Exercise: 1st order ODE-IVP

What you need to submit

- Submit the report+source files as a zip file online (LMS)
- **Report:** including pseudocode, output results, and source codes as instructed
- **Src Code:** (1) **Assignment_ode1_Name_ID.cpp, (2) myNP.h, (3) myNP.cpp**
- All the functions you have created should be updated in myNP.h and myNP.cpp

  Refer to MATLAB tutorial code: download source file here

Problem

Solve for the response of an RC circuit with a sinusoidal input

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

```
tau=1; T=1/tau; f=10; Vm=1; w=2*pi*f; a=0; b=0.1; h=0.001;  v(0)=0
```

Procedure

- Write down a pseudocode for the function of (1) Euler (2) Euler's modified method and (3) Runge-Kutta 2nd order method
- Complete MATLAB tutorial: TU_ODE_Par1_student.mlx
- Use MATLAB's function command " ode45()" to solve for the answer and plot the result.
- Create your own C/C++ function.
- You can plot the results in MATLAB and compare with MATLAB's ode45().

Example:

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

Parameter ○ y: 1-D array for output y(t). The length should be predefined and fixed.

○ myfunc is the user defined function that returns f(y,t)=dy/dt ○ t0,tf, h: start time, end time and time intervals, respectively.

- Also, create a function that calls different ODE method

void ode(double myfunc(const double t, const double y), double y[ ], double t0, double tf, double h, int method)

Parameter ○ method: 'EU', 'RK2' 'RK3' ○

Use preprocessor definitions such as

#define EU 0

#define RK2 1

#define RK3 2

Example:

ode(myfunc(t[i], y[i]), double y[ ], t0, tf, h, RK3)

For RK

For RK2, default value is alpha=1

$$y_{i+1} = y_i + (C_1K_1 + C_2K_2)h$$

$$C_1 = 1 - C_2,\ C_2 = \frac{1}{2\alpha} \quad \text{and} \quad \begin{cases} K_1 = f(t, y) \\ K_2 = f(t + \alpha h, y + \beta K_1 h) \end{cases}$$

For RK3, use classical third-order Runge-Kutta

$$C_1 = \frac{1}{6},\ C_2 = \frac{4}{6},\ C_3 = \frac{1}{6},\ \alpha_2 = \frac{1}{2},\ \alpha_3 = 1,\ \beta_{21} = \frac{1}{2},\ \beta_{31} = -1,\ \beta_{32} = 2$$

$$\begin{cases} K_1 = f(x_i, y_i) \\ K_2 = f(x_i + \frac{1}{2}h, y + \frac{1}{2}K_1 h) \\ K_3 = f(x_i + h, y_i - K_1 h + 2K_2 h) \end{cases}$$

$$y_{i+1} = y_i + \frac{1}{6}(K_1 + 4K_2 + K_3)h$$

PSEUDOCODE AND SOURCE CODE

Euler

```
for (t=to to tf, t+=h)
    slope = func(t, yi)
    yi += slope * h
end
```

odeEM

```
for (t=to to tf ;t+=h)
    slope1 = func(t, yi)
    yiE += slope1 * h;
    slope2 = func(t+h, yiE)
    yi += 0.5* (slope1 +slope2) *h
end
```

odeRK2

```
for (t=to to tf ; t+=h)
    K1= func(t, yi)
    yiE += beta * K1 * h
    K2 = func(t+alpha*h , yiE)

    yi += 0.5 *(K1+K2)*h
end
```

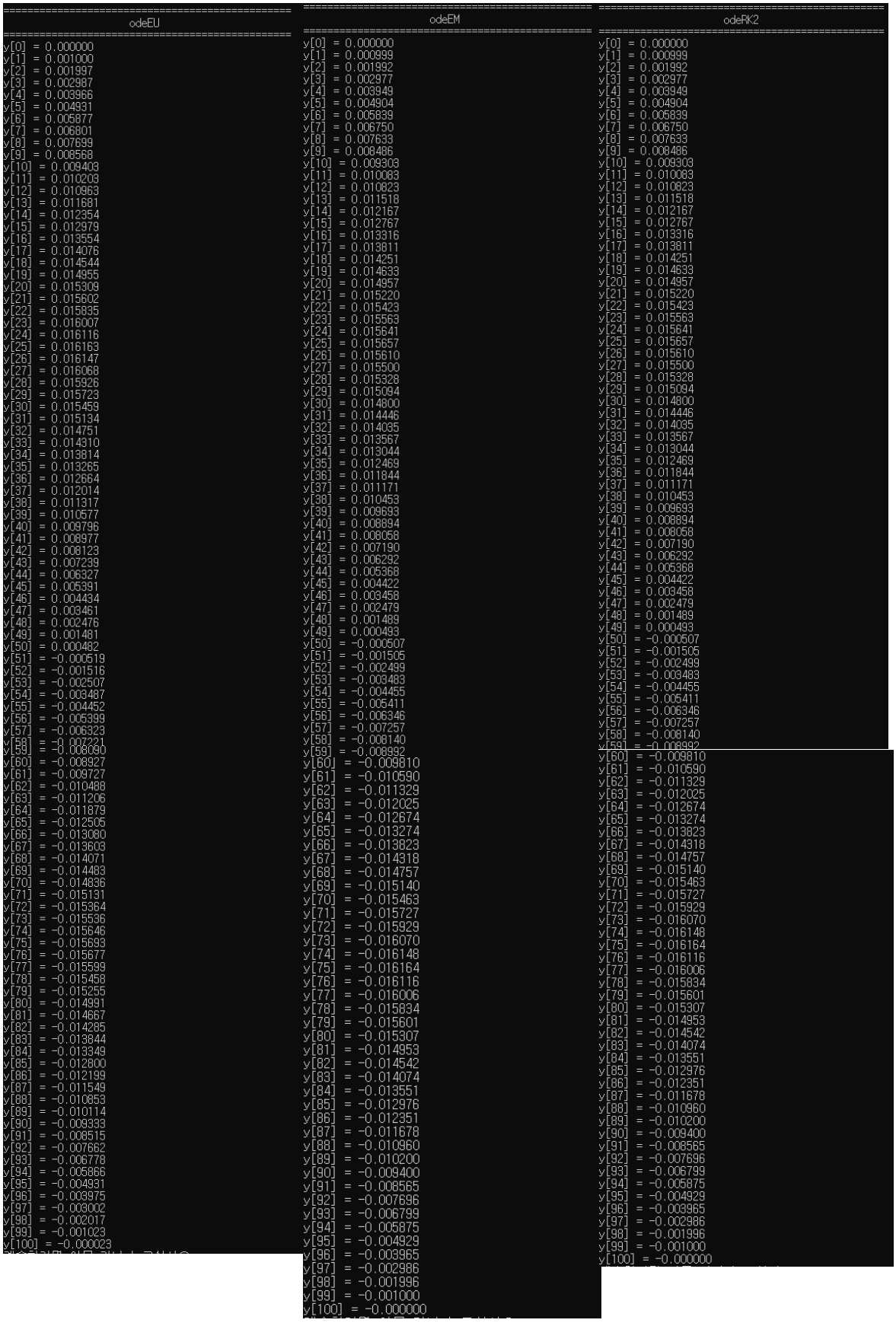
odeRK3

```
for (t=to to tf ;t+=h)
    K1= func(t, yi)
    K2 = func(t+alpha2*h ,yi+beta21 * K1 * h)
    K3 = func(t+alpha3*h ,yi+beta31 *K1 * h)
    yi+= (K1+4K2+K3)/6 * h
end
```

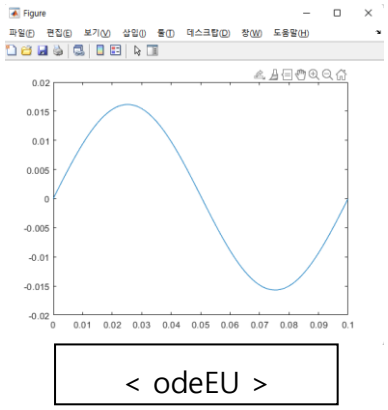
```c
void odeEU(double myfunc(const double t, const double y), double y[], double t0, double tf, double h) {

    double t;
    double yi = 0;
    double slope = 0;
    int k = 0;
    y[k] = yi;

    printf("y[%d] = %f\n", k, y[k]);

    for (t = t0; t <= tf; t+=h) {
        slope = myfunc(t, yi);
        yi = yi + slope * h;

        k++;
        y[k] = yi;
        printf("%f\n",y[k]);
    }
}
void odeEM(double myfunc(const double t, const double y), double y[], double t0, double tf, double h) {

    double yi = 0;
    double yiE = 0;
    double slope1 = 0;
    double slope2 = 0;
    int k = 0;
    y[k] = yi;

    printf("y[%d] = %f\n", k, y[k]);

    for (double t = t0; t <= tf; t += h) {
        slope1 = myfunc(t, yi);
        yiE = yiE + slope1 * h;
        slope2 = myfunc(t + h, yiE);
        yi = yi + 0.5 * (slope1 + slope2) * h;

        k++;
        y[k] = yi;
        printf("%f\n", y[k]);
    }
}
```

```c
void odeRK2(double myfunc(const double t, const double y), double y[], double t0, double tf, double h, double y0) {
    int alpha = 1;
    int beta = alpha;
    double C2 = 1 / 2 * alpha;
    double C1 = 1 - C2;
    double K1 = 0;
    double K2 = 0;
    double yi = y0;
    double yiE = y0;
    int k = 0;
    y[k] = y0;

    printf("y[%d] = %f\n", k, y[k]);

    for (double t = t0; t <= tf; t += h) {
        K1 = myfunc(t, yi);
        yiE = yiE + beta * K1 * h;
        K2 = myfunc(t + alpha * h, yiE);

        yi = yi + 0.5 * (K1 + K2) * h;

        k++;
        y[k] = yi;
        printf("%f\n", y[k]);
    }

}
void odeRK3(double myfunc(const double t, const double y), double y[], double t0, double tf, double h, double y0) {
    int alpha2 = 1;
    int alpha3 = 1;
    double beta21 = 0.5;
    int beta31 = -1;
    int beta32 = 2;
    double C1 = 1 / 6;
    double C2 = 4 / 6;
    double C3 = 1 / 6;
    double C4 = 1 / 2;

    double K1 = 0;
    double K2 = 0;
    double K3 = 0;

    double yi = y0;
    int k = 0;
    y[k] = y0;

    printf("y[%d] = %f\n", k, y[k]);

    for (int t = t0; t < tf; t+=h){

        K1 = myfunc(t, yi);
        K2 = myfunc(t + alpha2 * h, yi + beta21 * K1 * h);
        K3 = myfunc(t + alpha3 * h, yi + beta31 * K1 * h);

        yi = yi + (K1 + 4 * K2 + K3) * h / 6;

        k++;
        y[k] = yi;
        printf("%f\n", y[k]);
    }
}
```

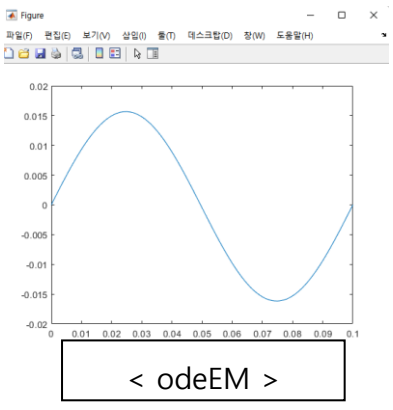
```c
#include "../../../include/myNP.h"

double myfunc(const double t, const double y);
int main(int argc, char* argv[]) {

    double t0 = 0;
    double tf = 0.1;
    double h = 0.001;
    double y[101] = { 0 };

    ode(myfunc, y, t0, tf, h, EU);
    ode(myfunc, y, t0, tf, h, EM);
    ode(myfunc, y, t0, tf, h, RK2);
    ode(myfunc, y, t0, tf, h, RK3);

    system("pause");
    return 0;

}

double myfunc(const double t, const double y) {

    double tau = 1;
    double T = 1 / tau;
    double f = 10;
    double Vm = 1;
    double w = 2 * PI * f;

    double F = -T * y + T * Vm * cos(w * t);

    return F;
}
```


odeEU
y[0] = 0.000000
y[1] = 0.001000
y[2] = 0.001997
y[3] = 0.002987
y[4] = 0.003966
y[5] = 0.004931
y[6] = 0.005877
y[7] = 0.006801
y[8] = 0.007699
y[9] = 0.008568
y[10] = 0.009403
y[11] = 0.010203
y[12] = 0.010963
y[13] = 0.011681
y[14] = 0.012354
y[15] = 0.012979
y[16] = 0.013554
y[17] = 0.014076
y[18] = 0.014544
y[19] = 0.014955
y[20] = 0.015309
y[21] = 0.015602
y[22] = 0.015835
y[23] = 0.016007
y[24] = 0.016116
y[25] = 0.016163
y[26] = 0.016147
y[27] = 0.016068
y[28] = 0.015926
y[29] = 0.015723
y[30] = 0.015459
y[31] = 0.015134
y[32] = 0.014751
y[33] = 0.014310
y[34] = 0.013814
y[35] = 0.013265
y[36] = 0.012664
y[37] = 0.012014
y[38] = 0.011317
y[39] = 0.010577
y[40] = 0.009796
y[41] = 0.008977
y[42] = 0.008123
y[43] = 0.007239
y[44] = 0.006327
y[45] = 0.005391
y[46] = 0.004434
y[47] = 0.003461
y[48] = 0.002476
y[49] = 0.001481
y[50] = 0.000482
y[51] = -0.000519
y[52] = -0.001516
y[53] = -0.002507
y[54] = -0.003487
y[55] = -0.004452
y[56] = -0.005399
y[57] = -0.006323
y[58] = -0.007221
y[59] = -0.008090
y[60] = -0.008927
y[61] = -0.009727
y[62] = -0.010488
y[63] = -0.011206
y[64] = -0.011879
y[65] = -0.012505
y[66] = -0.013080
y[67] = -0.013603
y[68] = -0.014071
y[69] = -0.014483
y[70] = -0.014836
y[71] = -0.015131
y[72] = -0.015364
y[73] = -0.015536
y[74] = -0.015646
y[75] = -0.015693
y[76] = -0.015677
y[77] = -0.015599
y[78] = -0.015458
y[79] = -0.015255
y[80] = -0.014991
y[81] = -0.014667
y[82] = -0.014285
y[83] = -0.013844
y[84] = -0.013349
y[85] = -0.012800
y[86] = -0.012199
y[87] = -0.011549
y[88] = -0.010853
y[89] = -0.010114
y[90] = -0.009333
y[91] = -0.008515
y[92] = -0.007662
y[93] = -0.006778
y[94] = -0.005866
y[95] = -0.004931
y[96] = -0.003975
y[97] = -0.003002
y[98] = -0.002017
y[99] = -0.001023
y[100] = -0.000023
odeEM
y[0] = 0.000000
y[1] = 0.000999
y[2] = 0.001992
y[3] = 0.002977
y[4] = 0.003949
y[5] = 0.004904
y[6] = 0.005839
y[7] = 0.006750
y[8] = 0.007633
y[9] = 0.008486
y[10] = 0.009303
y[11] = 0.010083
y[12] = 0.010823
y[13] = 0.011518
y[14] = 0.012167
y[15] = 0.012767
y[16] = 0.013316
y[17] = 0.013811
y[18] = 0.014251
y[19] = 0.014633
y[20] = 0.014957
y[21] = 0.015220
y[22] = 0.015423
y[23] = 0.015563
y[24] = 0.015641
y[25] = 0.015657
y[26] = 0.015610
y[27] = 0.015500
y[28] = 0.015328
y[29] = 0.015094
y[30] = 0.014800
y[31] = 0.014446
y[32] = 0.014035
y[33] = 0.013567
y[34] = 0.013044
y[35] = 0.012469
y[36] = 0.011844
y[37] = 0.011171
y[38] = 0.010453
y[39] = 0.009693
y[40] = 0.008894
y[41] = 0.008058
y[42] = 0.007190
y[43] = 0.006292
y[44] = 0.005368
y[45] = 0.004422
y[46] = 0.003458
y[47] = 0.002479
y[48] = 0.001489
y[49] = 0.000493
y[50] = -0.000507
y[51] = -0.001505
y[52] = -0.002499
y[53] = -0.003483
y[54] = -0.004455
y[55] = -0.005411
y[56] = -0.006346
y[57] = -0.007257
y[58] = -0.008140
y[59] = -0.008992
y[60] = -0.009810
y[61] = -0.010590
y[62] = -0.011329
y[63] = -0.012025
y[64] = -0.012674
y[65] = -0.013274
y[66] = -0.013823
y[67] = -0.014318
y[68] = -0.014757
y[69] = -0.015140
y[70] = -0.015463
y[71] = -0.015727
y[72] = -0.015929
y[73] = -0.016070
y[74] = -0.016148
y[75] = -0.016164
y[76] = -0.016116
y[77] = -0.016006
y[78] = -0.015834
y[79] = -0.015601
y[80] = -0.015307
y[81] = -0.014953
y[82] = -0.014542
y[83] = -0.014074
y[84] = -0.013551
y[85] = -0.012976
y[86] = -0.012351
y[87] = -0.011678
y[88] = -0.010960
y[89] = -0.010200
y[90] = -0.009400
y[91] = -0.008565
y[92] = -0.007696
y[93] = -0.006799
y[94] = -0.005875
y[95] = -0.004929
y[96] = -0.003965
y[97] = -0.002986
y[98] = -0.001996
y[99] = -0.001000
y[100] = -0.000000
odeRK2
y[0] = 0.000000
y[1] = 0.000999
y[2] = 0.001992
y[3] = 0.002977
y[4] = 0.003949
y[5] = 0.004904
y[6] = 0.005839
y[7] = 0.006750
y[8] = 0.007633
y[9] = 0.008486
y[10] = 0.009303
y[11] = 0.010083
y[12] = 0.010823
y[13] = 0.011518
y[14] = 0.012167
y[15] = 0.012767
y[16] = 0.013316
y[17] = 0.013811
y[18] = 0.014251
y[19] = 0.014633
y[20] = 0.014957
y[21] = 0.015220
y[22] = 0.015423
y[23] = 0.015563
y[24] = 0.015641
y[25] = 0.015657
y[26] = 0.015610
y[27] = 0.015500
y[28] = 0.015328
y[29] = 0.015094
y[30] = 0.014800
y[31] = 0.014446
y[32] = 0.014035
y[33] = 0.013567
y[34] = 0.013044
y[35] = 0.012469
y[36] = 0.011844
y[37] = 0.011171
y[38] = 0.010453
y[39] = 0.009693
y[40] = 0.008894
y[41] = 0.008058
y[42] = 0.007190
y[43] = 0.006292
y[44] = 0.005368
y[45] = 0.004422
y[46] = 0.003458
y[47] = 0.002479
y[48] = 0.001489
y[49] = 0.000493
y[50] = -0.000507
y[51] = -0.001505
y[52] = -0.002499
y[53] = -0.003483
y[54] = -0.004455
y[55] = -0.005411
y[56] = -0.006346
y[57] = -0.007257
y[58] = -0.008140
y[59] = -0.008992
y[60] = -0.009810
y[61] = -0.010590
y[62] = -0.011329
y[63] = -0.012025
y[64] = -0.012674
y[65] = -0.013274
y[66] = -0.013823
y[67] = -0.014318
y[68] = -0.014757
y[69] = -0.015140
y[70] = -0.015463
y[71] = -0.015727
y[72] = -0.015929
y[73] = -0.016070
y[74] = -0.016148
y[75] = -0.016164
y[76] = -0.016116
y[77] = -0.016006
y[78] = -0.015834
y[79] = -0.015601
y[80] = -0.015307
y[81] = -0.014953
y[82] = -0.014542
y[83] = -0.014074
y[84] = -0.013551
y[85] = -0.012976
y[86] = -0.012351
y[87] = -0.011678
y[88] = -0.010960
y[89] = -0.010200
y[90] = -0.009400
y[91] = -0.008565
y[92] = -0.007696
y[93] = -0.006799
y[94] = -0.005875
y[95] = -0.004929
y[96] = -0.003965
y[97] = -0.002986
y[98] = -0.001996
y[99] = -0.001000
y[100] = -0.000000

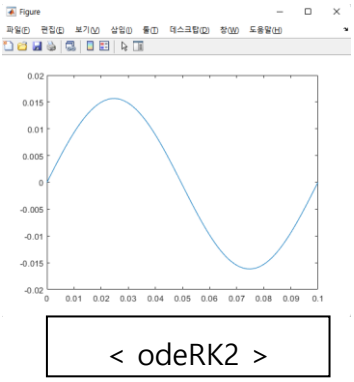
```
==========================================
                  odeRK3
==========================================
y[0] = 0.000000
y[1] = 0.000999
y[2] = 0.001992
y[3] = 0.002977
y[4] = 0.003949
y[5] = 0.004904
y[6] = 0.005839
y[7] = 0.006750
y[8] = 0.007633
y[9] = 0.008486
y[10] = 0.009303
y[11] = 0.010083
y[12] = 0.010823
y[13] = 0.011518
y[14] = 0.012167
y[15] = 0.012767
y[16] = 0.013316
y[17] = 0.013811
y[18] = 0.014251
y[19] = 0.014633
y[20] = 0.014957
y[21] = 0.015220
y[22] = 0.015423
y[23] = 0.015563
y[24] = 0.015641
y[25] = 0.015657
y[26] = 0.015610
y[27] = 0.015500
y[28] = 0.015328
y[29] = 0.015094
y[30] = 0.014800
y[31] = 0.014446
y[32] = 0.014035
y[33] = 0.013567
y[34] = 0.013044
y[35] = 0.012469
y[36] = 0.011844
y[37] = 0.011171
y[38] = 0.010453
y[39] = 0.009693
y[40] = 0.008894
y[41] = 0.008058
y[42] = 0.007190
y[43] = 0.006292
y[44] = 0.005368
y[45] = 0.004422
y[46] = 0.003458
y[47] = 0.002479
y[48] = 0.001489
y[49] = 0.000493
y[50] = -0.000507
y[51] = -0.001505
y[52] = -0.002499
y[53] = -0.003483
y[54] = -0.004455
y[55] = -0.005411
y[56] = -0.006346
y[57] = -0.007257
y[58] = -0.008140
y[59] = -0.008992
y[60] = -0.009810
y[61] = -0.010590
y[62] = -0.011329
y[63] = -0.012025
y[64] = -0.012674
y[65] = -0.013274
y[66] = -0.013823
y[67] = -0.014318
y[68] = -0.014757
y[69] = -0.015140
y[70] = -0.015463
y[71] = -0.015727
y[72] = -0.015929
y[73] = -0.016070
y[74] = -0.016148
y[75] = -0.016164
y[76] = -0.016116
y[77] = -0.016006
y[78] = -0.015834
y[79] = -0.015601
y[80] = -0.015307
y[81] = -0.014953
y[82] = -0.014542
y[83] = -0.014074
y[84] = -0.013551
y[85] = -0.012976
y[86] = -0.012351
y[87] = -0.011678
y[88] = -0.010960
y[89] = -0.010200
y[90] = -0.009400
y[91] = -0.008565
y[92] = -0.007696
y[93] = -0.006799
y[94] = -0.005875
y[95] = -0.004929
y[96] = -0.003965
y[97] = -0.002986
y[98] = -0.001996
y[99] = -0.001000
y[100] = -0.000000
```



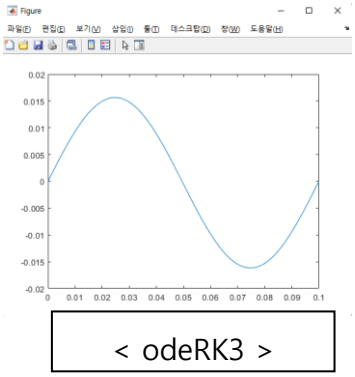
< odeEU >



< odeEM >



< odeRK2 >



< odeRK3 >

```
441  void ode(double myfunc(const double t, const double y), double y[], double t0, double tf, double h, int method) {
442      int y0 = 0;
443
444      if (method == EU)
445      {
446          printf("₩n==================================================₩n");
447          printf("                    odeEU                         ");
448          printf("₩n==================================================₩n");
449          odeEU(myfunc, y, t0, tf, h);
450      }
451      else if (method == EM)
452      {
453          printf("₩n==================================================₩n");
454          printf("                    odeEM                         ");
455          printf("₩n==================================================₩n");
456          odeEM(myfunc, y, t0, tf, h);
457      }
458      else if (method ==RK2)
459      {
460          printf("₩n==================================================₩n");
461          printf("                    odeRK2                        ");
462          printf("₩n==================================================₩n");
463          odeRK2(myfunc, y, t0, tf, h, y0);
464      }
465      else if (method == RK3)
466      {
467          printf("₩n==================================================₩n");
468          printf("                    odeRK3                        ");
469          printf("₩n==================================================₩n");
470          odeRK2(myfunc, y, t0, tf, h, y0);
471      }
472      else
473          printf("Error: Wrong method₩n");
474          system("pause");
475  }
```