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

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

Δt	1/5	1/10	1/20	1/40
$w_{\Delta t}$	0.311522	0.311669	0.311745	0.311783
$w_{\Delta t} - w_{2\Delta t}$	\	1.472262e-04	7.553923e-05	3.826791e-05

后向 Euler

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

Δt	1/5	1/10	1/20	1/40
$w_{\Delta t}$	0.312143	0.311979	0.311900	0.311783
$w_{\Delta t} - w_{2\Delta t}$	\	-1.637725e-04	-7.966938e-05	-3.930098e-05

梯形

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

Δt	1/5	1/10	1/20	1/40
$w_{\Delta t}$	0.311824	0.311822	0.311822	0.311822
$w_{\Delta t} - w_{2\Delta t}$	\	-2.075604e-06	-5.184184e-07	-1.299643e-07

从表中可以看出 Forward Euler 和 Backward Euler 都是 $\mathcal{O}(\Delta t)$, 而梯形法则是 $\mathcal{O}((\Delta t)^2)$.

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
$e(\Delta t)$	9.107512×10^{-5}	5.557597×10^{-6}	3.405711×10^{-7}	2.103596×10^{-8}
$e(2\Delta t)/e(\Delta t)$	\	16.3929	16.3182	16.1899

所以误差为 $\mathcal{O}((\Delta t)^4)$.

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.

N	10	20	40
Forward Euler	-1.966747e-03	-1.002326e-03	-5.060015e-04
Backward Euler	\	1.041232e-03	5.157695e-04
Trapezoidal	-3.402708e-05	-8.509214e-06	-2.130729e-06
Runge-Kutta	6.888679e-05	3.338454e-06	1.836303e-07

表 1: N with solution error

解. 各种方法在 $t=1$ 处得到的结果减真值如表1.

可以看出 Euler 方法误差约为 $\mathcal{O}(\Delta t)$, 梯形法为 $\mathcal{O}((\Delta t)^2)$, 而 Runge-Kutta 方法至少为 $\mathcal{O}((\Delta t)^4)$.

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

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

得到结果如图1所示.

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.

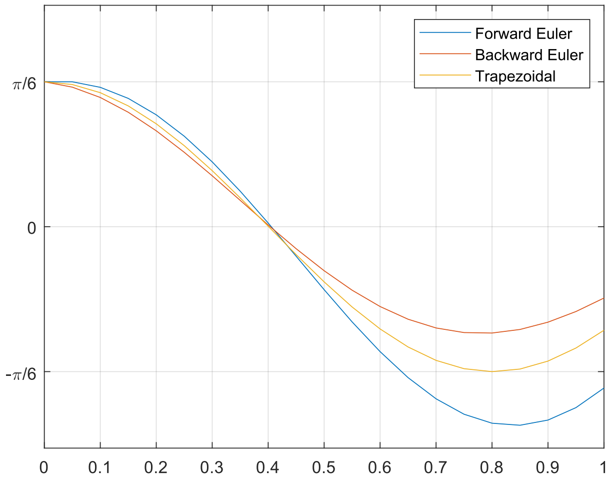

图 1: Plot of Numerical Solution

(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(){
    double N = 5.0, dt;
    double u0=0.3, u1, u2, ut;
    double ul[3];
    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);
        if(i>0){
            printf(",%e", u1-ul[0]);
        }
        ul[0] = 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);
        if(i>0){
```

```
        printf(",%e", u2-ul[1]);
    }
    ul[1] = 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);
    if(i>0){
        printf(",%e", u2-ul[2]);
    }
    ul[2] = u2;

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

A.2 2

代码

```
void Question2(){
    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");
    }
}
```

A.3 3

代码

```
void Question3(){
    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;
```

```c
            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");
    }
}
```

A.4 4

代码为

```
void Question4(){
    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
```