ASSIGNMENT: 2nd 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_ode2_Name_ID.cpp, (2) myNP.h, (3) myNP.cpp**
- All the functions you have created should be updated in myNP.h and myNP.cpp

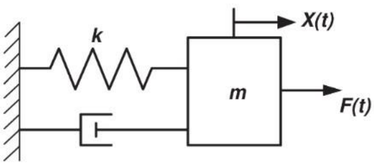
2nd order ODE

Problem

A mass-spring-damper (m-c-k) system is a 2nd order ODE

$$m\frac{d^2y}{dt^2}+c\frac{dy}{dt}+ky(t)=F(t)$$

where F(t) is the input force, y(t) is the displacement.



Create a program that gives the values of y(t), dy/dt(t) for the following response

1. Harmonic Response (F(t)=Acos(2*pi*f*t))
2. Free vibration (i.e. F(t)=0) from the initial condition
3. Step Response (F(t)=A)

- Parameters/Initial condition m=1kg, k=6.9N/m,

  c=7 N/m/s, A=2 N, f=5Hz,

  t=0 to 1 sec, h=0.01 sec

  Initial Condition: y(0)=0.0 m, dy/dt $_{|t=0}$=0.2 m/s

Exercise

- Write a function for the governing equation for m-c-k system.

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

 // EXERCISE: MODIFY HERE
dYdt[1] =
(-k * Y[0] - c * Y[1] + Fin) / m;

}
```

HINT:

$$\begin{cases} \dot{y} = z(t) \\ \ddot{y} = \dot{z} = \dfrac{1}{m}\left(-ky(t) - cz(t) + u(t)\right) \end{cases} \rightarrow \begin{bmatrix} \dot{y} \\ \dot{z} \end{bmatrix} = \frac{1}{m}\begin{bmatrix} z(t) \\ -ky(t) - cz(t) + u(t) \end{bmatrix}$$

```
int main(int argc, char* argv[]) {

    int t0 = 0;
    double tf = 1;
    double h = 0.01;
    double y1[200] = { 0 };
    double y2[200] = { 0 };
    double y1_init = 0;
    double y2_init = 0.2;
    double N = (tf - t0) / h + 1;
    double t = 0;

    sys2RK2(mckfunc, y1, y2, t0, tf, h, y1_init, y2_init);

    for (int k = 0; k < N; k++) {
        printf("t = %f ₩n y(t) = %f ₩n z(t) = %f ₩n₩n", t + k * h, y1[k], y2[k]);
    }
    system("pause");
    return 0;
}


void mckfunc(const double t, const double Y[], double dYdt[]) {
    double m = 1;
    double c = 7;
    double k = 6.9;
    double f = 5;

    double Fin = 2 * cos(2 * PI * f * t);
    //double Fin = A;
    //double Fin = 0;
                                                        // Y[0]  = y(t) , Y[1] = z(t)
    dYdt[0] = Y[1];                                     // dydt == z(t)
    dYdt[1] = (-k * Y[0] - c * Y[1] + Fin) / m;        // dzdt
}
```

- Write a pseudocode of RK2 for 2nd order system

```
For (i to N-1 i++)
  tempY = y1[i]
  tempY = y2[i]
    mckfunc(t, tempY, K1)
    tempY[0] += K1[0]*h
   tempY[1] += K1[1]*h

   mckfunc(t+h, tempY, K2)
   y1[i+1] = y1[i] + 0.5*(K1[0]+K2[0])*h
   y2[i+1] = y2[i] + 0.5*(K1[1]+K2[1])*h
   t+=th
end
```

- Create a function of RK2 for 2nd order system

```
void sys2RK2(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);
```

```
void sys2RK2(void mckfunc(const double t, const double Y[], double dYdt[]), double y1[], double y2[], double t0, double tf, double h, double y1_init,
    double y2_init) {
    double K1[2] = { 0 };
    double K2[2] = { 0 };
    double tempY[2] = { 0 };
    double tempZ[2] = { 0 };
    double N = (tf - t0) / h + 1;
    double t = t0;
    double K1_0 = 0;                        // K1 = { K1_0 ; K1_1 }
    double K1_1 = 0;
    double K2_0 = 0;                        // K2 = { K2_0 ; K2_0 }
    double K2_1 = 0;

    y1[0] = y1_init;                      // y(t)                Y = { y(t) ; z(t) }
    y2[0] = y2_init;                      // z(t)

    for (int i = 0; i < N - 1; i++) {

        // slope 1 :                        f(t[i], y[i], z[i])
        tempY[0] = y1[i];                   // Y[0] = y(t)
        tempY[1] = y2[i];                   // Y[1] = z(t)

        mckfunc(t, tempY, K1);              // *** K1 = f(t[i], y[i], z[i]) ***

        // saving the calculated value (K1[] = dydt[])
        K1_0 = K1[0];                       // K1_0 = dydt = z(t)
        K1_1 = K1[1];                       // K1_1 = z'(t)

        // slope 2

        tempY[0] += K1_0 * h;               // y(t)
        tempY[1] += K1_1 * h;               // z(t)

        mckfunc(t + h, tempY, K2);          // *** K2 = f(t[i] + h, y[i] + K1 * h, z[i] + K1 * h) ***

        // saving the calculated value (K2[] = dydt[])
        K2_0 = K2[0];                       // z(t)
        K2_1 = K2[1];                       // z'(t)

        y1[i + 1] = y1[i] + (0.5 * K1_0 + 0.5 * K2_0) * h;          // y(t)
        y2[i + 1] = y2[i] + (0.5 * K1_1 + 0.5 * K2_1) * h;          // z(t)

        t = t + h;                          // t[i+1] = t[i] + h
    }

}
```

- Compare the answer with MATLAB's ODE solver by copy-pasting your outputs in MATLAB and plotting results on the same graph.

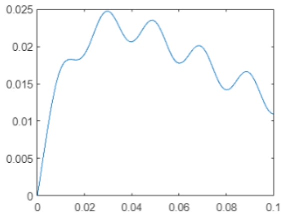
```
t = 0.000000
 y(t) = 0.000000
 z(t) = 0.200000

t = 0.010000
 y(t) = 0.002030
 z(t) = 0.205232

t = 0.020000
 y(t) = 0.004105
 z(t) = 0.208097

t = 0.030000
 y(t) = 0.006193
 z(t) = 0.207097

t = 0.040000
 y(t) = 0.008248
 z(t) = 0.201180

t = 0.050000
 y(t) = 0.010217
 z(t) = 0.189846

t = 0.060000
 y(t) = 0.012046
 z(t) = 0.173186

t = 0.070000
 y(t) = 0.013682
 z(t) = 0.151874

t = 0.080000
 y(t) = 0.015084
 z(t) = 0.127094

t = 0.090000
 y(t) = 0.016224
 z(t) = 0.100427

t = 0.100000
 y(t) = 0.017093
 z(t) = 0.073683

t = 0.110000
 y(t) = 0.017698
 z(t) = 0.048732

t = 0.120000
 y(t) = 0.018067
 z(t) = 0.027310

t = 0.130000
 y(t) = 0.018243
 z(t) = 0.010851

t = 0.140000
 y(t) = 0.018283
 z(t) = 0.000343

t = 0.150000
 y(t) = 0.018249
 z(t) = -0.003772
```

```
t = 0.860000
 y(t) = 0.016143
 z(t) = 0.036559

t = 0.870000
 y(t) = 0.016460
 z(t) = 0.024250

t = 0.880000
 y(t) = 0.016629
 z(t) = 0.007951

t = 0.890000
 y(t) = 0.016619
 z(t) = -0.010730

t = 0.900000
 y(t) = 0.016415
 z(t) = -0.029953

t = 0.910000
 y(t) = 0.016020
 z(t) = -0.047823

t = 0.920000
 y(t) = 0.015458
 z(t) = -0.062578

t = 0.930000
 y(t) = 0.014768
 z(t) = -0.072760

t = 0.940000
 y(t) = 0.014002
 z(t) = -0.077360

t = 0.950000
 y(t) = 0.013220
 z(t) = -0.075914

t = 0.960000
 y(t) = 0.012482
 z(t) = -0.068550

t = 0.970000
 y(t) = 0.011848
 z(t) = -0.055975

t = 0.980000
 y(t) = 0.011362
 z(t) = -0.039407

t = 0.990000
 y(t) = 0.011059
 z(t) = -0.020453

t = 1.000000
 y(t) = 0.010953
 z(t) = -0.000956
```

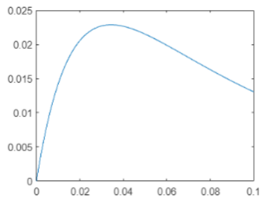
F=Acos(2*pi*f*t)

-

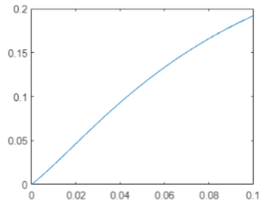
My Result



F=Acos(2*pi*f*t)

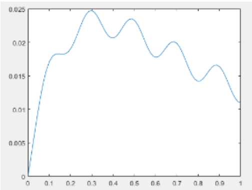


F=0

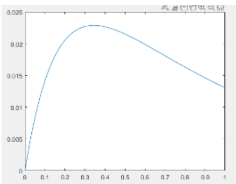


F=A

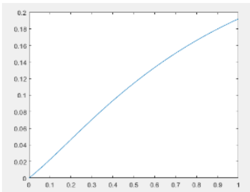
Matlab ODE solver



F=Acos(2*pi*f*t)



F=0



F=A