科学计算第七次作业

黄伟 123071910077

日期：2023 年 12 月 27 日 9:00 提交

Question 0.1. Let $\theta = 1/4$ and $T = 16$. Please write C/C++ computer programs to solve the following initial value problem of ordinary differential equation

$$\begin{cases} \frac{\mathrm{d}u(t)}{\mathrm{d}t} = u(1-u)(u-\theta) \quad \text{for } 0 < t < T \\ u(0) = \frac{3}{10} \end{cases} \tag{1}$$

with the forward Euler, backward Euler and trapezoidal methods, respectively. Let $\Delta t = T/N > 0$ be the timestep and $\{t_n = n\Delta t\}_{n=0}^{N}$ be the discrete times. Let u_n be the discrete approximation to the value of the exact solution $u(t)$ at time $t = t_n$. Let $w_{\Delta t}$ be the value of the numerical solution u_n at the discrete time $t_n = 1$. For each method, run your program with different time steps $\Delta t's$ to generate a table as the following one.

Δt	1/5	1/10	1/20	1/40
$w_{\Delta t}$				
$w_{\Delta t} - w_{2\Delta t}$				

Verify the accuracy order of the numerical methods by checking the relation of the data $w_{\Delta t} - w_{2\Delta t}$ with the timestep Δt in each table. Please generate a t_n v.s. u_n plot of the numerical solution for each method with $\Delta t = 1/40$.

解. 记 $f(x) := x(1-x)(x-\theta)$.

前向 Euler

$$\begin{cases} u(t_{n+1}) = u(t_n) + \Delta t \cdot f(u(t_n)), \\ u(0) = \frac{3}{10}. \end{cases} \tag{2}$$

后向 Euler

$$\begin{cases} u(t_{n+1}) = u(t_n) + \Delta t \cdot f(u(t_{n+1})), \\ u(0) = \frac{3}{10}. \end{cases} \tag{3}$$

梯形

$$\begin{cases} u(t_{n+1}) = u(t_n) + \frac{\Delta t}{2} \cdot [f(u(t_n)) + f(u(t_{n+1}))], \\ u(0) = \frac{3}{10}. \end{cases} \tag{4}$$

Question 0.2. Solve the initial value problem

$$\begin{cases} u'(t) = u(t) - \frac{2t}{u(t)} \quad \text{for } 0 < t < 4 \\ u(0) = 1 \end{cases} \tag{5}$$

by the classic four-stage Runge-Kutta method with different timestep $\Delta t \in \{1/5, 1/10, 1/20, 1/40\}$. The exact solution to the initial value problem reads $u(t) = \sqrt{1+2t}$. Let $e(\Delta t) = u(t_n) - u_n$ be the solution error at time $t_n = 1$. Generate a table as the following one.

解. 经典 Runge-Kutta 法可以写为

$$\begin{cases} k_1 = f(t_n, u_n), \\ k_2 = f(t_n + \frac{1}{2}\Delta t, u_n + \frac{\Delta t}{2} k_1), \\ k_3 = f(t_n + \frac{1}{2}\Delta t, u_n + \frac{\Delta t}{2} k_2), \\ k_4 = f(t_n + \Delta t, u_n + \Delta t k_3), \\ u_{n+1} = u_n + \frac{\Delta t}{6}(k_1 + 2k_2 + 2k_3 + k_4). \end{cases} \tag{6}$$

Δt	1/5	1/10	1/20	1/40
$e(\Delta t)$				
$e(2\Delta t)/e(\Delta t)$				

Question 0.3. Solve the initial value problem

$$\begin{cases} u'(t) = -10u(t) + 9e^{-t} \quad \text{for } 0 < t < 1 \\ u(0) = 1 \end{cases} \tag{7}$$

whose exact solution reads $u(t) = e^{-t}$, by any time integration method of your own choice with the timestep size $\Delta t = 1/10$, $1/20$ or $1/40$. Validate your code and explain on what you observe.

解.

Question 0.4. Please write C/C++ computer programs to solve the pendulum equation

$$\theta''(t) + 16\sin(\theta(t)) = 0, \theta(0) = \frac{\pi}{6}, \theta'(0) = 0 \tag{8}$$

for $t \in [0, 4]$ by the forward Euler, backward Euler and trapezoidal methods as well as a higher order method of your own choice. Denote by θ_n the finite difference approximation of $\theta(t_n)$ at the discrete time $t_n = n\Delta t$. For each integration method, run your computer program with the timestep $\Delta t = 0.05$ and generate a t_n versus θ_n plot of the numerical solution.

解. 令 $\theta_1(t) = \theta'(t)$, 方程可以表示为

$$\begin{cases} \theta'(t) = \theta_1(t), \\ \theta_1'(t) = -16\sin\theta(t). \end{cases} \tag{9}$$

初值为 $\theta(0) = \frac{\pi}{6}, \theta_1(0) = 0$.

(1) 前向 Euler 方法

$$\begin{cases} \theta(t_{n+1}) = \theta(t_n) + \theta_1(t_n) \cdot \Delta t, \\ \theta_1(t_{n+1}) = \theta_1(t_n) - 16\sin\theta(t_n) \cdot \Delta t. \end{cases} \tag{10}$$

(2) 后向 Euler 方法

$$\begin{cases} \theta(t_{n+1}) = \theta(t_n) + \theta_1(t_{n+1}) \cdot \Delta t, \\ \theta_1(t_{n+1}) = \theta_1(t_n) - 16 \sin\theta(t_{n+1}) \cdot \Delta t. \end{cases} \tag{11}$$

(3) 梯形方法

$$\begin{cases} \theta(t_{n+1}) = \theta(t_n) + [\theta_1(t_n) + \theta_1(t_{n+1}) \cdot \frac{\Delta t}{2}, \\ \theta_1(t_{n+1}) = \theta_1(t_n) - 16[\sin\theta(t_n) + \sin\theta(t_{n+1})] \cdot \frac{\Delta t}{2}. \end{cases} \tag{12}$$

Question 0.5. Suppose that $f(v)$ is a smooth function of $v \in \mathbb{R}$. Let us consider numerically solving the ordinary differential equation

$$u'(t) = f(u(t)), \tag{13}$$

subject to some initial condition. Let $\Delta t > 0$ be the timestep and $\{t_n = n\Delta t\}$ be the discrete times. Assume u_n is a finite difference approximation to the value of the exact solution $u(t)$ at time $t = t_n$. Show that the global solution error $e_n = u(t_n) - u_n$ by each method below has second order accuracy.

(a) the **(explicit) midpoint method** takes the form

$$u_{n+1} = u_n + \Delta t f\left(u_n + \frac{\Delta t}{2} f(u_n)\right) \tag{14}$$

for $n = 0, 1, 2, \cdots$.

(b) the **implicit midpoint method** takes the form

$$u_{n+1} = u_n + \Delta t f\left(\frac{1}{2}(u_n + u_{n+1})\right) \tag{15}$$

for $n = 0, 1, 2, \cdots$.

(c) the **modified Euler method** takes the form

$$u_{n+1} = u_n + \frac{\Delta t}{2}[f(u_n) + f(u_n + \Delta t f(u_n))] \tag{16}$$

for $n = 0, 1, 2, \cdots$.

解. (a) Taylor 展开

$$u(t_{n+1}) = u(t_n) + \Delta t u'(t_n) + \frac{(\Delta t)^2}{2!} u''(t_n) + \frac{(\Delta t)^3}{3!} u'''(\xi_n). \tag{17}$$

其中 $\xi_n \in (t_n, t_{n+1})$. 另一边

$$f\left(u_n + \frac{\Delta t}{2} f(u_n)\right) = f(u_n) + f'(u_n) \cdot \frac{\Delta t}{2} f(u_n) + \frac{f''(\eta_n)}{2} \cdot \left(\frac{\Delta t}{2} f(u_n)\right)^2. \tag{18}$$

其中 $\eta_n \in (u_n, u_n + \frac{\Delta t}{2} f(u_n))$, 设 $u(\theta_n) = \eta_n$. 同时我们有 $u_n = u(t_n)$, 以及 $u'(t) = f(u(t))$, 所以

$$u''(t) = u'(t) \cdot f'(u(t)) = f(u(t)) \cdot f'(u(t)). \tag{19}$$

将定义式 (14) 以及上述公式代入 e_n 得到

$$\begin{aligned}
e_{n+1} &= u(t_{n+1}) - u_{n+1} \\
&= \left(u(t_n) + \Delta t u'(t_n) + \frac{(\Delta t)^2}{2!}u''(t_n) + \frac{(\Delta t)^3}{3!}u'''(\xi_n)\right) - \left(u_n + \Delta t f\left(u_n + \frac{\Delta t}{2}f(u_n)\right)\right) \\
&= \Delta t u'(t_n) + \frac{(\Delta t)^2}{2!}u''(t_n) + \frac{(\Delta t)^3}{3!}u'''(\xi_n) - \Delta t \cdot \left(f(u_n) + f'(u_n)\cdot\frac{\Delta t}{2}f(u_n) + \frac{f''(\eta_n)}{2}\cdot\left(\frac{\Delta t}{2}f(u_n)\right)^2\right) \\
&= \frac{(\Delta t)^3}{24}\cdot\left(4u'''(\xi_n) - 3f''(\eta_n)(f(u_n))^2\right) \\
&= \frac{(\Delta t)^3}{24}\cdot\left(4u'''(\xi_n) - 3\cdot\frac{u'''(\theta_n)u'(\theta_n) - (u''(\theta_n))^2}{(u'(\theta_n))^3}\cdot(u'(t_n))^2\right)
\end{aligned}$$

所以是 2 阶的.

(b) Taylor 展开

$$u(t_{n+1}) = u(t_n) + \Delta t u'(t_n) + \frac{(\Delta t)^2}{2!}u''(t_n) + \frac{(\Delta t)^3}{3!}u'''(\xi_n). \tag{20}$$

其中 $\xi_n \in (t_n, t_{n+1})$. 另一边

$$f\left(\frac{1}{2}(u_n + u_{n+1})\right) = f(u_n) + f'(u_n)\cdot\frac{u_{n+1}-u_n}{2} + \frac{f''(\eta_n)}{2}\cdot\left(\frac{u_{n+1}-u_n}{2}\right)^2, \tag{21}$$

其中 $\eta_n \in (u_n, u_{n+1})$. 此外还有

$$u_{n+1} - u_n = \Delta t \cdot f\left(\frac{1}{2}(u_n + u_{n+1})\right), \tag{22}$$

所以

$$\begin{aligned}
&f\left(\frac{1}{2}(u_n + u_{n+1})\right) \\
=&f(u_n) + f'(u_n)\cdot\frac{\Delta t}{2}\cdot f\left(\frac{1}{2}(u_n+u_{n+1})\right) + \frac{f''(\eta_n)}{2}\cdot\frac{(\Delta t)^2}{4}\cdot f\left(\frac{1}{2}(u_n+u_{n+1})\right) \\
=&f(u_n) + f'(u_n)\cdot\frac{\Delta t}{2}\cdot\left(f(u_n) + f'(u_n)\cdot\frac{\Delta t}{2}\cdot f\left(\frac{1}{2}(u_n+u_{n+1})\right) + \frac{f''(\eta_n)}{2}\cdot\frac{(\Delta t)^2}{4}\cdot f\left(\frac{1}{2}(u_n+u_{n+1})\right)\right) \\
&+ \frac{f''(\eta_n)}{2}\cdot\frac{(\Delta t)^2}{4}\cdot f\left(\frac{1}{2}(u_n+u_{n+1})\right) \\
=&f(u_n) + \frac{1}{2}f'(u_n)f(u_n)\Delta t + \left(2(f'(u_n))^2 + f''(\eta_n)\right)f\left(\frac{1}{2}(u_n+u_{n+1})\right)\cdot\frac{(\Delta t)^2}{8} + \mathcal{O}((\Delta t)^3)
\end{aligned}$$

于是

$$u_{n+1} = u_n + \Delta t\cdot\left(f(u_n) + \frac{1}{2}f'(u_n)f(u_n)\Delta t + \mathcal{O}((\Delta t)^2)\right) \tag{23}$$

从而

$$e_{n+1} = u(t_{n+1}) - u_{n+1} = \mathcal{O}((\Delta t)^3). \tag{24}$$

所以 2 阶.

(c) 同样 Taylor 展开

$$u(t_{n+1}) = u(t_n) + \Delta t u'(t_n) + \frac{(\Delta t)^2}{2!}u''(t_n) + \frac{(\Delta t)^3}{3!}u'''(\xi_n). \tag{25}$$

其中 $\xi_n \in (t_n, t_{n+1})$. 另一边

$$f(u_n + \Delta t f(u_n)) = f(u_n) + f'(u_n)\cdot\Delta t f(u_n) + \mathcal{O}((\Delta t)^2), \tag{26}$$

所以

$$\begin{aligned} e_{n+1} &= u(t_{n+1}) - u_{n+1} \\ &= u(t_n) + \Delta t u'(t_n) + \frac{(\Delta t)^2}{2!}u''(t_n) + \frac{(\Delta t)^3}{3!}u'''(\xi_n) \\ &\quad - u_n - \frac{\Delta t}{2}\left[f(u_n) + f(u_n) + f'(u_n)\cdot \Delta t f(u_n) + \mathcal{O}((\Delta t)^2)\right] \\ &= \mathcal{O}((\Delta t)^3). \end{aligned} \tag{27}$$

所以是 2 阶精度.

Question 0.6. Let $t_n = n\Delta t$ for $n = 0, 1, 2, \cdots$. Show that the approximate solution $u_n \approx u(t_n)$ generated by the Runge-Kutta method below

$$\begin{cases} K_0 = f(u_n), \\ K_1 = f(u_n + \gamma\Delta t K_0), \\ K_2 = f(u_n + (1-\gamma)\Delta t K_1), \\ u_{n+1} = u_n + \frac{\Delta t}{2}(K_1 + K_2). \end{cases} \tag{28}$$

for the ODE,

$$\frac{\mathrm{d}u(t)}{\mathrm{d}t} = f(u(t)), \quad t > 0, \tag{29}$$

has second-order accuracy with any parameter $\gamma \in (0, \frac{1}{2})$, i.e. ,

$$e_n = u(t_n) - u_n = O(\Delta t^2), \tag{30}$$

provided that $u_0 = u(0)$ and the slope function $f(u)$ is sufficiently smooth.

证明. Taylor 展开得到

$$K_1 = f(u_n) + f'(u_n)\cdot\gamma\Delta t K_0 + f''(\theta_1)\cdot\frac{(\gamma\Delta t K_0)^2}{2}, \tag{31}$$

$$K_2 = f(u_n) + f'(u_n)\cdot(1-\gamma)\Delta t K_1 + f''(\theta_2)\cdot\frac{((1-\gamma)\Delta t K_1)^2}{2}, \tag{32}$$

所以

$$u_{n+1} = u_n + \frac{\Delta t}{2}\left(f(u_n) + (f'(u_n)\cdot(1-\gamma)\Delta t + 1)K_1 + f''(\theta_2)\cdot\frac{((1-\gamma)\Delta t K_1)^2}{2}\right) \tag{33}$$

其中

$$\begin{aligned} &(f'(u_n)\cdot(1-\gamma)\Delta t + 1)K_1 \\ =&(f'(u_n)\cdot(1-\gamma)\Delta t + 1)\left(f(u_n) + f'(u_n)\cdot\gamma\Delta t K_0 + f''(\theta_1)\cdot\frac{(\gamma\Delta t K_0)^2}{2}\right) \\ =&f(u_n) + f(u_n)f'(u_n)(\Delta t) + \gamma K_0\cdot\left(\frac{1}{2}f''(\theta_1)(\gamma\Delta t K_0) + (1-\gamma)(f'(u_n))^2\right)(\Delta t)^2 \\ &\quad + \mathcal{O}((\Delta t)^3) \end{aligned}$$

同第五题的 Taylor 展开, 代入有

$$e_{n+1} = u(t_{n+1}) - u_{n+1} = \mathcal{O}((\Delta t)^2). \tag{34}$$

□

附录

A　代码与结果

A.1　1

代码

```
void Question1(){
    printf("=========== Question 1 ===========\n");
    double N = 5.0, dt;
    double u0=0.3, u1, u2, ut;
    for(int i=0;i<4;i++){
        // Delta t
        dt = 1.0/N;



        u1 = u0;
        for(int j=0;j<N;j++){
            u1 += dt*f1(u1);
        }
        printf("%f",u1);

        //Backward Euler
        u1 = u0;
        for(int j=0;j<N;j++){
            ut = u1 + dt * f1(u1);
            u2 = u1 + dt * f1(ut);
            while(fabs(u2-ut)>EPS8){
                ut = u2;
                u2 = u1 + dt * f1(ut);
            }
            u1 = u2;
        }
        printf("   %f",u2);

        // Trapezoidal Method
        u1 = u0;
        for(int j=0;j<N;j++){
            ut = u1 + dt * f1(u1);
            u2 = u1 + 0.5 * dt * ( f1(u1) + f1(ut));
            while(fabs(u2-ut)>EPS8){
                ut = u2;
                u2 = u1 + 0.5 * dt * ( f1(u1) + f1(ut));
            }
            u1 = u2;
        }
        printf("   %f",u2);

        // double N: 5->10->20->40
        N = 2.0*N;
        printf("\n");
    }
}
```

结果

```
0.311522   0.312143   0.311824
0.311669   0.311979   0.311822
0.311745   0.311900   0.311822
0.311783   0.311860   0.311822
```

A.2 2

代码

```
void Question2(){
    printf("\n\n\n=========== Question 2 ===========\n");
    double N=5.0, dt, t;
    double u0 = 1.0, u1, u2;
    double k1,k2,k3,k4;
    double trueValue = sqrt(3.0);
    for(int i=0;i<4;i++){
        // Delta t
        dt = 1.0/N;

        // initial u
        u1 = u0;
        // u1: 0->1
        for(int j=0;j<N;j++){
            t  = dt*j; // Not j+1
            k1 = f2(t,u1);
            k2 = f2(t+0.5*dt,u1+0.5*dt*k1);
            k3 = f2(t+0.5*dt,u1+0.5*dt*k2);
            k4 = f2(t+dt,u1+dt*k3);
            u2 = u1 + dt*(k1+2.0*k2+2.0*k3+k4)/6.0;
            u1 = u2;
            //printf("k1=%f, k2=%f, k3=%f, k4=%f, u=%f\n",k1,k2,k3,k4,u2);
        }

        printf("%f, %e",u2,u2-trueValue);

        // double N: 5->10->20->40
        N = 2.0*N;
        printf("\n");
    }
}
```

结果

```
1.732142, 9.107512e-05
1.732056, 5.557597e-06
1.732051, 3.405711e-07
1.732051, 2.103596e-08
```

A.3 3

代码

```
void Question3(){
    printf("\n\n\n=========== Question 3 ===========\n");

    int N = 10, iter;
    double dt,t1,t2;
    double *trueValue;
    double u0=1.0,u1,u2,ut;
    double k1,k2,k3,k4;

    for(int i=0;i<3;i++){
        printf("--- N=%d ---\n",N);
        dt = 1.0/N;
        trueValue = (double*)malloc(sizeof(double)*(N+1));
        trueValue[0] = 1.0;
        for(int j=1;j<=N;j++){
            trueValue[j] = exp(-(double)j/N);
        }
        // Forward Euler
        printf("Forward Euler\n");
        u1 = u0;
        for(int j=0;j<N;j++){     // 0,1,2,...,N-1
            t1  = j*dt;
            t2  = (j+1)*dt;
            u2 = u1 + dt * f3(t1,u1);
            u1 = u2;
            printf("%f,%f,%e\n",t2,u2,u2-trueValue[j+1]);
        }
        // Backward Euler
        printf("Backward Euler\n");
        u1 = u0;
        for(int j=0;j<N;j++){      // 0,1,2,...,N-1
            t1 = j*dt;
            t2 = (double)(j+1)*dt;  // 1,2,3,...,N
            ut = u1 + dt * f3(t1,u1);
            u2 = u1 + dt * f3(t2,ut);
            iter = 0;
            while(fabs(u2-ut)>EPS8 && iter<MAXITER){
                ut = u2;
                u2 = u1 + dt * f3(t2,ut);
                iter++;
            }
            if(iter == MAXITER){
                printf("Error MAX Iterate ---");
            }
            printf("%f,%f,%e\n",t2,u2,u2-trueValue[j+1]);
            u1 = u2;
        }
        // Trapezoidal Method
        u1 = u0;
        for(int j=0;j<N;j++){     // 0,1,2,...,N-1
            t1 = j*dt;
            t2 = (j+1)*dt;
            ut = u1 + dt * f3(t1,u1);
            u2 = u1 + 0.5 * dt * ( f3(t1,u1) + f3(t2,ut));
            iter = 0;
            while(fabs(u2-ut)>EPS8 && iter<MAXITER){
                ut = u2;
                u2 = u1 + 0.5 * dt * ( f3(t1,u1) + f3(t2,ut));
                iter++;
            }
            if(iter == MAXITER){
                printf("Error MAX Iterate ---");
            }
            printf("%f,%f,%e\n",t2,u2,u2-trueValue[j+1]);
            u1 = u2;
        }
        // Runge-Kutta
        printf("Runge-Kutta\n");
        u1 = u0;
        for(int j=0;j<N;j++){
            t1  = dt*j;
            t2  = dt*(j+1);
            k1 = f3(t1,u1);
            k2 = f3(t1+0.5*dt,u1+0.5*dt*k1);
            k3 = f3(t1+0.5*dt,u1+0.5*dt*k2);
            k4 = f3(t1+dt,u1+dt*k3);
            u2 = u1 + dt*(k1+2.0*k2+2.0*k3+k4)/6.0;
            printf("%f,%f,%e\n",t2,u2,u2-trueValue[j+1]);
            u1 = u2;
            //printf("k1=%f, k2=%f, k3=%f, k4=%f, u=%f\n",k1,k2,k3,k4,u2);
        }

        N = 2*N; // 10->20->40
        printf("\n\n\n");
    }
```

```
}
```

结果

```
--- N=10 ---
Forward Euler
0.100000,0.900000,-4.837418e-03
0.200000,0.814354,-4.377077e-03
0.300000,0.736858,-3.960543e-03
0.400000,0.666736,-3.583647e-03
0.500000,0.603288,-3.242618e-03
0.600000,0.545878,-2.934042e-03
0.700000,0.493930,-2.654831e-03
0.800000,0.446927,-2.402191e-03
0.900000,0.404396,-2.173592e-03
1.000000,0.365913,-1.966747e-03
Backward Euler
Error MAX Iterate ---0.100000,0.914354,9.516258e-03
Error MAX Iterate ---0.200000,0.836858,1.812692e-02
Error MAX Iterate ---0.300000,0.766736,2.591818e-02
Error MAX Iterate ---0.400000,0.703288,3.296800e-02
Error MAX Iterate ---0.500000,0.645878,3.934693e-02
Error MAX Iterate ---0.600000,0.593930,4.511884e-02
Error MAX Iterate ---0.700000,0.546927,5.034147e-02
Error MAX Iterate ---0.800000,0.504396,5.506710e-02
Error MAX Iterate ---0.900000,0.465913,5.934303e-02
Error MAX Iterate ---1.000000,0.431091,6.321206e-02
0.100000,0.904785,-5.285701e-05
0.200000,0.818665,-6.544601e-05
0.300000,0.740753,-6.509101e-05
0.400000,0.670259,-6.085445e-05
0.500000,0.606475,-5.572050e-05
0.600000,0.548761,-5.063703e-05
0.700000,0.496539,-4.589129e-05
0.800000,0.449287,-4.154850e-05
0.900000,0.406532,-3.760275e-05
1.000000,0.367845,-3.402708e-05
Runge-Kutta
0.100000,0.904937,9.922886e-05
```

```
0.200000,0.818858,1.269968e-04
0.300000,0.740947,1.288655e-04
0.400000,0.670442,1.218351e-04
0.500000,0.606643,1.122033e-04
0.600000,0.548914,1.022616e-04
0.700000,0.496678,9.280604e-05
0.800000,0.449413,8.407786e-05
0.900000,0.406646,7.611560e-05
1.000000,0.367948,6.888679e-05



--- N=20 ---
Forward Euler
0.050000,0.950000,-1.229425e-03
0.100000,0.903053,-1.784177e-03
0.150000,0.858703,-2.004518e-03
0.200000,0.816670,-2.060434e-03
0.250000,0.776764,-2.036785e-03
0.300000,0.738842,-1.975869e-03
0.350000,0.702789,-1.898715e-03
0.400000,0.668504,-1.815718e-03
0.450000,0.635896,-1.731967e-03
0.500000,0.604881,-1.649899e-03
0.550000,0.575379,-1.570633e-03
0.600000,0.547317,-1.494633e-03
0.650000,0.520624,-1.422039e-03
0.700000,0.495232,-1.352835e-03
0.750000,0.471080,-1.286932e-03
0.800000,0.448105,-1.224205e-03
0.850000,0.426250,-1.164518e-03
0.900000,0.405462,-1.107734e-03
0.950000,0.385687,-1.053714e-03
1.000000,0.366877,-1.002326e-03
Backward Euler
0.050000,0.952035,8.060676e-04
0.100000,0.906142,1.304133e-03
0.150000,0.862307,1.598782e-03
```

```
0.200000,0.820490,1.759643e-03
0.250000,0.780634,1.833048e-03
0.300000,0.742668,1.849797e-03
0.350000,0.706518,1.830348e-03
0.400000,0.672108,1.788263e-03
0.450000,0.639361,1.732503e-03
0.500000,0.608200,1.668978e-03
0.550000,0.578551,1.601560e-03
0.600000,0.550344,1.532771e-03
0.650000,0.523510,1.464231e-03
0.700000,0.497982,1.396961e-03
0.750000,0.473698,1.331592e-03
0.800000,0.450597,1.268491e-03
0.850000,0.428623,1.207853e-03
0.900000,0.407719,1.149764e-03
0.950000,0.387835,1.094234e-03
1.000000,0.368921,1.041232e-03
0.050000,0.951221,-8.129255e-06
0.100000,0.904825,-1.261034e-05
0.150000,0.860693,-1.492186e-05
0.200000,0.818715,-1.595003e-05
0.250000,0.778785,-1.622569e-05
0.300000,0.740802,-1.606648e-05
0.350000,0.704672,-1.566219e-05
0.400000,0.670305,-1.512590e-05
0.450000,0.637614,-1.452474e-05
0.500000,0.606517,-1.389829e-05
0.550000,0.576937,-1.326961e-05
0.600000,0.548799,-1.265194e-05
0.650000,0.522034,-1.205259e-05
0.700000,0.496574,-1.147540e-05
0.750000,0.472356,-1.092211e-05
0.800000,0.449319,-1.039325e-05
0.850000,0.427405,-9.888659e-06
0.900000,0.406560,-9.405275e-06
0.950000,0.386732,-8.945910e-06
1.000000,0.367871,-8.509214e-06
Runge-Kutta
```

```
0.050000,0.951233,3.126302e-06
0.100000,0.904842,4.870779e-06
0.150000,0.860714,5.784242e-06
0.200000,0.818737,6.200542e-06
0.250000,0.778807,6.321907e-06
0.300000,0.740824,6.270715e-06
0.350000,0.704694,6.120909e-06
0.400000,0.670326,5.917056e-06
0.450000,0.637634,5.685920e-06
0.500000,0.606536,5.443468e-06
0.550000,0.576955,5.199136e-06
0.600000,0.548817,4.958403e-06
0.650000,0.522051,4.724365e-06
0.700000,0.496590,4.498680e-06
0.750000,0.472371,4.282143e-06
0.800000,0.449333,4.075040e-06
0.850000,0.427419,3.877353e-06
0.900000,0.406573,3.688893e-06
0.950000,0.386745,3.509372e-06
1.000000,0.367883,3.338454e-06



--- N=40 ---
Forward Euler
0.025000,0.975000,-3.099120e-04
0.050000,0.950695,-5.346943e-04
0.075000,0.927048,-6.958182e-04
0.100000,0.904028,-8.093825e-04
0.125000,0.881609,-8.874569e-04
0.150000,0.859769,-9.390891e-04
0.175000,0.838486,-9.710605e-04
0.200000,0.817742,-9.884532e-04
0.225000,0.797521,-9.950744e-04
0.250000,0.777807,-9.937756e-04
0.275000,0.758585,-9.866914e-04
0.300000,0.739843,-9.754191e-04
0.325000,0.721566,-9.611528e-04
```

```
0.350000,0.703743,-9.447845e-04
0.375000,0.686362,-9.269797e-04
0.400000,0.669412,-9.082340e-04
0.425000,0.652881,-8.889157e-04
0.450000,0.636759,-8.692979e-04
0.475000,0.621035,-8.495821e-04
0.500000,0.605701,-8.299162e-04
0.525000,0.590745,-8.104083e-04
0.550000,0.576159,-7.911364e-04
0.575000,0.561933,-7.721560e-04
0.600000,0.548058,-7.535060e-04
0.625000,0.534526,-7.352128e-04
0.650000,0.521328,-7.172936e-04
0.675000,0.508457,-6.997584e-04
0.700000,0.495903,-6.826125e-04
0.725000,0.483659,-6.658572e-04
0.750000,0.471717,-6.494909e-04
0.775000,0.460070,-6.335102e-04
0.800000,0.448711,-6.179103e-04
0.825000,0.437632,-6.026852e-04
0.850000,0.426827,-5.878282e-04
0.875000,0.416289,-5.733322e-04
0.900000,0.406010,-5.591897e-04
0.925000,0.395986,-5.453931e-04
0.950000,0.386209,-5.319347e-04
0.975000,0.376674,-5.188067e-04
1.000000,0.367373,-5.060015e-04
Backward Euler
0.025000,0.975556,2.458716e-04
0.050000,0.951666,4.364983e-04
0.075000,0.928327,5.830789e-04
0.100000,0.905532,6.945689e-04
0.125000,0.883275,7.781289e-04
0.150000,0.861547,8.394840e-04
0.175000,0.840340,8.832133e-04
0.200000,0.819644,9.129717e-04
0.225000,0.799448,9.316823e-04
0.250000,0.779742,9.416806e-04
```

```
0.275000,0.760517,9.448317e-04
0.300000,0.741761,9.426248e-04
0.325000,0.723464,9.362482e-04
0.350000,0.705615,9.266496e-04
0.375000,0.688204,9.145845e-04
0.400000,0.671221,9.006545e-04
0.425000,0.654655,8.853382e-04
0.450000,0.638497,8.690159e-04
0.475000,0.622737,8.519892e-04
0.500000,0.607365,8.344971e-04
0.525000,0.592372,8.167281e-04
0.550000,0.577749,7.988309e-04
0.575000,0.563486,7.809220e-04
0.600000,0.549575,7.630924e-04
0.625000,0.536007,7.454127e-04
0.650000,0.522774,7.279373e-04
0.675000,0.509867,7.107076e-04
0.700000,0.497279,6.937547e-04
0.725000,0.485002,6.771014e-04
0.750000,0.473027,6.607642e-04
0.775000,0.461349,6.447543e-04
0.800000,0.449958,6.290787e-04
0.825000,0.438849,6.137415e-04
0.850000,0.428014,5.987440e-04
0.875000,0.417446,5.840857e-04
0.900000,0.407139,5.697643e-04
0.925000,0.397087,5.557766e-04
0.950000,0.387283,5.421182e-04
0.975000,0.377721,5.287843e-04
1.000000,0.368395,5.157695e-04
0.025000,0.975309,-1.142900e-06
0.050000,0.951227,-2.003605e-06
0.075000,0.927741,-2.645520e-06
0.100000,0.904834,-3.119174e-06
0.125000,0.882493,-3.461362e-06
0.150000,0.860704,-3.701946e-06
0.175000,0.839453,-3.864135e-06
0.200000,0.818727,-3.965966e-06
```

```
0.225000,0.798512,-4.021452e-06
0.250000,0.778797,-4.041479e-06
0.275000,0.759568,-4.034496e-06
0.300000,0.740814,-4.007062e-06
0.325000,0.722523,-3.964267e-06
0.350000,0.704684,-3.910053e-06
0.375000,0.687285,-3.847474e-06
0.400000,0.670316,-3.778893e-06
0.425000,0.653766,-3.706136e-06
0.450000,0.637625,-3.630610e-06
0.475000,0.621882,-3.553398e-06
0.500000,0.606527,-3.475330e-06
0.525000,0.591552,-3.397042e-06
0.550000,0.576946,-3.319016e-06
0.575000,0.562702,-3.241618e-06
0.600000,0.548808,-3.165119e-06
0.625000,0.535258,-3.089723e-06
0.650000,0.522043,-3.015578e-06
0.675000,0.509153,-2.942787e-06
0.700000,0.496582,-2.871424e-06
0.725000,0.484322,-2.801535e-06
0.750000,0.472364,-2.733148e-06
0.775000,0.460701,-2.666275e-06
0.800000,0.449326,-2.600918e-06
0.825000,0.438232,-2.537070e-06
0.850000,0.427412,-2.474716e-06
0.875000,0.416860,-2.413838e-06
0.900000,0.406567,-2.354413e-06
0.925000,0.396529,-2.296418e-06
0.950000,0.386739,-2.239824e-06
0.975000,0.377190,-2.184604e-06
1.000000,0.367877,-2.130729e-06
Runge-Kutta
0.025000,0.975310,9.809751e-08
0.050000,0.951230,1.720746e-07
0.075000,0.927744,2.273264e-07
0.100000,0.904838,2.680531e-07
0.125000,0.882497,2.975244e-07
```

```
0.150000,0.860708,3.182853e-07
0.175000,0.839457,3.323166e-07
0.200000,0.818731,3.411597e-07
0.225000,0.798517,3.460135e-07
0.250000,0.778801,3.478108e-07
0.275000,0.759572,3.472764e-07
0.300000,0.740819,3.449740e-07
0.325000,0.722528,3.413411e-07
0.350000,0.704688,3.367175e-07
0.375000,0.687290,3.313667e-07
0.400000,0.670320,3.254926e-07
0.425000,0.653770,3.192531e-07
0.450000,0.637628,3.127703e-07
0.475000,0.621885,3.061379e-07
0.500000,0.606531,2.994282e-07
0.525000,0.591556,2.926964e-07
0.550000,0.576950,2.859846e-07
0.575000,0.562705,2.793246e-07
0.600000,0.548812,2.727403e-07
0.625000,0.535262,2.662496e-07
0.650000,0.522046,2.598653e-07
0.675000,0.509157,2.535967e-07
0.700000,0.496586,2.474503e-07
0.725000,0.484325,2.414302e-07
0.750000,0.472367,2.355389e-07
0.775000,0.460704,2.297777e-07
0.800000,0.449329,2.241468e-07
0.825000,0.438235,2.186455e-07
0.850000,0.427415,2.132727e-07
0.875000,0.416862,2.080270e-07
0.900000,0.406570,2.029063e-07
0.925000,0.396532,1.979087e-07
0.950000,0.386741,1.930317e-07
0.975000,0.377193,1.882731e-07
1.000000,0.367880,1.836303e-07
```

A.4 4

代码为

```
void Question4(){
    printf("\n\n\n=========== Question 4 ===========\n");

    double dt = 0.05;
    int N = 20; // N*dt=1.0
    double x0=PI/6.0,y0=0.0; // x:theta, y:theta prime
    double x1,y1,x2,y2,xt,yt;

    // Forward Euler
    printf("Forward Euler\n");
    x1 = x0;
    y1 = y0;
    printf("%f,%f,%f\n",0.0,x1,y1);
    for(int j=0;j<N;j++){
        x2 = x1 + y1 * dt;
        y2 = y1 - 16.0 * sin(x1) *dt;
        x1 = x2;
        y1 = y2;
        printf("%f,%f,%f\n",dt*(j+1),x1,y1);
    }

    // Backward Euler
    printf("Backward Euler\n");
    x1 = x0;
    y1 = y0;
    printf("%f,%f,%f\n",0.0,x1,y1);
    for(int j=0;j<N;j++){
        xt = x1 + y1 * dt;
        yt = y1 - 16.0 * sin(x1) * dt;
        x2 = x1 + yt * dt;
        y2 = y1 - 16.0 * sin(xt) * dt;
        while(fabs(xt-x2)>EPS8 || fabs(yt-y2)>EPS8){
            xt = x2;
            yt = y2;
            x2 = x1 + yt * dt;
            y2 = y1 - 16.0 * sin(xt) * dt;
        }
        x1 = x2;
        y1 = y2;
        printf("%f,%f,%f\n",dt*(j+1),x1,y1);
    }

    // Trapezoidal Method
    printf("Trapezoidal Method\n");
    x1 = x0;
    y1 = y0;
    printf("%f,%f,%f\n",0.0,x1,y1);
    for(int j=0;j<N;j++){
        xt = x1 + y1 * dt;
        yt = y1 - 16.0 * sin(x1) * dt;
        x2 = x1 + 0.5 * (y1+yt) * dt;
        y2 = y1 - 8.0 * (sin(x1) + sin(xt)) * dt;
        while(fabs(xt-x2)>EPS8 || fabs(yt-y2)>EPS8){
            xt = x2;
            yt = y2;
            x2 = x1 + 0.5 * (y1+yt) * dt;
        y2 = y1 - 8.0 * (sin(x1) + sin(xt)) * dt;
        }
        x1 = x2;
        y1 = y2;
        printf("%f,%f,%f\n",dt*(j+1),x1,y1);
    }

}
```

结果如下

Forward Euler

```
0.000000,0.523599,0.000000
0.050000,0.523599,-0.400000
0.100000,0.503599,-0.800000
0.150000,0.463599,-1.186065
0.200000,0.404296,-1.543800
0.250000,0.327106,-1.858497
```

```
0.300000,0.234181,-2.115540
0.350000,0.128404,-2.301177
0.400000,0.013345,-2.403618
0.450000,-0.106836,-2.414293
0.500000,-0.227551,-2.328987
0.550000,-0.344000,-2.148513
0.600000,-0.451426,-1.878709
0.650000,-0.545361,-1.529710
0.700000,-0.621847,-1.114728
0.750000,-0.677583,-0.648698
0.800000,-0.710018,-0.147169
0.850000,-0.717376,0.374309
0.900000,-0.698661,0.900237
0.950000,-0.653649,1.414791
1.000000,-0.582910,1.901261
```

Backward Euler

```
0.000000,0.523599,0.000000
0.050000,0.504272,-0.386536
0.100000,0.466939,-0.746660
0.150000,0.413532,-1.068137
0.200000,0.346539,-1.339853
0.250000,0.268919,-1.552405
0.300000,0.183981,-1.698761
0.350000,0.095239,-1.774837
0.400000,0.006248,-1.779835
0.450000,-0.079565,-1.716250
0.500000,-0.159043,-1.589552
0.550000,-0.229424,-1.407619
0.600000,-0.288427,-1.180064
0.650000,-0.334305,-0.917573
0.700000,-0.365873,-0.631361
0.750000,-0.382511,-0.332760
0.800000,-0.384158,-0.032937
0.850000,-0.371292,0.257319
0.900000,-0.344902,0.527803
0.950000,-0.306445,0.769140
1.000000,-0.257790,0.973096
```

Trapezoidal Method

```
0.000000,0.523599,0.000000
0.050000,0.513685,-0.396556
0.100000,0.484287,-0.779343
0.150000,0.436437,-1.134660
0.200000,0.371844,-1.449078
0.250000,0.292870,-1.709892
0.300000,0.202477,-1.905811
0.350000,0.104136,-2.027829
0.400000,0.001688,-2.070083
0.450000,-0.100826,-2.030496
0.500000,-0.199364,-1.911016
0.550000,-0.290074,-1.717388
0.600000,-0.369472,-1.458529
0.650000,-0.434577,-1.145669
0.700000,-0.483006,-0.791481
0.750000,-0.513027,-0.409377
0.800000,-0.523588,-0.013054
0.850000,-0.514321,0.383720
0.900000,-0.485549,0.767175
0.950000,-0.438280,1.123606
1.000000,-0.374200,1.439570
```