

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

$$y_{i+1} = y_i + (C_1 K_1 + C_2 K_2)h$$

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

For RK3, use classical third-order Runge-Kutta

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

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

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

PSEUDOCODE AND SOURCE CODE

Euler

for $t = t_0$ to t_f , $t += h$
 $slope = f_{unc}(t, y_i)$
 $y_{i+1} = slope * h$
end

odeEM

for $t = t_0$ to t_f , $t += h$
 $slope1 = f_{unc}(t, y_i)$
 $y_{iE} = slope1 * h$
 $slope2 = f_{unc}(t+h, y_{iE})$
 $y_{i+1} = 0.5 * (slope1 + slope2) * h$
end

odeRK2

for $t = t_0$ to t_f , $t += h$
 $K1 = f_{unc}(t, y_i)$
 $y_{iE} = K1 * h$
 $K2 = f_{unc}(t + \alpha * h, y_{iE})$
 $y_{i+1} = 0.5 * (K1 + K2) * h$
end

odeRK3

for $t = t_0$ to t_f , $t += h$
 $K1 = f_{unc}(t, y_i)$
 $K2 = f_{unc}(t + \alpha_1 * h, y_i + \beta_1 * K1 * h)$
 $K3 = f_{unc}(t + \alpha_2 * h, y_i + \beta_2 * K1 * h + \beta_3 * K2 * h)$
 $y_{i+1} = \frac{K1 + 4K2 + K3}{6} * h$
end

```

338 void odeEU(double myfunc(const double t, const double y), double y[], double t0, double tf, double h) {
339
340     double t;
341     double y1 = 0;
342     double slope = 0;
343     int k = 0;
344     y[k] = y1;
345
346     printf("y[%d] = %f\n", k, y[k]);
347
348     for (t = t0; t <= tf; t += h) {
349         slope = myfunc(t, y1);
350         y1 = y1 + slope * h;
351
352         k++;
353         y[k] = y1;
354         printf("%f\n", y[k]);
355     }
356 }
357
358 void odeEM(double myfunc(const double t, const double y), double y[], double t0, double tf, double h) {
359
360     double y1 = 0;
361     double yIE = 0;
362     double slope1 = 0;
363     double slope2 = 0;
364     int k = 0;
365     y[k] = y1;
366
367     printf("y[%d] = %f\n", k, y[k]);
368
369     for (double t = t0; t <= tf; t += h) {
370         slope1 = myfunc(t, y1);
371         yIE = yIE + slope1 * h;
372         slope2 = myfunc(t + h, yIE);
373         y1 = y1 + 0.5 * (slope1 + slope2) * h;
374
375         k++;
376         y[k] = y1;
377         printf("%f\n", y[k]);
378     }
379 }
380
381 void odeRK2(double myfunc(const double t, const double y), double y[], double t0, double tf, double h, double y0) {
382
383     int alpha = 1;
384     int beta = alpha;
385     double C2 = 1 / 2 * alpha;
386     double C1 = 1 - C2;
387     double K1 = 0;
388     double K2 = 0;
389     double y1 = y0;
390     double yIE = y0;
391     int k = 0;
392     y[k] = y0;
393
394     printf("y[%d] = %f\n", k, y[k]);
395
396     for (double t = t0; t <= tf; t += h) {
397         K1 = myfunc(t, y1);
398         yIE = yIE + beta * K1 * h;
399         K2 = myfunc(t + alpha * h, yIE);
400
401         y1 = y1 + 0.5 * (K1 + K2) * h;
402
403         k++;
404         y[k] = y1;
405         printf("%f\n", y[k]);
406     }
407 }
408
409 void odeRK3(double myfunc(const double t, const double y), double y[], double t0, double tf, double h, double y0) {
410
411     int alpha1 = 1;
412     int alpha2 = 1;
413     int alpha3 = 1;
414     double beta1 = 0.5;
415     double beta2 = 1;
416     double beta3 = 2;
417     double C1 = 1 / 6;
418     double C2 = 4 / 6;
419     double C3 = 1 / 6;
420     double C4 = 1 / 2;
421
422     double K1 = 0;
423     double K2 = 0;
424     double K3 = 0;
425
426     double y1 = y0;
427     int k = 0;
428     y[k] = y0;
429
430     printf("y[%d] = %f\n", k, y[k]);
431
432     for (int t = t0; t <= tf; t += h) {
433         K1 = myfunc(t, y1);
434         K2 = myfunc(t + alpha1 * h, y1 + beta1 * K1 * h);
435         K3 = myfunc(t + alpha2 * h, y1 + beta2 * K1 * h + beta3 * K2 * h);
436
437         y1 = y1 + (K1 + 4 * K2 + K3) * h / 6;
438
439         k++;
440         y[k] = y1;
441         printf("%f\n", y[k]);
442     }
443 }
444
445 #include "../..//include/myNP.h"
446
447 double myfunc(const double t, const double y) {
448
449     int main(int argc, char* argv[]) {
450
451         double t0 = 0;
452         double tf = 0.1;
453         double h = 0.001;
454         double y[101] = { 0 };
455
456         ode(myfunc, y, t0, tf, h, EU);
457         ode(myfunc, y, t0, tf, h, EM);
458         ode(myfunc, y, t0, tf, h, RK2);
459         ode(myfunc, y, t0, tf, h, RK3);
460
461         system("pause");
462         return 0;
463     }
464 }
465
466 double myfunc(const double t, const double y) {
467
468     double tau = 1;
469     double T = 1 / tau;
470     double f = 10;
471     double Va = 1;
472     double w = 2 * PI * f;
473
474     double F = -T * y + T * Va * cos(w * t);
475
476     return F;
477 }

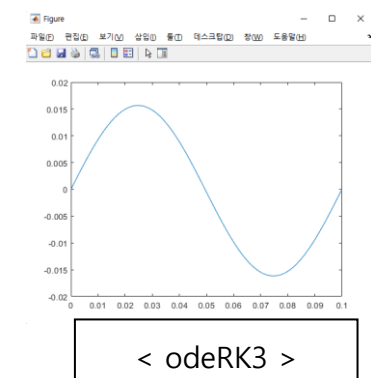
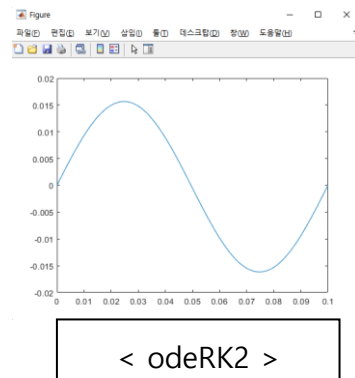
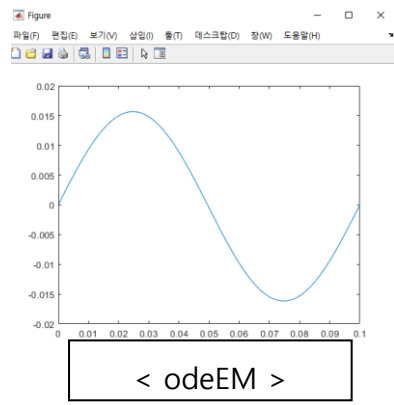
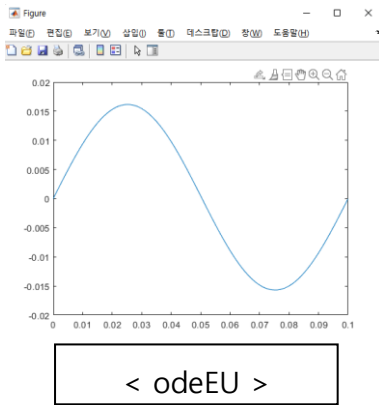
```

odeEuler	odeEM	odeRK2
y[0] = 0.000000	y[0] = 0.000000	y[0] = 0.000000
y[1] = 0.001000	y[1] = 0.000999	y[1] = 0.000999
y[2] = 0.001997	y[2] = 0.001992	y[2] = 0.001992
y[3] = 0.002987	y[3] = 0.002977	y[3] = 0.002977
y[4] = 0.003966	y[4] = 0.003949	y[4] = 0.003949
y[5] = 0.004931	y[5] = 0.004904	y[5] = 0.004904
y[6] = 0.005877	y[6] = 0.005839	y[6] = 0.005839
y[7] = 0.006801	y[7] = 0.006750	y[7] = 0.006750
y[8] = 0.007699	y[8] = 0.007633	y[8] = 0.007633
y[9] = 0.008568	y[9] = 0.008486	y[9] = 0.008486
y[10] = 0.009403	y[10] = 0.009303	y[10] = 0.009303
y[11] = 0.010203	y[11] = 0.010083	y[11] = 0.010083
y[12] = 0.010963	y[12] = 0.010823	y[12] = 0.010823
y[13] = 0.011681	y[13] = 0.011518	y[13] = 0.011518
y[14] = 0.012354	y[14] = 0.012167	y[14] = 0.012167
y[15] = 0.012979	y[15] = 0.012767	y[15] = 0.012767
y[16] = 0.013554	y[16] = 0.013316	y[16] = 0.013316
y[17] = 0.014076	y[17] = 0.013811	y[17] = 0.013811
y[18] = 0.014544	y[18] = 0.014251	y[18] = 0.014251
y[19] = 0.014955	y[19] = 0.014633	y[19] = 0.014633
y[20] = 0.015309	y[20] = 0.014957	y[20] = 0.014957
y[21] = 0.015602	y[21] = 0.015220	y[21] = 0.015220
y[22] = 0.015835	y[22] = 0.015423	y[22] = 0.015423
y[23] = 0.016007	y[23] = 0.015563	y[23] = 0.015563
y[24] = 0.016116	y[24] = 0.015641	y[24] = 0.015641
y[25] = 0.016163	y[25] = 0.015657	y[25] = 0.015657
y[26] = 0.016147	y[26] = 0.015610	y[26] = 0.015610
y[27] = 0.016068	y[27] = 0.015500	y[27] = 0.015500
y[28] = 0.015926	y[28] = 0.015328	y[28] = 0.015328
y[29] = 0.015723	y[29] = 0.015094	y[29] = 0.015094
y[30] = 0.015459	y[30] = 0.014800	y[30] = 0.014800
y[31] = 0.015134	y[31] = 0.014446	y[31] = 0.014446
y[32] = 0.014751	y[32] = 0.014035	y[32] = 0.014035
y[33] = 0.014310	y[33] = 0.013567	y[33] = 0.013567
y[34] = 0.013814	y[34] = 0.013044	y[34] = 0.013044
y[35] = 0.013265	y[35] = 0.012469	y[35] = 0.012469
y[36] = 0.012664	y[36] = 0.011844	y[36] = 0.011844
y[37] = 0.012014	y[37] = 0.011171	y[37] = 0.011171
y[38] = 0.011317	y[38] = 0.010453	y[38] = 0.010453
y[39] = 0.010577	y[39] = 0.009693	y[39] = 0.009693
y[40] = 0.009796	y[40] = 0.008894	y[40] = 0.008894
y[41] = 0.008977	y[41] = 0.008058	y[41] = 0.008058
y[42] = 0.008123	y[42] = 0.007190	y[42] = 0.007190
y[43] = 0.007239	y[43] = 0.006292	y[43] = 0.006292
y[44] = 0.006327	y[44] = 0.005368	y[44] = 0.005368
y[45] = 0.005391	y[45] = 0.004422	y[45] = 0.004422
y[46] = 0.004434	y[46] = 0.003458	y[46] = 0.003458
y[47] = 0.003461	y[47] = 0.002479	y[47] = 0.002479
y[48] = 0.002476	y[48] = 0.001489	y[48] = 0.001489
y[49] = 0.001481	y[49] = 0.000493	y[49] = 0.000493
y[50] = 0.000482	y[50] = -0.000507	y[50] = -0.000507
y[51] = -0.000519	y[51] = -0.001506	y[51] = -0.001506
y[52] = -0.001516	y[52] = -0.002499	y[52] = -0.002499
y[53] = -0.002507	y[53] = -0.003483	y[53] = -0.003483
y[54] = -0.003487	y[54] = -0.004455	y[54] = -0.004455
y[55] = -0.004452	y[55] = -0.005411	y[55] = -0.005411
y[56] = -0.005399	y[56] = -0.006346	y[56] = -0.006346
y[57] = -0.006323	y[57] = -0.007257	y[57] = -0.007257
y[58] = -0.007221	y[58] = -0.008140	y[58] = -0.008140
y[59] = -0.008090	y[59] = -0.008992	y[59] = -0.008992
y[60] = -0.008927	y[60] = -0.009810	y[60] = -0.009810
y[61] = -0.009727	y[61] = -0.010590	y[61] = -0.010590
y[62] = -0.010488	y[62] = -0.011329	y[62] = -0.011329
y[63] = -0.011206	y[63] = -0.012025	y[63] = -0.012025
y[64] = -0.011879	y[64] = -0.012674	y[64] = -0.012674
y[65] = -0.012505	y[65] = -0.013274	y[65] = -0.013274
y[66] = -0.013080	y[66] = -0.013823	y[66] = -0.013823
y[67] = -0.013603	y[67] = -0.014318	y[67] = -0.014318
y[68] = -0.014071	y[68] = -0.014757	y[68] = -0.014757
y[69] = -0.014483	y[69] = -0.015140	y[69] = -0.015140
y[70] = -0.014836	y[70] = -0.015463	y[70] = -0.015463
y[71] = -0.015131	y[71] = -0.015727	y[71] = -0.015727
y[72] = -0.015364	y[72] = -0.015929	y[72] = -0.015929
y[73] = -0.015536	y[73] = -0.016070	y[73] = -0.016070
y[74] = -0.015646	y[74] = -0.016148	y[74] = -0.016148
y[75] = -0.015693	y[75] = -0.016164	y[75] = -0.016164
y[76] = -0.015677	y[76] = -0.016116	y[76] = -0.016116
y[77] = -0.015599	y[77] = -0.016006	y[77] = -0.016006
y[78] = -0.015458	y[78] = -0.015834	y[78] = -0.015834
y[79] = -0.015255	y[79] = -0.015601	y[79] = -0.015601
y[80] = -0.014991	y[80] = -0.015307	y[80] = -0.015307
y[81] = -0.014667	y[81] = -0.014953	y[81] = -0.014953
y[82] = -0.014285	y[82] = -0.014542	y[82] = -0.014542
y[83] = -0.013844	y[83] = -0.014074	y[83] = -0.014074
y[84] = -0.013349	y[84] = -0.013551	y[84] = -0.013551
y[85] = -0.012800	y[85] = -0.012976	y[85] = -0.012976
y[86] = -0.012199	y[86] = -0.012351	y[86] = -0.012351
y[87] = -0.011549	y[87] = -0.011678	y[87] = -0.011678
y[88] = -0.010853	y[88] = -0.010960	y[88] = -0.010960
y[89] = -0.010114	y[89] = -0.010200	y[89] = -0.010200
y[90] = -0.009333	y[90] = -0.009400	y[90] = -0.009400
y[91] = -0.008515	y[91] = -0.008565	y[91] = -0.008565
y[92] = -0.007662	y[92] = -0.007696	y[92] = -0.007696
y[93] = -0.006778	y[93] = -0.006799	y[93] = -0.006799
y[94] = -0.005866	y[94] = -0.005875	y[94] = -0.005875
y[95] = -0.004931	y[95] = -0.004929	y[95] = -0.004929
y[96] = -0.003975	y[96] = -0.003965	y[96] = -0.003965
y[97] = -0.003002	y[97] = -0.002986	y[97] = -0.002986
y[98] = -0.002017	y[98] = -0.001996	y[98] = -0.001996
y[99] = -0.001023	y[99] = -0.001000	y[99] = -0.001000
y[100] = -0.000023	y[100] = -0.000000	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] = 0.009308
y[11] = 0.010093
y[12] = 0.010832
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] = -0.001986
y[99] = -0.001000
y[100] = -0.000000

```



```

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\n");
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\n");
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\n");
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\n");
469         printf("\n===== \n");
470         odeRK3(myfunc, y, t0, tf, h, y0);
471     }
472     else
473     {
474         printf("Error: Wrong method\n");
475         system("pause");
476     }

```