F;
}

int main(){
    double x0 = 4;
    double x1 = 3;
    double sol_sm;

    sol_sm = secantfzero(func, x0, x1, tol);
    printf("%f", sol_sm);
    return 0;
}
```

output

```
2.430466
```

Warning

- 계산 시 원하는 특정 함수가 필요하다.
- 초기에 x(0), x(1), tollerance 값을 설정해주어야 한다.

Error Handling

- f(x(k))-f(x(k-1))=0이 되면 안 된다.