数值试验三

董建宇　2019511017

6 月 4 日

1 体会数值方法不稳定对数值结果的影响

求解 y'=50y,y(0)=100。

1.1 显式欧拉公式

显式欧拉公式为:

$$y_{n+1} = y_n - 50hy_n.$$

理论推导：由显示欧拉公式可以得到误差方程为

$$\epsilon_{n+1} = (1 - 50h)\epsilon_n.$$

为使误差不增加，要求

$$|\epsilon_{n+1}| < |\epsilon_n|.$$

即有:

$$|1 - 50h| < 1.$$

所以步长 h 取值范围为 (0,0.04)。接下来进行数值计算。

当 h=0.01 时,迭代 50 次 y_n 数据如下 (保留四位小数):100,50,25,12.5,6.25,3.125,1.5625,0.7813,0.3906,0.1953, 0.0977,0.0488,0.0244,0.0122,0.0061,0.0031,0.0015,0.0008,0.0004,0.0002,0.0001,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0, 0,0,0,0,0,0,0,0,0,0,0,0,0,0.

当 h=0.02 时,迭代 50 次 y_n 数据如下 (保留四位小数):100,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0, 0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0.

当 h=0.03 时, 迭代 50 次 y_n 数据如下 (保留四位小数): 100,-50,25,-12.5,6.25,-3.125,1.5625,-0.7813,0.3906,-0.1953,0.0977,-0.0488,0.0244,-0.0122,0.0061,-0.0031,0.0015,-0.0008,0.0004,-0.0002,0.0001,-0,0,-0,0,-0,0,-0,0, -0,0,-0,0,-0,0,-0,0,-0,0,-0,0,-0,0,-0,0,-0,0,-0,0,-0,0.

当 h=0.04 时, 迭代 50 次 y_n 数据如下 (保留四位小数): 100,-100,100,-100,100,-100,100,-100,100,-100,100,-100,100,-100,100,-100,100,-100,100,-100,100,-100,100,-100,100,-100,100,-100,100,-100,100,-100,100,-100,100,-100,100, -100,100,-100,100,-100,100,-100,100,-100,100,-100,100,-100,100,-100,100.

当 h=0.05 时，迭代 50 次 y_n 数据如下 (保留四位小数): 100,-150,225,-337.5,506.25,-759.375,1139.0625,-1708.5938,2562.8906,-3844.3359,5766.5039,-8649.7559,12974.6338,-19461.9507,29192.926,-43789.389,65684.0836, -98526.1253,147789.188,-221683.782,332525.673,-498788.5095,748182.7643,-1122274.1464,1683411.2196, -2525116.8294,3787675.2441,-5681512.8662,8522269.2992,-12783403.9489,19175105.9233,-28762658.8849, 43143988.3274,-64715982.4911,97073973.7366,-145610960.605,218416440.9075,-327624661.3612,491436992.0418, -737155488.0627,1105733232.094,-1658599848.141,2487899772.2115,-3731849658.3173,5597774487.4759, -8396661731.2138,12594992596.8207,-18892488895.2311,28338733342.8467,-42508100014.27,63762150021.405

即当 $h \geqslant 0.04$ 时开始出现数值不稳定。

Matlab 脚本程序如下:

```
y=ones(1,51);
h=0.01;
y(1)=100;
for i=1:51
    y(i+1)=y(i)-50*h*y(i); % 显式欧拉公式
end
x=[];
for i=1:51
    x(i)=roundn(y(i),-4);% 当小数位数过多时，保留四位小数
end
```

1.2 隐式欧拉公式

隐式欧拉公式为:

$$y_{n+1} = y_n - 50hy_{n+1}.$$

从而误差方程为

$$\epsilon_{n+1} = \epsilon_n - 50h\epsilon_{n+1}.$$

即

$$\epsilon_{n+1} = \frac{1}{1+50h}\epsilon_n.$$

要使误差不增大，要求

$$|\epsilon_{n+1}| \leqslant |\epsilon_n|.$$

当 h>0 时，显然有:

$$\frac{1}{1+50h} \leqslant 1.$$

即用隐式欧拉公式求解上述问题不会出现数值不稳定现象。接下来进行数值计算。

当 h=0.05 时，迭代 50 次 y_n 数据如下 (保留四位小数): 100,28.5714,8.1633,2.3324,0.6664,0.1904,0.0544, 0.0155,0.0044,0.0013,0.0004,0.0001,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0.

当 h=0.04 时，迭代 50 次 y_n 数据如下 (保留四位小数): 100,33.3333,11.1111,3.7037,1.2346,0.4115,0.1372, 0.0457,0.0152,0.0051,0.0017,0.0006,0.0002,0.0001,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0, 0,0,0,0,0.

当 h=0.03 时，迭代 50 次 y_n 数据如下 (保留四位小数): 100,40,16,6.4,2.56,1.024,0.4096,0.1638,0.0655, 0.0262,0.0105,0.0042,0.0017,0.0007,0.0003,0.0001,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0, 0,0,0,0

即当 h>0 时，隐式欧拉方法不会出现数值不稳定现象，符合理论计算的预期。

2 体会算法的保结构性

求解一节常微分方程组:

$$\begin{cases} y'(t) = -2x(t), \\ x'(t) = \frac{9}{2}y(t), \\ x(0) = 3,\ y(0) = 2. \end{cases}$$

设 h 为步长，$t_i = ih$,$y_i \approx y(t_i), x_0 = x(0) = 3, y_0 = y(0) = 2$。

2.1 梯形法

利用梯形公式可得:

$$\begin{cases} y_{n+1} = y_n + \frac{h}{2}(-2x_n - 2x_{n+1}) = y_n - h(x_n + x_{n+1}), \\ x_{n+1} = x_n + \frac{h}{2}(\frac{9}{2}y_n + \frac{9}{2}y_{n+1}) = x_n + \frac{9h}{4}(y_n + y_{n+1}). \end{cases}$$

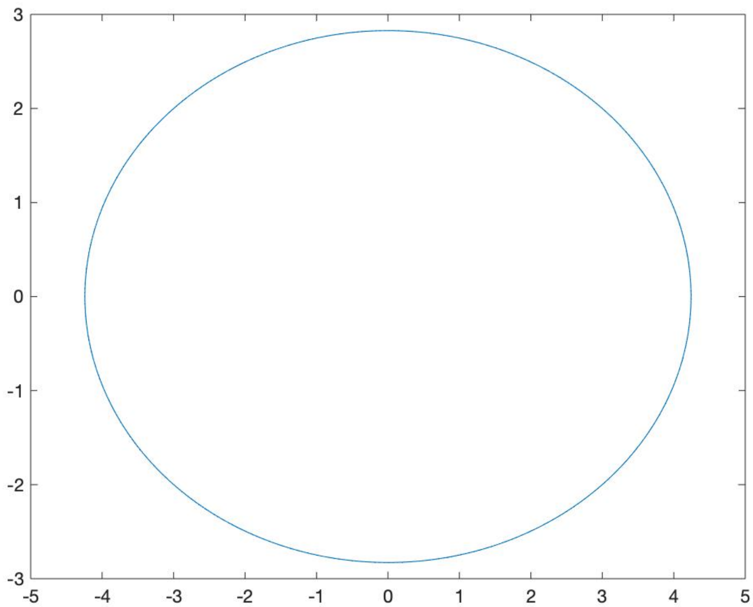
可以解得:

$$\begin{cases} y_{n+1} = \frac{4-9h^2}{4+9h^2}y_n - \frac{8h}{4+9h^2}x_n, \\ x_{n+1} = \frac{4-9h^2}{4+9h^2}x_n + \frac{18h}{4+9h^2}y_n. \end{cases}$$

列举前七次迭代结果如下:

迭代次数 i	0	1	2	3	4	5	6	7
x	3	3.17	3.34	3.49	3.63	3.75	3.86	3.96
y	2	1.88	1.75	1.61	1.47	1.32	1.17	1.01

取 h=0.02, 使用 Matlab 绘图如下:



即使用梯形法求的的结果能保持相平面中轨迹不变。Matlab 脚本程序如下:

```
x=[];y=[];
h=0.02;
x(1)=3;y(1)=2;
for i=1:2000
    x(i+1)=(4-9*h*h)/(4+9*h*h)*x(i)+18*h/(4+9*h*h)*y(i);
    y(i+1)=(4-9*h*h)/(4+9*h*h)*y(i)-8*h/(4+9*h*h)*x(i);
end
plot(x,y)
```

2.2 Eular 法

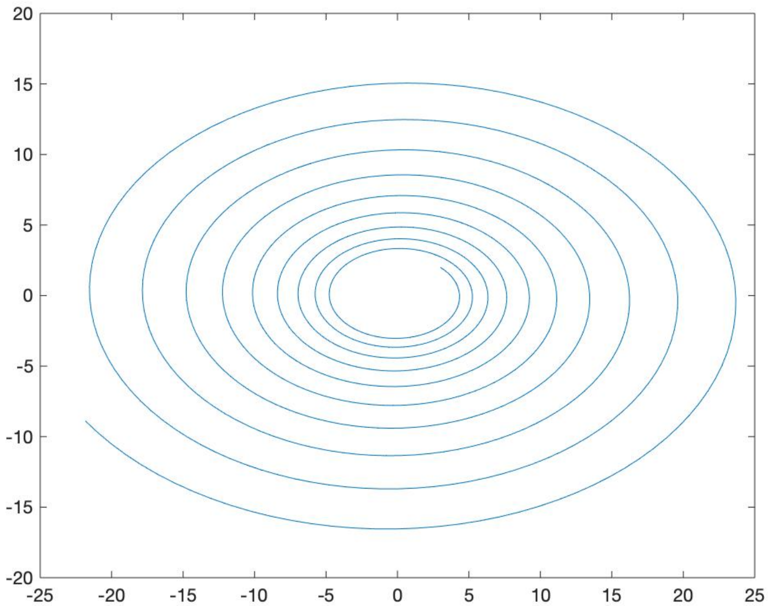
利用显式欧拉公式可得:

$$\begin{cases} y_{n+1} = y_n - 2hx_n, \\ x_{n+1} = x_n + \dfrac{9}{2}hy_n. \end{cases}$$

列举前七次迭代结果如下:

迭代次数 i	0	1	2	3	4	5	6	7
x	3	3.18	3.35	3.51	3.65	3.79	3.91	4.01
y	2	1.88	1.75	1.62	1.48	1.33	1.18	1.02

取 h=0.02, 使用 Matlab 绘图如下:



观察图中点的坐标可知：相平面中轨迹随时间的增加发散到无穷大。

Matlab 脚本程序如下:

```
x=[];y=[];
h=0.02;
x(1)=3;y(1)=2;
for i=1:1000
    x(i+1)=x(i)+9/2*h*y(i);
    y(i+1)=y(i)-2*h*x(i);
end
plot(x,y)
```

当使用隐式欧拉方法时，有:

$$\begin{cases} y_{n+1} = y_n - 2hx_{n+1} \\ x_{n+1} = x_n + \dfrac{9}{2}hy_{n+1} \end{cases}$$

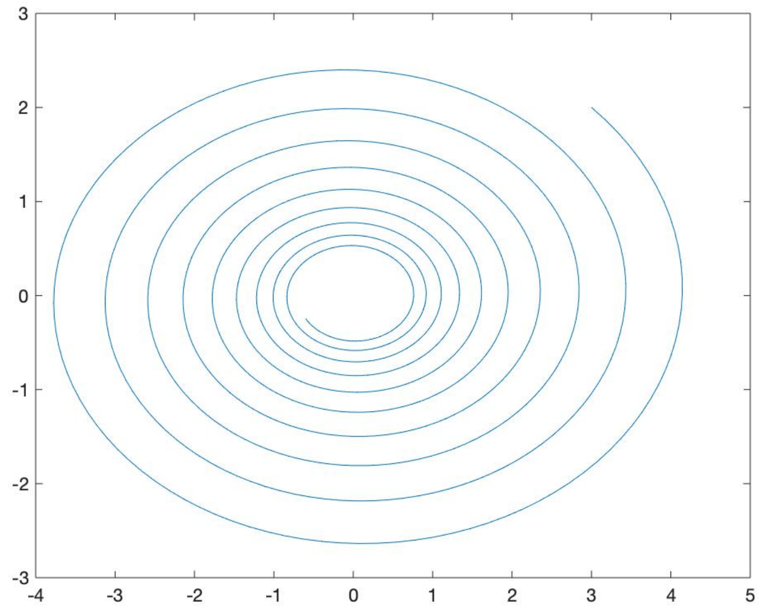
可以解得:

$$\begin{cases} y_{n+1} = \dfrac{y_n - 2hx_n}{9h^2+1} \\ x_{n+1} = \dfrac{9}{2}hy_{n+1} + x_n \end{cases}$$

列举前七次迭代结果如下:

迭代次数 i	0	1	2	3	4	5	6	7
x	3	3.17	3.33	3.47	3.60	3.72	3.82	3.91
y	2	1.87	1.74	1.60	1.46	1.31	1.16	1.00

取 h=0.02, 使用 Matlab 绘图如下:



观察图中点的坐标可知：相平面中轨迹随时间的增加越来越趋向于原点。

Matlab 脚本程序如下:

```
x=[];y=[];
h=0.02;
x(1)=3;y(1)=2;
for i=1:1000
    y(i+1)=(y(i)-2*h*x(i))/(9*h*h+1);
    x(i+1)=9/2*h*y(i+1)+x(i);
end
plot(x,y)
```

2.3 四阶 Runge-Kutta 法

利用四阶 Runge-Kutta 公式可得:

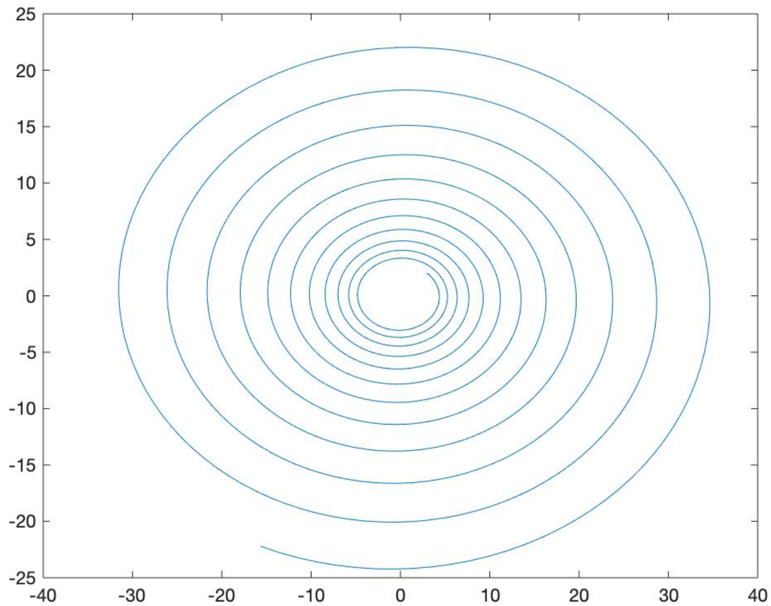
$$\begin{cases} k_{y1} = -2x_n \\ k_{y2} = -2x_n - h \\ k_{y3} = -2x_n - h \\ k_{y4} = -2x_n - 2h \\ y_{n+1} = y_n + \dfrac{h}{6}(k_{y1} + k_{y2} + k_{y3} + k_{y4}) \end{cases}$$

$$\begin{cases} k_{x1} = \dfrac{9}{2}y_n \\ k_{x2} = \dfrac{9}{2}(y_n + \dfrac{h}{2}) \\ k_{x3} = \dfrac{9}{2}(y_n + \dfrac{h}{2}) \\ k_{x4} = \dfrac{9}{2}(y_n + h) \\ x_{n+1} = x_n + \dfrac{h}{6}(k_{x1} + k_{x2} + k_{x3} + k_{x4}) \end{cases}$$

列举前七次迭代结果如下:

迭代次数 i	0	1	2	3	4	5	6	7
x	3	3.18	3.35	3.51	3.66	3.79	3.91	4.01
y	2	1.88	1.75	1.62	1.48	1.33	1.18	1.02

取 h=0.02, 使用 Matlab 绘图如下:



观察图中点的坐标可知：相平面中轨迹随时间的增加发散到无穷大。

Matlab 脚本程序如下:

```
x=[];y=[];
h=0.02;
x(1)=3;y(1)=2;
for i=1:1200
    k1=-2*x(i);
    k2=-2*x(i)-h;
    k3=-2*x(i)-h;
    k4=-2*x(i)-2*h;
    y(i+1)=y(i)+h/6*(k1+2*k2+2*k3+k4);
    k5=9/2*y(i);
    k6=9/2*(y(i)+h/2);
    k7=9/2*(y(i)+h/2);
    k8=9/2*(y(i)+h);
    x(i+1)=x(i)+h/6*(k5+2*k6+2*k7+k8);
end
plot(x,y)
```

综上所述，使用梯形方法求得的结果能保持相平面中轨迹不变；使用隐式欧拉方法求得的结果在相平面中的轨迹趋向于原点；使用显式欧拉方法和 Runge-Kutta 方法求得的结果在相平面中的轨迹趋向于无穷远。

3 体会非线性方程的迭代求解

令 $y=\frac{d\theta}{dt}$，则原方程可化为:

$$\begin{cases}\dfrac{dy}{dt}=-\sin\theta\\ \dfrac{d\theta}{dt}=y\\ y(0)=-\dfrac{1}{2}\\ \theta(0)=\dfrac{\pi}{3}\end{cases}$$

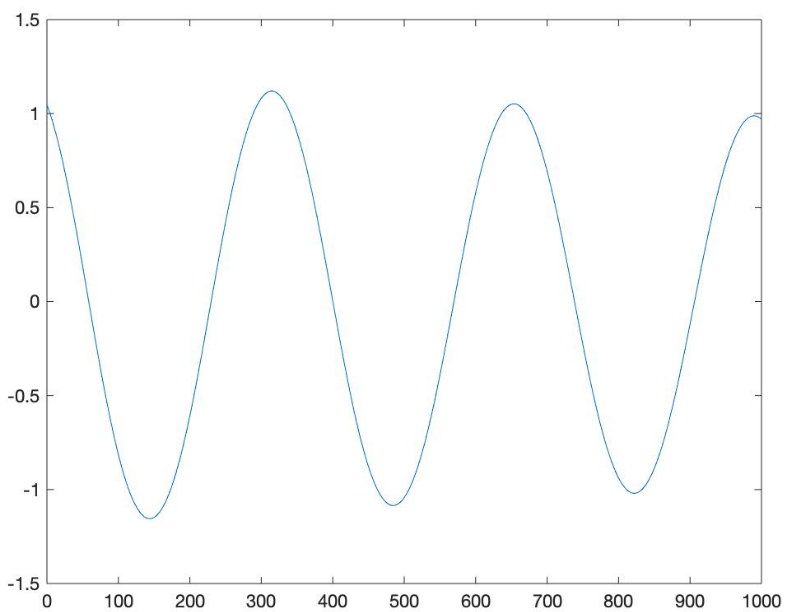
3.1 隐式欧拉公式

利用隐式欧拉公式得:

$$\begin{cases}y_{n+1}=y_n-h\sin\theta_{n+1}\\ \theta_{n+1}=\theta_n+hy_{n+1}\end{cases}$$

由于方程组为非线性方程组, 因而采用迭代法求解 θ_{n+1}。θ_{n+1} 满足的方程为 $\theta_{n+1}=\theta_n+hy_n-h^2\sin\theta_{n+1}$。令 $\varphi(x)=\theta_n+hy_n-h^2\sin x$，$x_0=1$ 迭代格式为 $x_{k+1}=\varphi(x_k)$, 停止迭代条件为 $|x_{k+1}-x_k|<\epsilon$。

绘图如下:



Matlab 脚本程序如下:

```
y=ones(1,1000);
theta=ones(1,1000);
h=0.02;
epslion=10^(-6);
y(1)=-1/2;theta(1)=pi/3;
for i=1:1000
    x=[];
    x(1)=1;
    for k=1:50000
        x(k+1)=theta(i)+h*y(i)-h*h*sin(x(k));
```

```
            if abs(x(k+1)-x(k))<epslion
                theta(i+1)=x(k+1);
                break;
            end
        end
        y(i+1)=y(i)-h*sin(theta(i+1));
    end
    t=[];
    t(1)=0;
    for i=1:1000
        t(i+1)=t(i)+1;
    end
    plot(t,theta)
```

3.2 梯形公式

利用梯形公式可得:

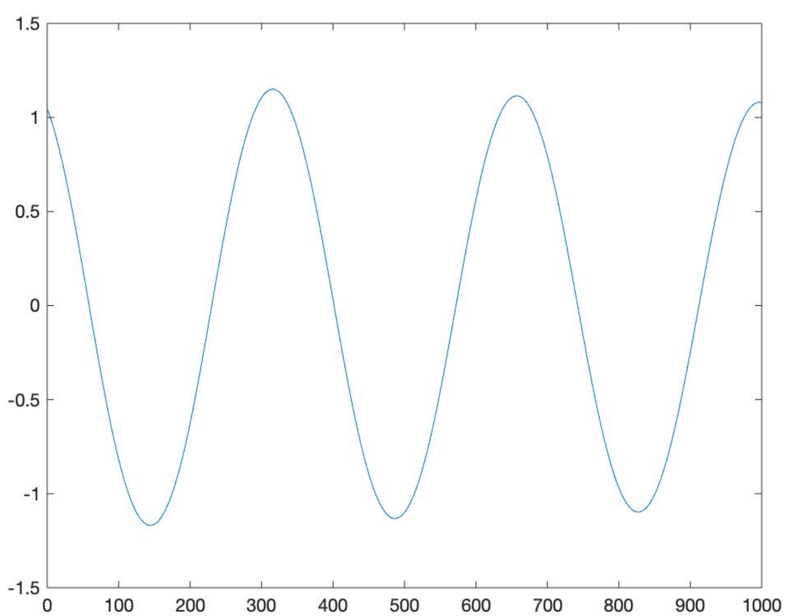
$$\begin{cases} y_{n+1} = y_n - \dfrac{h}{2}(\sin\theta_{n+1} + \sin\theta_n) \\ \theta_{n+1} = \theta_n + \dfrac{h}{2}(y_{n+1} + y_n) \end{cases}$$

可以解得:

$$\begin{cases} \theta_{n+1} = \theta_n + hy_n - \dfrac{h^2}{4}(\sin\theta_{n+1} + \sin\theta_n) \\ y_{n+1} = y_n - \dfrac{h}{2}(\sin\theta_{n+1} + \sin\theta_n) \end{cases}$$

由于 θ_{n+1} 满足的方程为非线性方程，所以使用迭代法求解。令 $\varphi(x) = \theta_n + hy_n - \frac{h^2}{4}(\sin x + \sin\theta_n)$，则迭代格式为 $x_{k+1} = \varphi(x_k)$，停止迭代条件为 $|x_{k+1} - x_k| < \epsilon$.

绘图如下:

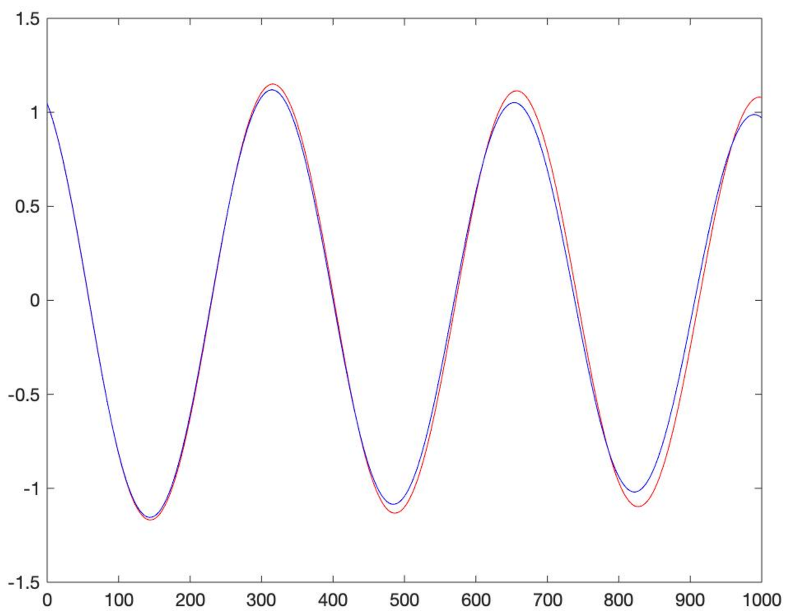


Matlab 脚本程序如下:

```
y=ones(1,1000);
```

```
theta=ones(1,1000);
h=0.02;
epslion=10^(-6);
y(1)=-1/2;theta(1)=pi/3;
for i=1:1000
    x=[];
    x(1)=1;
    for k=1:50000
        x(k+1)=theta(i)+h*y(i)-h*h/4*(sin(x(k))+sin(theta(i)));
        if abs(x(k+1)-x(k))<epslion
            theta(i+1)=x(k+1);
            break;
        end
    end
    y(i+1)=y(i)-h*sin(theta(i+1));
end
t=[];
t(1)=0;
for i=1:1000
    t(i+1)=t(i)+1;
end
plot(t,theta)
```

将两幅图线绘制于一张图中:



其中红线为梯形公式绘制;蓝线为隐式欧拉公式绘制。