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

ODE

- odeRK2()
- ode()
- sysRK2()
- RC1()
- mckfunc()

odeRK2()

Runge-Kutta 2nd order 방식을 이용해서 다음 값을 구한다.

$$y_{i+1} = y_i + (C_1K_1 + C_2K_2)h$$
$$C_1 = 1 - \frac{1}{2\alpha}, C_2 = \frac{1}{2\alpha}$$
$$K_1 = f(t, y)$$
$$K_2 = f(t + \alpha h, y + \beta K_1 h)$$

이 함수의 경우 Modified Euler's method를 이용한다.

$$y_{i+1} = y_i + (C_1K_1 + C_2K_2)h$$
$$C_1 = \frac{1}{2}, C_2 = \frac{1}{2}$$
$$\alpha = \beta = 1$$
$$K_1 = f(t, y)$$
$$K_2 = f(t + h, y + K_1 h)$$

```
void odeRK2(double myfunc(const double t, const double y), double y[], double
t0, double tf, double h, double y0);
```

Parameters

> myfunc(const double t, const double y): 해당 함수의 slope, 변수는 t, y를 이용해서 구한다. t는 x축의 값, y는 y축의 값이다.
>
> y[]: y값들을 모아 놓은 배열. 배열의 크기는 (tf-to)/h+1이다.
>
> t0: 두 값의 범위를 지정하는데 범위가 시작하는 값
>
> tf: 두 값의 범위를 지정하는데 범위가 끝나는 값
>
> h: 구간 간격
>
> y0: y의 초기값

Example code

```
#include "../../include/myNP.h"
double ODE1(const double t, const double y) {
    //t는 시간 y는 전압을 의미
    double tau = 1;
    double T = 1 / tau;
    double f = 10.0;//주파수
    double Vm = 1.0;//크기
    double vout = 0.0;
    vout = -1 / tau * y + 1 / tau * Vm * cos(2 * PI * f * t);
    return vout;
}

int main(){
    double a = 0.0;
    double b = 0.1;
    double h = 0.001;

    double y[101] = {0};//(a-b)/h+1개
    double y0 = y[0];
    odeRK2(ODE1, y, a, b, h, y0);
    for (int i = 0; i < 101; i++) {
        printf("y[%d] %f\n", i, y[i]);
    }
    return 0;
}
```

output

```
y[0] 0.000000
y[1] 0.000999
y[2] 0.001992
...
y[98] -0.001972
y[99] -0.000975
y[100] 0.000024
```

Warning

- y배열의 사이즈를 미리 선언해주어야 한다.
- y배열의 크기는 (tf-t0)/h+1이다.
- 계산을 원하는 함수도 같이 입력해주어야 한다.

odeRKmid()

Runge-Kutta 2nd order 방식을 이용해서 다음 값을 구한다.

$$y_{i+1} = y_i + (C_1K_1 + C_2K_2)h$$
$$C_1 = 1 - \frac{1}{2\alpha}, C_2 = \frac{1}{2\alpha}$$
$$K_1 = f(t, y)$$
$$K_2 = f(t + \alpha h, y + \beta K_1 h)$$

이 함수의 경우 Midpoint method를 이용한다.

$$y_{i+1} = y_i + (C_1K_1 + C_2K_2)h$$
$$C_1 = 0, C_2 = 1$$
$$\alpha = \beta = \frac{1}{2}$$
$$K_1 = f(t, y)$$
$$K_2 = f(t + \frac{1}{2}h, y + \frac{1}{2}K_1 h)$$

```
void odeRKmid(double myfunc(const double t, const double y), double y[], double
t0, double tf, double h, double y0);
```

Parameters

> myfunc(const double t, const double y): 해당 함수의 slope, 변수는 t, y를 이용해서 구한다. t는 x축의 값, y는 y축의 값이다.
>
> y[]: y값들을 모아 놓은 배열. 배열의 크기는 (tf-to)/h+1이다.
>
> t0: 두 값의 범위를 지정하는데 범위가 시작하는 값
>
> tf: 두 값의 범위를 지정하는데 범위가 끝나는 값
>
> h: 구간 간격
>
> y0: y의 초기값

Example code

```
#include "../../include/myNP.h"
double ODE1(const double t, const double y) {
    //t는 시간 y는 전압을 의미
    double tau = 1;
    double T = 1 / tau;
    double f = 10.0;//주파수
    double Vm = 1.0;//크기
    double vout = 0.0;
    vout = -1 / tau * y + 1 / tau * Vm * cos(2 * PI * f * t);
    return vout;
}

int main(){
    double a = 0.0;
    double b = 0.1;
    double h = 0.001;
```

```
    double y[101] = {0};//(a-b)/h+1개
    double y0 = y[0];
    odeRKmid(ODE1, y, a, b, h, y0);
    for (int i = 0; i < 101; i++) {
        printf("y[%d] %f\n", i, y[i]);
    }
    return 0;
}
```

output

```
y[0] 0.000000
y[1] 0.000999
y[2] 0.001996
...
v[98] -0.000884
v[99] -0.001883
v[100] -0.002880
```

Warning

- y배열의 사이즈를 미리 선언해주어야 한다.
- y배열의 크기는 (tf-t0)/h+1이다.
- 계산을 원하는 함수도 같이 입력해주어야 한다.

ode()

Runge-Kutta 3rd order 방식을 이용해서 다음 값을 구한다.

$$y_{i+1} = y_i + (C_1K_1 + C_2K_2 + C_3K_3)h$$
$$K_1 = f(x_i, y_i)$$
$$K_2 = f(x_i + \alpha_2 h, y + \beta_{21}K_1 h)$$
$$K_3 = f(x_i + \alpha_3 h, y_i + \beta_{31}K_1 h + \beta_{32}K_2 h)$$

classical third-order Runge-Kutta일 때 값들이다.

$$C_1 = \frac{1}{6}, C_2 = \frac{4}{6}, C_3 = \frac{1}{6}$$
$$\alpha_2 = \frac{1}{2}, \alpha_3 = 1$$
$$\beta_{21} = \frac{1}{2}, \beta_{31} = -1, \beta_{32} = 2$$
$$y_{i+1} = y_i + \frac{1}{6}(K_1 + 4K_2 + K_3)h$$

```
void ode(double myfunc(const double t, const double y), double y[], double t0,
double tf, double h, double y0);
```

Parameters

> myfunc(const double t, const double y): 해당 함수의 slope, 변수는 t, y를 이용해서 구한다. t는 x축의 값, y는 y축의 값이다.

> y[]: y값들을 모아 놓은 배열. 배열의 크기는 (tf-to)/h+1이다.
>
> t0: 두 값의 범위를 지정하는데 범위가 시작하는 값
>
> tf: 두 값의 범위를 지정하는데 범위가 끝나는 값
>
> h: 구간 간격
>
> y0: y의 초기값

Example code

```
#include "../../include/myNP.h"
double ODE1(const double t, const double y) {
    //t는 시간 y는 전압을 의미
    double tau = 1;
    double T = 1 / tau;
    double f = 10.0;//주파수
    double Vm = 1.0;//크기
    double vout = 0.0;
    vout = -1 / tau * y + 1 / tau * Vm * cos(2 * PI * f * t);
    return vout;
}

int main(){
    double a = 0.0;
    double b = 0.1;
    double h = 0.001;

    double y[101] = {0};//(a-b)/h+1개
    double y0 = y[0];
    odeRK2(ODE1, y, a, b, h, y0);
    for (int i = 0; i < 101; i++) {
        printf("y[%d] %f\n", i, y[i]);
    }
    return 0;
}
```

output

```
y[0] 0.000000
y[1] 0.000999
y[2] 0.001993
...
y[98] -0.001973
y[99] -0.000976
y[100] 0.000024
```

Warning

- y배열의 사이즈를 미리 선언해주어야 한다.
- y배열의 크기는 (tf-t0)/h+1이다.
- 계산을 원하는 함수도 같이 입력해주어야 한다.

sysRK2()

Runge-Kutta 2rd order 방식을 이용해서 다음 값을 구한다.

y'=z이다.

$$C_1 = 1 - \frac{1}{2\alpha}, C_2 = \frac{1}{2\alpha}$$
$$K_{y1} = f_1(t_i, y_i, z_i)$$
$$K_{z1} = f_2(t_i, y_i, z_i)$$
$$yE = y_i + K_{y1}h$$
$$zE = y_i + K_{z1}h$$
$$K_{y2} = f_1(t+h, yE, zE)$$
$$K_{z2} = f_2(t+h, yE, zE)$$
$$y_{i+1} = y_i + (C_1K_{y1} + C_2K_{y2})h$$
$$z_{i+1} = z_i + (C_1K_{z1} + C_2K_{z2})h$$

```
void sysRK2(void func(const double t, const double Y[], double dYdt[]), double
y1[], double y2[], double t0, double tf, double h, double y1_init, double
y2_init);
```

Parameters

> func(const double t, const double Y[], double dYdt[]): 해당 함수의 slope, 변수는 t, y, z를 이용해서 구한다. t는 x축의 값, y는 y축의 값이다. z도 y축에 해당하는 값이지만 y를 미분한 것 이다. Y[]는 크기가 2인 행렬이다. Y[0]=y값, Y[1]=z값을 저장한다. dYdt[0]은 f1의 함수에 의한 값, dYdt[1]은 f2 함수에 의한 값이다.
>
> y1[]: y값들을 모아 놓은 배열. 배열의 크기는 (tf-to)/h+1이다.
>
> y2[]: z값들을 모아 놓은 배열. 배열의 크기는 (tf-to)/h+1이다.
>
> t0: 두 값의 범위를 지정하는데 범위가 시작하는 값
>
> tf: 두 값의 범위를 지정하는데 범위가 끝나는 값
>
> h: 구간 간격
>
> y1_init: y의 초기값
>
> y2_init: z의 초기값

Example code

```
#include "../../include/myNP.h"
void mckfunc(const double t, const double Y[], double dYdt[])
{//[z(t); 1/m(-ky(t)-cz(t)+u(t))
    double m = 1; double c = 7; double k = 6.9; double f = 5;
    double Fin = 2 * cos(2 * PI * f * t);
    dYdt[0] = Y[1];//Y[1]는 z를 의미
    dYdt[1] = 1 * (-k * Y[0] - c * Y[1] + Fin) / m;//dYdt[1]는 z'을 의미
}

int main(){
    double y[101] = { 0 };//(tf-t0)/h+1개
    double y0 = y[0];
    double z[101] = { 0 };//(tf-t0)/h+1개, y'을 의미
    z[0] = 0.2;
```

```
    double z0 = z[0];//0.2m/s

    sysRK2(mckfunc, y, z, t0, tf, h1, y0, z0);
    for (int i = 0; i < 101; i++) {
        printf("y[%d] %f\n", i, y[i]);
    }
    printf("\n");
    for (int i = 0; i < 101; i++) {
        printf("z[%d] %f\n", i, z[i]);
    }
    return 0;
}
```

output

```
y[0] 0.000000
y[1] 0.002030
...
y[99] 0.011059
y[100] 0.010953

z[0] 0.200000
z[1] 0.205232
...
z[99] -0.020453
z[100] -0.000956
```

Warning

- y배열의 사이즈를 미리 선언해주어야 한다.
- y배열의 크기는 (tf-t0)/h+1이다.
- z배열의 사이즈를 미리 선언해주어야 한다.
- z배열의 크기는 (tf-t0)/h+1이다.
- 계산을 원하는 함수도 같이 입력해주어야 한다. 이때 y[], dYdt[]는 크기가 2인 행렬이어야 한다.

RC1()

RC회로에서 기울기값을 알기 위해 사용한다. t는 시간 y는 전압을 의미한다.

$$f(t,v) = \frac{dv}{dt} = -\frac{1}{\tau}v(t) + \frac{1}{\tau}V_m cos(2\pi ft)$$

```
double RC1(const double t, const double y);//RC 회로
```

Parameters

> t: x축에 해당하는 값이다. 시간을 의미한다.
>
> y: y축에 해당하는 값이다. RC회로에서는 전압을 의미한다.

Example code

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

int main(){
    double t = 2.4;
    double y = 0.3;
    printf("y is %f\n",RC1(t, y));
    return 0;
}
```

output

```
y is 0.700000
```

Warning

- t, y는 double 형식이어야 한다.

mckfunc()

mck 함수를 의미한다. 변수는 t, y, z이다.

y'=z이다.

$$\begin{bmatrix} z(t) \\ \frac{1}{m}(-ky(t) - cz(t) + u(t)) \end{bmatrix}$$

```
void mckfunc(const double t, const double Y[], double dYdt[]);
```

Parameters

> t: x축에 해당하는 값이다.
>
> Y[]: x축에 해당하는 y값과 z값을 모은 행렬이다. 행렬의 크기는 2이고 y[0]=y, y[1]=z이다. y는 거리 v는 속도를 의미한다.
>
> dYdt: f1, f2를 모은 행렬이다. 크기는 2이고 dYdt[0]=f1, dYdt[1]=f2이다.

Example code

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

int main(){
    double t = 0;
    double Y[2] = { 0 };
    double dYdt[2] = { 0 };
    Y[0] = 0.0;
    Y[1] = 2.0;
    mckfunc(t, Y, dYdt);
    printf("dYdt[0]=%f\n", dYdt[0]);
    printf("dYdt[1]=%f", dYdt[1]);
    return 0;
}
```

output

```
dYdt[0]=2.000000
dYdt[1]=-12.000000
```

Warning

- y배열의 크기는 2이다.
- y[0]는 y, y[1]은 z의 값이다.
- dYdt배열의 크기는 2이다.
- dYdt[0]는 f1, dYdt[1]은 f2의 값이다.